rMate Chart 사용 설명서

Version 5.0

정품을 구입하신 고객에게는 기술상담 및 지원을 제공합니다
(Tel : 02-2655-9767, riamore@riamore.net)

㈜리아모어소프트

목 차

1. 개요

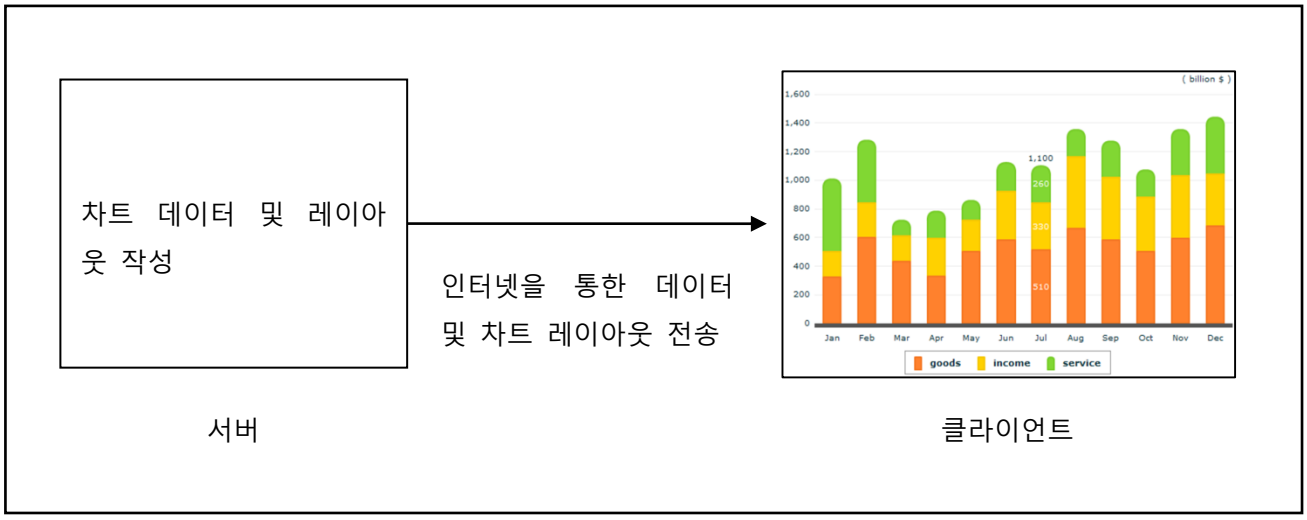
1.1. rMate Chart 의 주요 특징.

rMate Chart 는 Adobe® Flex™ 제품을 사용해 개발된 어플리케이션으로 사용자에게 데이터를 시각적으로 알기 쉽게 제시할 수 있는 솔루션을 제공합니다. 단순한 2 차원이 아닌 3D 와 그라데이션 효과를 넣은 2D 를 통해 수치 데이터를 표현하기 때문에 한 차원 업그레이드 된 UI 를 경험하실 수 있습니다.
HTML, .NET, JSP, PHP, ASP, ColdFusion 등과 같은 웹 스크립트 언어에 구애 받지 않고, 강력하고 화려한 차트를 제공합니다.
http://www.riamore.net 의 데모 메뉴에서 강력하고 화려한 UI 의 차트를 직접 체험하실 수 있습니다.

1.2. 시스템 요구사항.

- 서버 사이드 : rMate Chart 는 톰캣, IIS, 웹로직, 웹스피어, 제우스, LCDS 등등 모든 WAS 서버에서 작동하며, 웹 스크립트 언어에 의존적이지 않습니다.
- 클라이언트 사이드 : **Adobe Flash Player 9 버전 이상의 클라이언트 환경**이 필요합니다.
  (단, 차트 이미지 파일로 저장은 10 버전 이상에서 가능합니다.)

1.3. 차트 시스템 구성 환경 설명.



<그림 1 서버와 클라이언트간의 데이터 흐름도>

rMate 차트를 생성하기 위해서는 차트 레이아웃과 수치 데이터가 필요합니다. 차트 레이아웃은 차트 유형의 선택을 시작으로 차트 컬러 변경, 차트 축 조작 등등 세밀한 부분을 제어하는 XML 파일입니다. 이에 대한 자세한 설명은 "5.차트와 레이아웃."을 참고하십시오.
차트 데이터는 **XML 형식과 배열 객체**를 지원합니다. 사용자의 환경과 편의에 따라 데이터를 작성하실 수 있습니다. 차트 데이터 작성에 대한 자세한 설명은 "3.차트 데이터 형식"을 참고하십시오.

1.4. 차트 5.0 로 업그레이드 하기.

차트 5.0 은 기본적으로 이전 버전의 모든 기능을 수용합니다. 즉, 5.0 차트 swf 와 라이선스 교체만으로 이전 버전의 차트를 5.0 으로 표현할 수 있습니다.
그러나 버전 3.0 프리미엄 라이선스의 게이지는 모든 속성 및 레이아웃이 전반적으로 변경되었습니다. 따라서 제공된 샘플과 함께 제공된 속성 설명표(레이아웃 API)를 토대로 다시 작성하여 주십시오.

1.5. 차트 리스트 및 샘플.

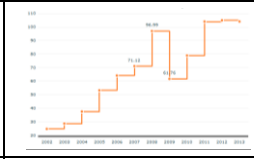
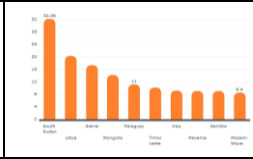
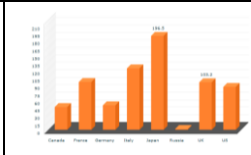
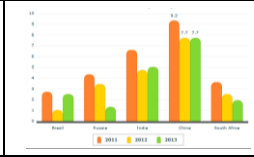
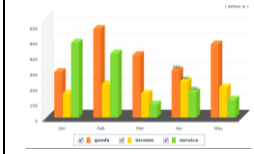
아래 도표는 rMate Chart 리스트를 나타냅니다. 표현하고자 하는 차트 및 데이터 유형에 따라 레이아웃 파일과 swf 파일을 선택하십시오. 제공된 레이아웃 파일은 기본 설정이며 사용자의 취향에 맞게 변경하실 수 있습니다. 이에 대한 자세한 사항은 "5.차트와 레이아웃." 을 참고하십시오.

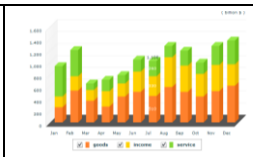
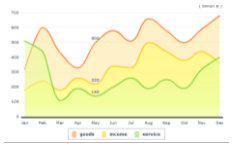
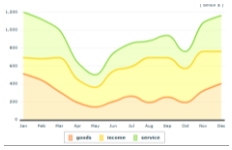
*** 제공된 차트 swf 파일 및 레이아웃 파일은 구매자의 라이선스에 따라 빠진 부분이 있을 수 있습니다.**

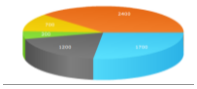
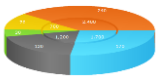
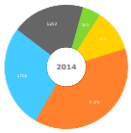
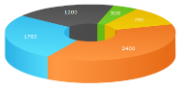
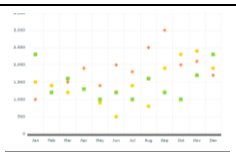
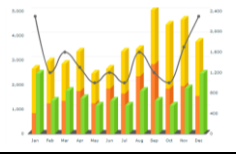
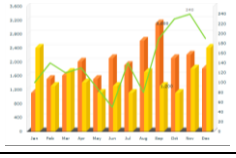
파일명	설 명
rMateChart.swf	칼럼, 바, 파이, 라인, 영역 등의 기본 차트
rMateRadarChart.swf	레이더 차트
rMateHistoryChart.swf	히스토리 차트
rMateRealtimeChart.swf	실시간 차트
rMateScrollChart.swf	스크롤 차트
rMateBrokenChart.swf	브로큰 축 차트
rMateCandleChart.swf	캔들스틱 차트
rMateMatrixChart.swf	매트릭스 차트
rMateImageChart.swf	이미지 차트
rMateWingChart.swf	윙 차트
rMateRealtimePremium.swf	실시간-프리미엄 차트
rMateGaugeChart.swf	게이지 차트
rMateIntegration.swf	해당 라이선스에서 생성 가능한 모든 차트 내포(통합버전)
ChartDataEditor.swf	차트 에디터
ChartSideBar.swf	차트 사이드 바
playerProductInstall.swf	플래시 플레이어 업데이트 인스톨러
Pattern.swf	패턴

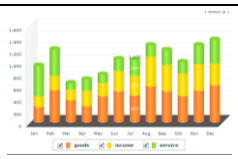
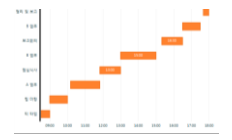
<표 1 제공된 swf 파일 목록>

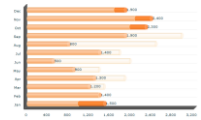
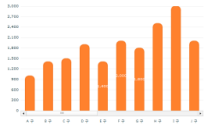
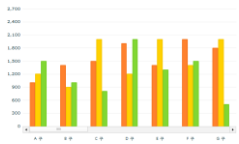
- **플래시 플레이어 자동 업데이트는 playerProductInstall.swf 에 의해 이루어집니다. 사용자의 PC 에 플래시 플레이어가 설치되지 않았거나, 구버전인 경우 자동 업데이트를 실행하길 원하시면 playerProductInstall.swf 를 차트 컴포넌트와 같은 디렉토리에 위치시키십시오.**

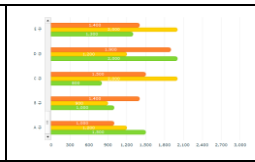
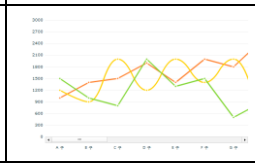
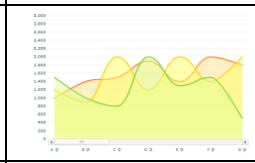
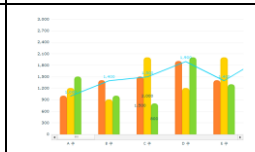
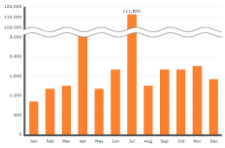
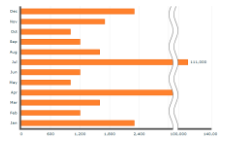
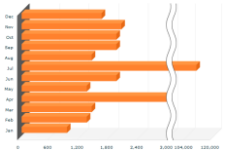
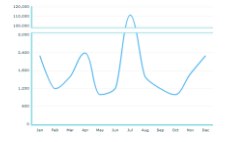
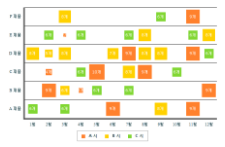
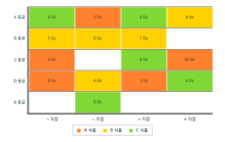
차트 유형	데이터 표현(샘플 Layout 파일명)	샘플 이미지	Swf 파일명
라인 차트	Line Segment		rMateChart.swf
	Line Curve		
	Line Step		
	Line ItemRenderer		
칼럼 차트	Column Chart		rMateChart.swf
	Column 3D Chart		
	Multi Series Column Chart		
	Multi Series Column 3D		
	Stacked Column Chart		

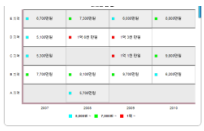
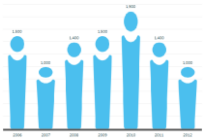
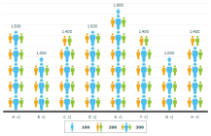
	Stacked Column 3D		
바 차트	Bar 2D		rMateChart.swf
	Bar 3D Chart		
	Multi Series Bar 2D		
	Multi Series Bar 3D		
	Stacked Bar 2D		
	Stacked Bar 3D		
영역 차트	Area Chart		rMateChart.swf
	Multi Series Area Chart		
	Stacked Area Chart		
파이 차트	Pie 2D		rMateChart.swf

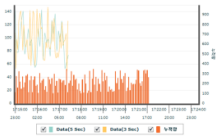
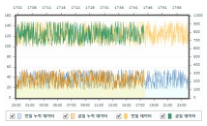
	Pie 3D		
	Stacked 3D		
도넛 차트	Doughnut 2D		rMateChart.swf
	Doughnut 3D		
버블 차트	Bubble 3D		rMateChart.swf
	Multi Series Bubble 3D		
플롯 차트	Plot chart		
컴비네이션 차트	Line + Column 2D		rMateChart.swf
	Line + Column 3D		
	Line + Stacked Column 3D		
	Line + Multi Column 3D		

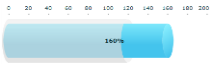
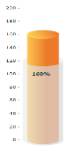
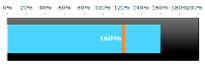
	Area + Line + Column 2D		
	Area + Line + Multi Column 2D		
Dashed-Line Chart	Dashed Line		rMateChart.swf
	Line + Dashed_Line		
실린더 3D 차트	Cylinder 3D Chart		rMateChart.swf
	Multi Series Cylinder 3D Chart		
	Stacked Cylinder 3D Chart		
	Cylinder Bar 3D Chart		
From-To Chart	Steps From-To Chart		rMateChart.swf
	Waterfall From-To Chart		
	Area From-To 2D Chart		

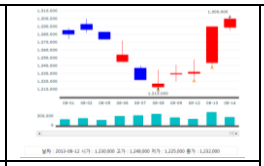
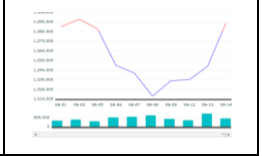
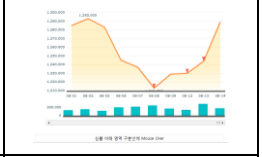
방사형(Radar) 차트	Radar Polygon Type		rMateRadarChart.swf
	Radar Circle Type		
목표 대비 실적 차트	3D Cylinder Type		rMateChart.swf
	Bar 3D Cylinder Type		
	2D Linear Type		
히스토리 차트	History Display Column Chart		rMateHistoryChart.swf
	History Display Column with Line Chart		
실시간 차트	데이터 개수 기준		rMateRealtimeChart.swf
	시간 기준		
스크롤 차트	Scroll Column		rMateScrollChart.swf
	Scroll Column Multi-Series		

	Scroll Bar Multi-Series		
	Scroll Line Multi-Series		
	Scroll Area Multi-Series		
	Scroll Combination		
브로큰 축 (Broken Axis) 차트	Column2DChart Broken Axis		rMateBrokenChart.swf
	Bar2DChart Broken Axis		
	Column3DChart Broken Axis		
	Bar3DChart Broken Axis		
	Area2DChart Broken Axis		
매트릭스 (Matrix) 차트	Basic Type		rMateMatrixChart.swf
	Matrix Fill Type		

	Matrix Plot Type		
	Matrix Image Type (외부이미지 삽입)		
이미지 차트	정배율-단일이미지		rMateImageChart.swf
	차등배율-단일이미지		
	정배율-반복이미지		
	차등배율-복수이미지		
슬라이드 차트	Slide + Effect Chart 다수의 차트를 슬라이드 형식으로 표현		rMateIntegration.swf
윙 차트	컬럼형태 Column 2D Wing		rMateWingChart.swf
	컬럼형태 Column 2D Wing Multi		
	컬럼형태 Column 2D Wing Stacked		
	컬럼형태 Column 2D Wing Stacked 연결선 잇기		

	바형태	Bar 2D Wing		
		Bar 2D Wing Multi		
		Bar 2D Wing Stacked		
		Bar 2D Wing Stacked 연결선 잇기		
실시간-프리미엄 차트		다른 주기 데이터 실시간 표현		rMateRealtimePremium.swf
		전일, 금일 실시간 비교		
		다른 주기 데이터 실시간 표현		
Circular 게이지 차트		Circular Dual		rMateGaugeChart.swf
		Circular Rainbow		
		Circular Orange		
		Circular Gradient		

Half-Circular 게이지 차트	Half-Circular Gradient		rMateGaugeChart.swf
	Half-Circular Rainbow2		
	Half-Circular Rainbow1		
	Half-Circular Notice		
Cylinder 게이지 차트	Horizontal Cylinder Gauge		rMateGaugeChart.swf
	Vertical Cylinder Gauge		
Linear 게이지 차트	Horizontal Linear Gauge		rMateGaugeChart.swf
	Vertical Linear Gauge		
Candlestick 캔들스틱 차트	Candlestick Chart		**rMateCandleChart.swf**
	Candlestick Chart Reverse		

Candlestick 캔들스틱 차트	Candlestick Symbol		**rMateCandleChart.swf**
	Candlestick Symbol Another		
	CandleLine Symbol		
	CandleLine Baseline		
	CandleArea Symbol		
	CandleArea Baseline		

<그림 2 매칭도>

2. 기본적인 차트의 생성.

2.1. rMate Chart 라이선스 등록

정상적인 차트를 생성하기 위해서는 rMate Chart 라이선스를 등록하셔야 합니다. 제공된 시디의 다음경로에 있는 파일이 차트에 삽입할 라이선스입니다.

- /LicenseKey/rMateChartLicense.js

위 자바스크립트 파일을 <head> 태그 안에 삽입하시면 됩니다. **다른 작업을 하실 필요는 없습니다.**

```
<html>
<head>

<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>
```

```
<!—rMate Chart 라이선스 등록 완료 모습 -->
<script src="rMateChartLicense.js" language="javascript"></script>


<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">
                                                    ...
                                                    ...
                                                    ...
```

<예제 1 rMate Chart 라이선스 등록 예제>

2.2. 제공된 샘플을 토대로 차트 생성하기(Quick Learning).

rMate 차트를 Local 환경에서 테스트하기 위해서는 다음 작업이 선행되어야 합니다. 이는 Adobe Flash Player 보안정책에 해당됩니다.

Step1. 아래 해당되는 경로를 따라 폴더로 이동하십시오.

- Windows XP: `Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust`
  (for example, `C:\Documents and Settings\User1\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust`)

- Vista :
  `appData\Roaming\Macromedia\FlashPlayer\#Security\ FlashPlayerTrust`
  (for example, `C:\Users\User1\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust`

- Windows7:

```
C:\Users\User1\AppData\Roaming\Macromedia\
                        Flash Player\#Security\FlashPlayerTrust
```

*** Application Data 폴더는 숨겨진 폴더입니다. 탐색기 메뉴 중 도구-> 폴더옵션-> 보기-> 숨긴 파일 및 폴더 표시를 체크하십시오.**

Step2. FlashPlayerTrust 폴더 안에 **rMateTrust.cfg** 라는 텍스트 형식의 파일을 만들고 메모장을 이용하여 편집하기 위해 여십시오.

Step3. 제공된 rMate 차트 저장 폴더 경로를 rMateTrust.cfg 에 넣어주십시오.

예를 들어 제공된 샘플의 저장 위치가 C:₩Users₩User1₩Desktop₩rMateChart 입니다.

위 경로를 그대로 rMateTrust.cfg 파일에 추가하세요.
최종적인 결과는 다음과 같습니다.

```
C:₩Users₩User1₩Desktop₩rMateChart
```

<예제 2 rMateTrust.cfg 파일의 내용>

위 과정은 현재 등록된 폴더 안의 플래시 파일들은 신뢰할 수 있다는 의미입니다.

이제 3D 칼럼 싱글 데이터 차트를 생성해 보도록 하겠습니다.
제공된 제품에서 아래의 경로를 따라 각각의 파일을 복사하여 작업폴더 안의 같은 폴더 안에 복사하십시오.

- 데이터 파일 Samples/dataXml/singleData.xml
- 레이아웃 파일 : Samples/LayoutXml/Column_3D_Layout.xml
- 차트 swf 파일 : rMateChart /omponent/rMateChart.swf
- 자바스크립트 파일 rMateChart /JS/AC_OETags.js
  rMateChart /JS/rMateChart.js
- 라이선스 파일 : LicenseKey / rMateChartLicense.js

* **자바스크립트 파일은 플래시 차트를 생성시키기 위해 반드시 필요하니 반드시 포함시켜야 합니다. 그리고 사용자가 스크립트 파일을 수정 하실 필요는 없습니다.**

* 정상적인 차트 생성을 위해 라이선스 파일은 반드시 포함시켜야 합니다.

위 파일을 아래와 같이 HTML 파일에 삽입합니다.

```
<html>
<head>

<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>
<script src="rMateChartLicense.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
```

```
<script language="JavaScript" type="text/javascript">

// -------------------- flashVars 설정 시작 ----------------------

// 차트 레이아웃 URL 경로를 설정하십시오.
var layoutURL = encodeURIComponent("Column_3D_Layout.xml");
var flashVars = "layoutURL="+layoutURL;

// 차트 데이타 URL 경로를 설정하십시오.
var dataURL =encodeURIComponent("singleData.xml");
flashVars += "&dataURL="+dataURL;

</script>
<!-- 사용자 정의 설정 끝 -->
</head>

<body onload="rMateChartInit()"> //동기화를 위한 함수입니다.
<table>
	<tr>
		<td>
			<script>
			<!—
			 // 차트를 생성하고자 하는 위치에서 다음과 같이 함수 호출하십시오.
			 // 파라메터 설명(순서대로)
			 // 1. ID (임의로 정의하십시오)
			 // 2. swf 파일 URL(확장자는 생략)
			 // 3. URL 경로 등을 정의한 변수명.
			 // 4. 차트 가로 사이즈.
			 // 5. 차트 세로 사이즈.
			 // 6. 차트 배경색.

				rMateChartCreate("chart1","rMateChart",flashVars, 500, 500, "#FFFFFF");
			// -->
			</script>
		</td>
	</tr>
</table>
</body>
</html>
```

<예제 3 Html 문서에 차트 삽입하기>

웹 브라우저를 통하여 생성된 HTML 파일을 로딩하면 3D 칼럼 차트를 볼 수 있을 것입니다. rMate 차트의 기본 원리와 커스텀 데이터 삽입에 대한 자세한 설명은 2.3 장에서 계속됩니다.

***참고** : rMate 차트와 스크립트간의 동기화 확인을 위한 함수인 rMateChartInit() 는 반드시 처음 스크립트가 로드될 때 호출되어야 합니다.

제공된 모든 샘플은 html 문서 안의 body 태그에 onload 함수를 이용한
<body onload="rMateChartInit()"> 로 호출됩니다. onload 시 호출해야 할 함수가 더 있을 경우 아래를 참고하세요.

```
<html>
   <head>          //헤더 부분에 아래와 같이 window.onload 삽입하시고
      ...          // onload 할 때 호출해야 할 함수를 넣으십시오.
      ...
       <script type="text/javascript">
                window.onload = function ()
                {
                    rMateChartInit();
                    ......................
                }
       </script>
    </head>
...
<html>
```

<예제 4 rMateChartInit() 동기화 예제>

2.3. 사용자 정의 데이터를 바탕으로 차트 생성하기(Step by Step).

매월 이윤에 관한 데이터를 바탕으로 rMate 3D 칼럼 차트를 생성해 보겠습니다.
그러기 위해 다음 단계를 순차적으로 밟으십시오.

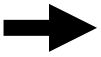
1. **swf 파일을** <그림 2 매칭도>**를 참고하여 선택하십시오.** 원하는 차트가 3D 칼럼 차트이니 **rMateChart.swf** 입니다.

2. **데이터를 작성합니다.** 매월 이윤에 관한 데이터는 아래와 같이 XML 형식으로 변환이 가능합니다.

3.

<일반적인 데이터>

월(Month)	이윤(Profit)
Jan.	10,000
Feb.	15,000
Mar.	12,000
Apr.	30,200
May.	28,000
Jun.	12,000
Jul.	22,000
Aug.	13,000
Sep	22,000
Oct.	29,000
Nov.	18,000
Dec.	30,000



<XML 형식의 데이터>

```
<items>
        <item>
                <Month>Jan</Month>
                <Profit>10000</Profit>
        </item>
        <item>
                <Month>Feb</Month>
                <Profit>15000</Profit>
        </item>
                .
                .
                .
        <item>
                <Month>Dec</Month>
                <Profit>30000</Profit>
        </item>
</items>
```

XML 형식으로 데이터를 변환할 때 데이터 한 개의 시작과 끝은 반드시 item 이여야 합니다.

<예제 5 데이터 변환>

데이터 표현은 위와 같이 변환이 됩니다. 다음 경로에서 샘플로 제공된 데이터 파일을 확인하실 수 있습니다(Samples/DataXml/singleData.xml)

3. **레이아웃 파일을 선택합니다.** 레이아웃의 선택은 데이터 표현 형식과 차트 유형에 따라 달라집니다. 3D 칼럼 차트이고 수치 데이터는 이윤(Profit) 한 개이므로 싱글데이타 3D 칼럼에 해당되는 레이아웃을 위의 <그림 2 매칭도> 를 참고하여 찾습니다. 찾은 결과 파일명은 **Column_3D_Layout.xml** 입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart borderStyle="solid" borderThickness="2" paddingLeft="5" paddingRight="5" cornerRadius="15">
        <Options>
```

```
            <Caption text="Annual Report"/>
            <SubCaption text="2008"/>
        </Options>
        <Column3DChart showDataTips="true" width="100%" height="100%" >
            <horizontalAxis>
                <CategoryAxis categoryField="Month"/>
            </horizontalAxis>
            <series>
                <Column3DSeries yField="Profit" displayName="Profit">
                    <showDataEffect>
                        <SeriesInterpolate/>
                    </showDataEffect>
                </Column3DSeries>
            </series>
        </Column3DChart>
</rMateChart>
```

<예제 6 Column_3D_Layout.xml 의 내용>

만약 XML 형식으로 데이터를 작성 할 때 <item> 하위 노드를 <Month>와 <Profit>으로 작성하지 않고 예를 들어 <month>와 <profit>으로 작성하였다면 차트는 제대로 데이터를 표현하지 않습니다. **데이터와 레이아웃 필드 명을 반드시 일치시켜야 합니다.**
레이아웃에 관련된 자세한 설명은 "5.차트와 레이아웃."을 참고하여 주십시오.

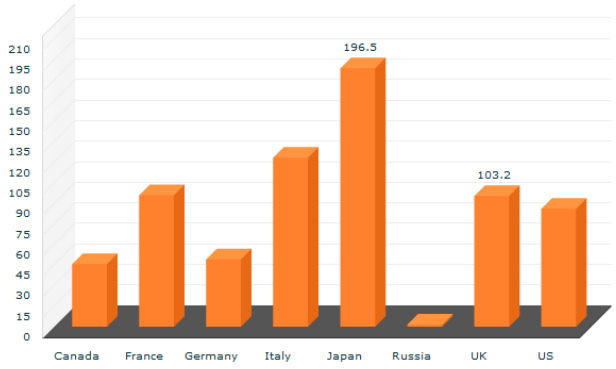
4. **차트를 웹 상에 표현할 HTML 파일을 만드십시오.** 차트 swf 파일과 데이터 xml 파일, 레이아웃 xml 파일을 아래와 같이 삽입합니다.

```
<html>
<head>
<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>
<script src="rMateChartLicense.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ---------- flashVars 설정 시작 ----------

// 차트 레이아웃 URL 경로를 설정하십시오.
var layoutURL = encodeURIComponent("Column_3D_Layout.xml");
var flashVars = "layoutURL="+layoutURL;

// 차트 데이타 URL 경로를 설정하십시오.
var dataURL =encodeURIComponent("singleData.xml");
```

```
flashVars += "&dataURL="+dataURL;

<!-- 사용자 정의 설정 끝  -->
</script>
</head>

<body onload="rMateChartInit()"> //동기화를 위한 함수입니다.
<table>
	<tr>
		<td>
			<script>
			<!—
			 // 차트를 생성하고자 하는 위치에서 다음과 같이 함수 호출하십시오.
			 // 파라메터 설명(순서대로)
			 // 1. ID (임의로 정의하십시오)
			 // 2. swf 파일 URL(확장자는 생략)
			 // 3. URL 경로 등을 정의한 변수명.
			 // 4. 차트 가로 사이즈.
			 // 5. 차트 세로 사이즈.
			 // 6. 차트 배경색.

				rMateChartCreate("chart1","rMateChart",flashVars, 500, 500, "#FFFFFF");
			// -->
			</script>
		</td>
	</tr>
</table>
</body>
</html>
```

<예제 7 Html 문서에 차트 삽입하기>

위와 같이 입력 후 해당 html 파일을 웹 브라우저를 통해 로딩합니다.
다음과 같은 차트 화면을 볼 수 있을 것입니다.



<그림 3 매월 이윤을 표현한 rMate Column 3D Chart>

만약 위와 같이 차트가 생성된 화면을 볼 수 없다면 위의 HTML 코드를 다시 한번 확인하여 주십시오.

2.4. 다른 유형의 차트, 파이차트 생성하기.

이전의 예제 "2.3 사용자 정의 데이터를 바탕으로 차트 생성하기(Step by Step)."에서 매월 이윤에 대하여 3D 칼럼 차트를 생성하였습니다. 이를 2D 파이 차트로 표현해 보도록 하겠습니다.
<그림 2 매칭도>를 바탕으로 2D 파이 차트 swf 와 레이아웃 파일을 선택하십시오. 매월 이윤의 수치 데이터는 Profit 한 개 이므로 데이터 표현(즉, 차트 시리즈)은 한 개 입니다. 그러므로 선택한 파일은 각각 rMateChart.swf 와 Pie_2D_Layout.xml 입니다.
이제 3D 칼럼 차트를 2D 파이 차트로 변환해 보도록 하겠습니다. 위에 <그림 2 매칭도>를 바탕으로 선택한 파일을 이전의 작업 폴더에 그대로 복사하여 넣습니다.
그리고 이전의 HTML 파일을 열어 편집합니다.
편집해야 할 부분은 레이아웃 경로입니다.

레이아웃을 바꿔 줍니다.

Column_3D_Layout.xml ----→ Pie_2D_Layout.xml

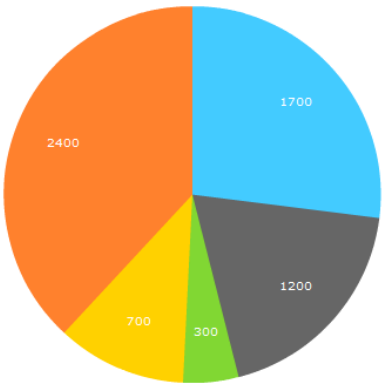
아래 수정된 HTML 문서 예제를 참고하여 주십시오.

```
<html>
<head>
<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>
<script src="rMateChartLicense.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ---------- flashVars 설정 시작 ----------

// 차트 레이아웃 URL 경로를 설정하십시오.
var layoutURL = encodeURIComponent("Pie_2D_Layout.xml");
var flashVars = "layoutURL="+layoutURL;

// 차트 데이타 URL 경로를 설정하십시오.
var dataURL =encodeURIComponent("singleData.xml");
flashVars += "&dataURL="+dataURL;
```

```
<!-- 사용자 정의 설정 끝  -->
</script>
</head>

<body onload="rMateChartInit()"> //동기화를 위한 함수입니다.
<table>
	<tr>
		<td>
			<script>
			<!—
			 // 차트를 생성하고자 하는 위치에서 다음과 같이 함수 호출하십시오.
			 // 파라메터 설명(순서대로)
			 // 1. ID (임의로 정의하십시오)
			 // 2. swf 파일 URL(확장자는 생략)
			 // 3. URL 경로 등을 정의한 변수명.
			 // 4. 차트 가로 사이즈.
			 // 5. 차트 세로 사이즈.
			 // 6. 차트 배경색.

				rMateChartCreate("chart1","rMateChart",flashVars, 500, 500, "#FFFFFF");
			// -->
			</script>
		</td>
	</tr>
</table>
</body>
</html>
```

<예제 8 매월 이윤을 파이 2D 차트로 표현하기 위한 HTML 문서>

아래와 같은 파이 차트를 볼 수 있을 것입니다.



<그림 4 매월 이윤을 표현한 rMate 파이 2D 차트>

3. 차트 데이터 형식

rMate Chart 의 수치 데이터는 **XML 형식과 배열 형식**을 지원합니다. 사용자는 몇 가지 규칙에 따라 차트 데이터를 작성할 필요가 있습니다. 규칙에 의거하여 단일 데이터(차트의 싱글 시리즈)와 다중 데이터(차트의 멀티 시리즈)를 XML 형식과 배열형식으로 각각 변환하여 어떻게 차트에 삽입하는지를 이번 장에서 설명합니다.

3.1. XML 형식의 데이터 작성 및 차트에 삽입하기.

XML 형식으로 데이터를 작성 할 때 반드시 준수해야 할 점은 한 개의 데이터를 감싸는 노드는 반드시 <item>으로 시작하여 </item>으로 끝을 맺어야 합니다. rMate Chart 는 <item>의 개수만큼 데이터를 출력하게 됩니다. 데이터 XML 파일에서 <item></item>이 존재 하지 않는다면 차트는 데이터를 표현하지 않습니다.

단일 데이터(즉, 차트의 싱글 시리즈)는 표현하고자 하는 수치가 한 개일 때 유효합니다. 2.3 장에서 작성한 매월 이윤에 관한 데이터가 단일 데이터의 좋은 예라고 볼 수 있습니다.
단일 데이터 차트의 데이터 작성의 예는 "2.3 사용자 정의 데이터를 바탕으로 차트 생성하기(Step by Step). 을 참고하여 주십시오.

다중 데이터(즉, 차트의 멀티 시리즈)는 표현하고자 하는 수치가 2 개 이상일 때 유효합니다. 매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 월간 보고서를 차트로 표현하기 위해 데이터를 작성해 보도록 하겠습니다.

월(Month)	매출(Revenue)	비용(Cost)	이윤(Profit)
Jan.	10,000	5,000	5,000
Feb.	15,000	7,000	8,000
Mar.	12,000	6,000	6,000
Apr.	30,200	4,000	26,200
May.	28,000	10,000	18,000
Jun.	12,000	5,000	7,000
Jul.	22,000	10000	12,000
Aug.	13,000	6,000	7,000
Sep	22,000	10,000	12,000
Oct.	29,000	8,000	11,000
Nov.	18,000	7,500	10,500
Dec.	30,000	12,000	28,000

<XML 형식의 데이터>

```
<items>
        <item>
            <Month>Jan</Month>
            <Revenue>10000</ Revenue >
            <Cost>5000</Cost>
            <Profit>5000</Profit>
        </item>
        <item>
            <Month>Feb</Month>
            <Revenue>15000</Revenue>
            <Cost>7000</Cost>
            <Profit>8000</Profit>
        </item>
                    .
                    .
                    .
        <item>
            <Month>Dec</Month>
            <Revenue>30000</Revenue>
            <Cost>12000</Cost>
            <Profit>18000</Profit>
        </item>
</items>
```

XML 형식으로 데이터를 변환할 때 데이터 한 개의 시작과 끝은 반드시 item 이여야 합니다.

<예제 9 XML 형식으로 데이터 변환>

위와 같이 다중 데이터(멀티 시리즈)를 XML 유형으로 표현할 수 있습니다.

예를 들어 위에 작성한 XML 데이터를 "singleData.xml" 로 저장하였다면 차트에 singleData.xml 의 경로를 넘겨줘야 합니다. 이에 대한 예제는 <예제 3 Html 문서에 차트 삽입하기> 를 참고하세요.

3.2. 배열 형식의 데이터 작성 및 차트에 삽입하기.

rMate Chart 는 사용자의 편의를 위하여 배열 형식의 데이터를 차트에 삽입할 수 있도록 하였습니다. 그 방법에 대한 설명입니다.
매월 이윤에 대해 자바스크립트에서 배열 형식으로 데이터를 작성해 보도록 하겠습니다.

월(Month)	이윤(Profit)
Jan.	10,000
Feb.	15,000
Mar.	12,000
Apr.	30,200
May.	28,000
Jun.	12,000
Jul.	22,000
Aug.	13,000
Sep	22,000
Oct.	29,000
Nov.	18,000
Dec.	30,000

```
var chartData = [{"Month":"Jan", "Profit":10000},
        {"Month":"Feb", "Profit":15000},
        {"Month":"Mar", "Profit":12000},
        {"Month":"Apr", "Profit":30200},
        {"Month":"May", "Profit":28000},
        {"Month":"Jun", "Profit":12000},
        {"Month":"Jul", "Profit":22000},
        {"Month":"Aug", "Profit":13000},
        {"Month":"Sep", "Profit":22000},
        {"Month":"Oct", "Profit":29000},
        {"Month":"Nov", "Profit":18000},
        {"Month":"Dec", "Profit":30000} ];
```

<예제 10 배열형식으로 데이터 변환>

위와 같이 배열 형식으로 변환할 수 있습니다.

3.1 장에서 작성된 매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 작성된 다중 데이터를 배열형식으로 변환하면 다음과 같습니다.

```
var chartData = [{"Month":"Jan", "Revenue":10000, "Cost":5000, "Profit":5000},
        {"Month":"Feb", "Revenue":15000, "Cost":7000, "Profit":8000},
        {"Month":"Mar", "Revenue":12000 , "Cost":6000, "Profit":6000},
        {"Month":"Apr", "Revenue":30200, "Cost":4000, "Profit":26200},
        {"Month":"May", "Revenue":28000, "Cost":10000, "Profit":18000},
        {"Month":"Jun", "Revenue":12000, "Cost":5000, "Profit":7000},
        {"Month":"Jul", "Revenue":22000, "Cost":10000, "Profit":12000},
        {"Month":"Aug", "Revenue":13000, "Cost":6000, "Profit":7000},
        {"Month":"Sep", "Revenue":22000, "Cost":10000, "Profit":12000},
        {"Month":"Oct", "Revenue":29000, "Cost":8000, "Profit":21000},
        {"Month":"Nov", "Revenue":18000, "Cost":7500, "Profit":10500},
```

```
        {"Month":"Dec", "Revenue":30000, "Cost":12000, "Profit":18000} ];
```

<예제 11 배열형식으로 다중 데이터 변환>

배열형식의 데이터를 차트에 삽입하는 방법은 아래와 같습니다.
아래 예제는 싱글 데이터를 삽입하는 예제입니다.

```
<html>
<head>
<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>
<script src="rMateChartLicense.js" language="javascript"></script>


<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ---------------------- flashVars 설정 시작----------------------

// 차트 레이아웃 URL 경로. 반드시 설정하십시오.
var layoutURL = encodeURIComponent("Column_3D_Layout.xml");
var flashVars = "layoutURL="+layoutURL;


// 배열형태로 데이터를 삽입할 경우 XML URL 입력부분은 주석처리나 삭제하십시오.  1


// rMate Chart와 스크립트간 동기화가 완료된 후 Chart가 함수를 호출합니다.
// 이 함수를 통하여 차트와 통신하기 때문에 배열형태로 데이터를 삽입할 경우 반드시
// 정의하여야 합니다.
var rMateOnLoadCallFunction = "rMateChartOnLoad";  2
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;


// 스크립트와 차트간의 동기화가 완료된 후 호출할 함수로 사용자가 정의한 함수입니다.
// setData 함수를 통해 차트에 데이터를 전달하십시오.
// setData 함수는 rMate 차트가 배열을 통해 데이터를 받기 위해 열어 놓은 함수입니다.
// 이 함수의 파라메터는 곧 차트의 데이터입니다.

function rMateChartOnLoad()  3
{
        chart1.setData(chartData);  5
```

```
}

var chartData = [{"Month":"Jan", "Profit":10000},
                 {"Month":"Feb", "Profit":15000},
                 {"Month":"Mar", "Profit":12000},
                 {"Month":"Apr", "Profit":30200},
                 {"Month":"May", "Profit":28000},
                 {"Month":"Jun", "Profit":12000},
                 {"Month":"Jul", "Profit":22000},
                 {"Month":"Aug", "Profit":13000},
                 {"Month":"Sep", "Profit":22000},
                 {"Month":"Oct", "Profit":29000},
                 {"Month":"Nov", "Profit":18000},
                 {"Month":"Dec", "Profit":30000} ];

<!-- 사용자 정의 설정 끝  -->
</script>
</head>

<body onload="rMateChartInit()">
<table>
        <tr>
        <td>
                <script>
                <!--
                  // 차트를 생성하고자 하는 위치에서 다음과 같이 함수 호출하십시오.
                 // 파라메터 설명(순서대로)
                 // 1. ID (임의로 정의하십시오)
                 // 2. swf 파일 URL(확장자는 생략)
                 // 3. URL 경로 등을 정의한 변수명.
                 // 4. 차트 가로 사이즈.
                 // 5. 차트 세로 사이즈.
                 // 6. 차트 배경색.

                        rMateChartCreate("chart1","rMateChart",flashVars, 500, 500, "#FFFFFF");
                // -->
                </script>
        </td>
</tr>
</table>

</body>
</html>
```

4

<예제 12 배열형식으로 데이터 삽입하는 HTML 문서>

위의 차트를 삽입한 html 문서를 보면 배열로 데이터를 삽입하기 위해 다음 과정을 밟아야 합니다.

첫째로, XML 경로를 통한 데이터 삽입부분을 주석처리 하거나 삭제하십시오.

둘째, rMate Chart 와 스크립트간 동기화가 완료된 후 rMate Chart 가 호출할 함수를 지시하십시오. 이는 flashVars 라는 플래시 변수를 통하여 지시합니다. 지시법은 아래와 같습니다.

```
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;
             변경 불가                   사용자에 의해 정의
```

rMateOnLoadCallFunction 은 차트가 인식할 문자열 상수입니다. 그렇기 때문에 그대로 유지하시고 사용자가 정의할 함수 명을 그 값으로 입력하고 함수를 정의하십시오. 그러면 차트는 동기화가 완료된 후 사용자가 정의한 함수(rMateOnLoadCallFunction 으로 설정한 함수)를 호출합니다.

셋째, rMateOnLoadCallFunction 에 할당된 함수 명에 맞는 함수를 직접 정의하십시오.
위 예에서 함수명은 "rMateChartOnLoad" 이므로 `function rMateChartOnLoad()`를 정의하였습니다.

넷째, 차트 데이터를 배열 형태로 작성해 주십시오. 작성방법은 "3.2. 배열 형식의 데이터 작성 및 차트에 삽입하기." 를 참고하십시오.

다섯째, `function rMateChartOnLoad()` 함수를 통해 차트에 데이터를 배열로 삽입하십시오.
위 예에서 `chart1.setData(chartData);` 부분입니다. 차트에 데이터를 배열형태로 삽입하기 위해서 차트는 setData 라는 함수를 열어 놓았습니다. 이 함수는 반드시 배열형태 객체를 파라메터로 갖습니다.

다음 장에서 setData 와 같이 차트에 접근할 수 있는 함수들에 대하여 설명합니다.

4. 차트 설정 및 연동방식

4.1. flashVars 설정

flashVars 변수는 rMate Chart 의 데이터, 레이아웃 등을 설정할 수 있는 가장 기본적인 변수입니다. 이 변수는 스크립트 단에서 차트를 임베딩할 때 함께 파라메터로 함께 삽입하게 됩니다.

flashVars 변수 설정은 다음과 같은 규칙을 따릅니다.

- flashVars 변수는 스트링입니다.
- 두 개 이상의 설정을 하기 위해 구분자 & 를 사용합니다. 예를 들어 데이터 XML URL 과 레이아웃 XML URL 을 설정하는 방법은 아래와 같습니다

```
var layoutURL = encodeURIComponent("./Column_3D_Layout.xml");
var flashVars = "layoutURL="+layoutURL;

var dataURL =encodeURIComponent("./singleData.xml");
flashVars += "&dataURL="+dataURL;
```

flashVars 로 설정 가능한 속성값은 다음과 같습니다.

속성명	유효값	설명
layoutURL	URL 주소	레이아웃 URL 경로
dataURL	URL 주소	데이터 URL 경로
rMateOnLoadCallFunction	자바스크립트 함수명	스크립트와 차트 동기화 시 차트가 호출할 함수 설정(최초 1번만 호출됨)
displayCompleteCallFunction	자바스크립트 함수명	차트가 데이터까지 모두 출력 완료 후 즉, 모든 작업 완료 후 호출할 함수 설정.(최초 1번만 호출됨)
chartImageDpi	Number(기본값:72)	차트 이미지 저장 및 스냅샷 찍을 때 적용될 Png 이미지 DPI 설정
debugMode	Boolean(기본값:false)	디버그 모드 사용 여부. true로 설정 시 마우스 오른쪽 버튼 메뉴에 Chart Log 메뉴가 활성화됩니다.
useSideBar	Boolean(기본값:false)	사이드 메뉴 사용 여부, true 로 설정 시 차트는 ChartSideBar.swf 를 로딩하여 차트에 적용시킵니다.
useDataEditor	Boolean(기본값:false)	데이터 에디터 사용 여부, true 로 설정 시 차트는 ChartDataEditor.swf 를 로딩하여 차

		트에 적용시킵니다.
saveImage	Boolean(기본값:true)	마우스 오른쪽 클릭 시 나타나는 컨텍스트 메뉴에 '차트 이미지 저장' 추가 여부
addCrossHairToSide	Boolean(기본값:false)	사이드바 메뉴에 십자가 표시 메뉴 아이템 설정 여부
addEditorToSide	Boolean(기본값:false)	사이드바 메뉴에 데이터 에디터 메뉴 아이템 설정 여부
addZoomToSide	Boolean(기본값:false)	사이드바 메뉴에 확대/축소 메뉴 아이템 설정 여부
addDrawingToSide	Boolean(기본값:true)	사이드바 메뉴에 그리기 도구 메뉴 아이템 설정 여부
addSaveAsImgToSide	Boolean(기본값:true)	사이드바 메뉴에 이미지로 저장 메뉴 아이템 설정 여부
addPrintToSide	Boolean(기본값:true)	사이드바 메뉴에 프린트 차트 메뉴 아이템 설정 여부
fontURL	URL 주소	외부 폰트 URL 경로
flash.system.System.useCodePage	Boolean(기본값:false)	EUC-KR 로 인코딩 된 레이아웃의 한글지원 (디폴트로 UTF-8임)
preloaderColor	Uint(RGB 컬러)	추가적인 프리로더의 색상을 변경합니다.
javascriptTryCount	Int(기본값:20)	동기화가 될때까지 20초동안 동기화함수를 실행합니다. 20초가 지나면 동기화 에러가 발생합니다.
accessibility	Boolean(기본값:true)	시각 장애인을 위한 대체 텍스트를 사용할지 설정 여부
usePattern	Boolean(기본값:false)	색맹, 색약을 가지고 있는 분들을 위한 패턴 모듈인 Pattern.swf를 로딩하여 차트에 적용합니다.

- rMateOnLoadCallFunction 과 displayCompleteCallFunction 의 차이점.

rMateOnLoadCallFunction 은 스크립트 파일(html, jsp, php 등) 과 차트 간의 동기화가 완료된 후 호출되는 함수 입니다. 이 함수는 알메이트 차트 준비가 완료되었기에 데이터, 레이아웃을 삽입할 때 많이 쓰입니다.

그러나 displayCompleteCallFunction 은 차트에 주어진 데이터와 레이아웃을 바탕으로 차트 출력이 완전히 완료되었을 경우 호출되는 함수입니다. 따라서 이 함수가 호출되었을 경우에는 화면에 차트 출력이 끝끝난 것을 볼 수 있습니다. 차트 출력이 마무리 되고 어떤 작업을 해야 할 경우 유용하게 쓸 수 있는 함수입니다.

- flashVars 사용 예.

```
var rMateOnLoadCallFunction = "rMateChartOnLoad";
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;

function rMateChartOnLoad()
{
    ...
    ...
}
```

위와 같이 flashVars 를 설정한 후 실제적으로 차트를 생성하는 함수에 설정한 flashVars 를 파라메터로 넣어줍니다.

```
rMateChartCreate("chart1","rMateChart",flashVars, 500, 500, "#FFFFFF");
```

4.2. 차트에 접근 가능한 함수들

사용자의 편의를 위해 rMate Chart 는 다음과 같은 함수를 열어 놓았습니다.

1. setDataURL(value) : 데이터 XML URL 경로를 차트에 삽입합니다.
2. setLayoutURL (value): 레이아웃 XML URL 경로를 차트에 삽입합니다.
3. setData (value): 배열 형태 또는 XML 스트링 형태의 데이터를 차트에 삽입합니다.
4. setLayout (value): 스트링 형태의 레이아웃을 차트에 삽입합니다.
5. printChart (): 이 함수 호출 시 차트 인쇄를 시작합니다.
6. showDataEditor(): 데이터 에디터를 하단에 표시합니다.
7. removeDataEditor():표시된 데이터 에디터를 제거합니다.
8. saveAsImage (): 차트 이미지를 PC 에 저장할 수 있는 창을 띄웁니다.(이미지로 저장과 같은 기능)
9. getSnapshot(): 차트 이미지를 base64 로 인코딩된 형태의 스냅샷을 얻어냅니다.
10. setSlideDataSet (value): 슬라이더 기능에서 사용할 데이터 셋을 설정합니다.
11. setSlideLayoutSet (value): 슬라이더 기능에서 사용할 레이아웃 셋을 설정합니다.
12. legendAllCheck(value): 레이아웃에서 Legend 를 useVisibleCheck='true'로 설정 했을 경우 범례의 전체선택(true)/전체해제(false) 를 제어 할 수 있습니다.
13. showAdditionalPreloader(): 프리로더를 강제로 표시합니다.
14. removeAdditionalPreloader(): 표시된 프리로더를 제거합니다.
15. getChartData(): 차트의 데이터를 반환합니다.
16. visibleItemSize(): 스크롤 차트 출력 후 화면에 보여지는 아이템 개수를 제어할 수 있습니다.
17. defaultColors(value): 배열 형태의 기본색을 설정합니다.

위 함수들을 활용하여 rMate Chart 는 총 6 가지 방법으로 데이터와 레이아웃을 삽입할 수 있습니다. 이것을 정리한 표는 아래와 같습니다.

	레이아웃	데이터	사용 함수
첫 번째 방법	XML 경로	XML 경로	setLayoutURL(경로), setDataURL(경로)
두 번째 방법	XML 경로	배열 형태	setLayoutURL(경로), setData(배열)
세 번째 방법	XML 경로	XML 스트링	setLayoutURL(경로), setData(스트링)
네 번째 방법	스트링 형태	XML 경로	setLayout(스트링), setDataURL(경로)
다섯 번째 방법	스트링 형태	배열 형태	setLayout(스트링), setData(배열)
여섯 번째 방법	스트링 형태	XML 스트링	setLayout(스트링), setData(스트링)

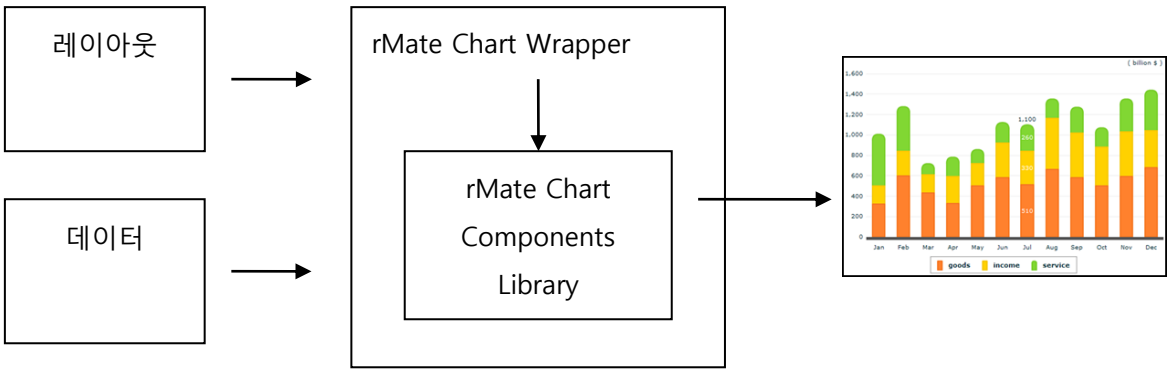
<표 2 차트에 레이아웃과 데이터를 삽입하는 방법을 나타낸 표>

위 함수를 이용하여 레이아웃을 동적으로 바꾸거나 변경된 데이터를 동적으로 삽입할 수 있습니다. 이에 대한 예제는 "8.19.동적으로 차트 레이아웃 또는 데이터 변경하기."를 참고하여 주십시오.

5. 차트와 레이아웃.

5.1. rMate Chart 에서 레이아웃의 역할.

rMate Chart 는 미리 작성된 레이아웃의 XML 을 파싱하여 동적으로 생성됩니다. 사용자가 작성한 레이아웃을 rMate Chart 의 특정 swf 파일(예를 들면 rMateChart.swf)에 입력하면 차트는 미리 내포하고 있는 rMate Chart Components 클래스를 기반으로 레이아웃에 작성된 차트를 생성하고 차트의 수치 데이터를 출력하게 됩니다.
다음은 이런 과정을 다이어그램으로 표현한 것입니다.



<그림 5 rMate Chart 동작 다이어그램>

사용자는 레이아웃을 작성하고 차트를 생성할 때 반드시 레이아웃에서 작성된 차트와 같은 차트를 표현하는 swf 파일을 선택하여야 합니다. 제공된 각각의 swf 파일은 rMate Chart Components 중 특정 차트만을 표현하는 Components 를 내포하고 있기 때문입니다.

5.2. 레이아웃 작성 방법에 대하여.

rMate 차트의 모든 레이아웃은 <rMateChart>태그로 시작하여 </rMateChart>태그로 끝납니다. 레이아웃은 크게 Options 와 차트(칼럼, 바, 파이 등의 차트), Style 세 부분으로 구분됩니다. 사용자는 Options 태그에서 차트의 제목(Caption)과 부제목(SubCaption) 그리고 범례(Legend)를 삽입할 수 있습니다.

레이아웃을 작성하는 방법은 기본적으로 XML 형식입니다.

rMate 차트는 ID 에 의한 객체 접근을 지원합니다. 예를 들어 ColumnSeries 객체의 속성으로 fill 이 존재합니다. 이 fill 속성의 속성값으로 SolidColor 인스턴스를 할당합니다. 이를 표현한 일반적인 방법은 아래와 같습니다.

```
<rMateChart>
    <Column3DChart showDataTips="true">
            .
            .
            .
        <series>
            <Column3DSeries yField="Profit">
                <fill>
                    <SolidColor color="0xFF0000">
                </fill>
            </Column3DSeries>
        </series>
    </Column3DChart>
</rMateChart>
```

<예제 13 일반적인 속성 값 할당 법>

그러나 사용자의 편의를 위해 <예제 14 ID 에 의한 속성 값 할당 법>과 같이 SolidColor 인스턴스를 미리 생성한 후 그 ID 를 통해 SolidColor 인스턴스를 활용할 수 있습니다. 이는 여러 객체들이 하나의 SolidColor 인스턴스에 접근하고자 할 때 불필요하게 SolidColor 인스턴스를 생성 하지 않고 미리 만들어진 하나의 SolidColor 에 접근하기 때문에 효율적입니다. 또한 어느 특정 객체를 반드시 참조해야 하는 경우에도 효율적입니다.

```
<rMateChart>
    <SolidColor id="color1" color="0xFF0000"/>
    <Column3DChart showDataTips="true">
            .
            .
            .
        <series>                                    //객체 ID인 경우엔 시작과 끝을 중괄호({,})로 묶음
            <Column3DSeries yField="Profit" fill="{color1}"/>
            <Column3DSeries yField="Cost" fill="{color1}"/>
        </series>
    </Column3DChart>
</rMateChart>
```

<예제 14 ID 에 의한 속성 값 할당 법>

인스턴스 ID 에 의한 접근을 활용한 좋은 예는 수직 축 2 개를 생성하여 서로 다른 시리즈가 각각의 축을 참조하는 것입니다. 8.4.4 장 <예제 71 수직축 2 개 설정한 예> 를 참고하여 주십시오.

앞으로 레이아웃을 작성할 때(차트 종류에 관계없이) Chart 와 Series 가 자주 나오게 될 것입니다. 이에 대한 개념을 정리하면 아래와 같습니다.

	설 명	비고
차트(Chart)	실질적인 차트입니다. 차트는 외형을 담당합니다. 축, 배경, 차트사이즈 등등이 포함됩니다. 가장 중요한 역할로 시리즈의 위치(좌표)를 지정합니다.	ColumnChart, Column3DChart, PieChart, Pie3DChart, BarChart 등등
시리즈(Series)	실질적인 데이터를 표현합니다. 표현하고자 하는 수치데이터가 3개인 경우 반드시 시리즈를 3개 정의할 필요가 있습니다.	ColumnSeries, Column3DSeries, BarSeries, Pie3DSereis 등등

<표 3 차트와 시리즈 설명표>

5.3. 레이아웃의 rMateChart 노드 설정하기.

rMateChart 노드는 레이아웃의 시작과 끝을 나타냅니다. 또한 이 노드의 속성은 차트 전체의 꾸미기를 담당합니다.

속성 명	유효 값	설명
backgroundColor	RGB(기본값: 0xFFFFFF)	바탕 색을 나타냅니다.
backgroundAlpha	0.0 ~ 1.0(기본값:1.0)	바탕 투명도를 나타냅니다.
borderStyle	none | solid | inset | outset (기본값:none)	테두리 선을 정의합니다.
borderColor	RGB(기본값: 0xB7BABC)	테두리 색을 정의합니다.
borderThickness	Number (기본값:1)	테두리 두께를 정의합니다.
cornerRadius	Number(기본값:0)	테두리의 둥글게 처리 정도를 나타냅니다.
paddingTop	Number(기본값:0)	위쪽 여백
paddingBottom	Number(기본값:0)	아래 여백
paddingLeft	Number(기본값:0)	왼쪽 여백
paddingRight	Number(기본값:0)	오른쪽 여백

<표 4 rMateChart 노드 속성과 유효값 설명>

아래는 rMateChart 노드의 속성 사용 예를 나타낸 것 입니다.

```
<rMateChart cornerRadius="12" borderStyle="solid" backgroundColor="0xFFFF77">
    <Options>
        <Caption text="Anual Report" />

            ......
```

```
</rMateChart>
```

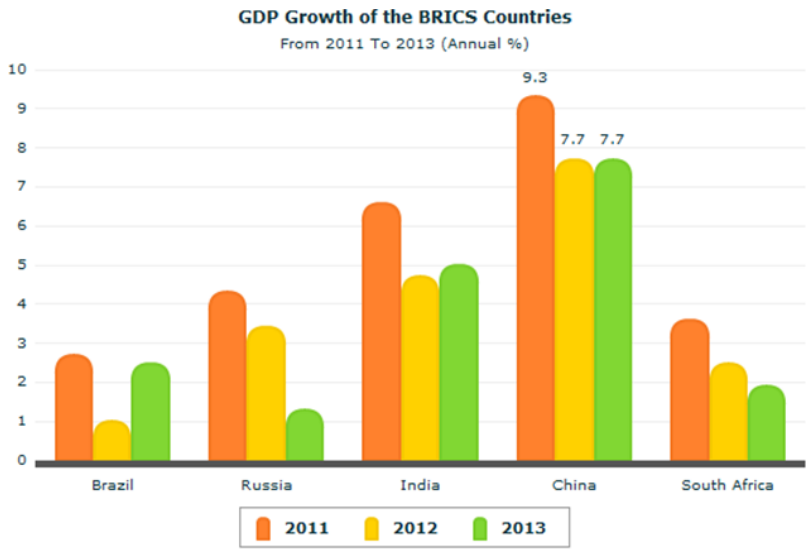
<예제 15 rMateChart 노드의 속성 사용 예>

5.4. 레이아웃의 Options 노드 설정하기.

Options 는 레이아웃의 필수사항이 아닌 선택사항입니다. 차트에 제목이나 부제목, 범례(Legend)를 삽입하고자 할 때 작성하십시오.

```
<rMateChart>
        <Options>
                <Caption text="Annual Report"/>   // 제목
                <SubCaption text="2008"/>   //부제목
                <Legend/>   //범례
        </Options>
        <Column3DChart showDataTips="true" width="100%" height="100%" >
            <series>
                <Column3DSeries yField="Profit" displayName="Profit">
                    .
                  .
                  .
        </Column3DChart>
</rMateChart>
```

<예제 16 Options 태그 사용 예>



<그림 6 Options 태그를 사용하여 제목과 부제목, 범례를 표현한 차트>

5.4.1. 차트 제목(Caption), 부제목(SubCaption) 넣기.

<예제 16 Options 태그 사용 예>는 칼럼 3D 차트의 레이아웃입니다. 제목은 <Caption>태그의 text 속성에 원하는 문구를 넣으시면 됩니다.

Caption 태그와 SubCaption 태그의 속성에 대한 설명은 아래 표와 같습니다.

속성 명	유효 값	설명
text	문자열	원하는 텍스트를 입력하세요.
fontSize	숫자 (기본값 : 12, 부제목 : 10)	텍스트의 사이즈를 나타냅니다.
fontWeight	bold | normal (기본값: normal)	볼드를 결정합니다.
fontStyle	normal | italic(기본값:normal)	이텔릭 를 결정합니다.
textDecoration	none | underline(기본값: none)	Underline을 결정합니다.
color	RGB 컬러(기본값: 0x000000)	글자색을 결정합니다.
textAlign	center | left | right (기본값:center)	글자의 위치를 결정합니다.
paddingLeft	숫자 (기본값 : 0)	왼쪽 여백.
paddingRight	숫자 (기본값 : 0)	오른쪽 여백.
paddingTop	숫자 (기본값 : 0)	윗 여백.
paddingBottom	숫자 (기본값 : 0)	아래 여백.

<표 5 제목과 부제목 속성과 유효값 설명>

5.4.2. 차트 범례(Legend) 넣기.

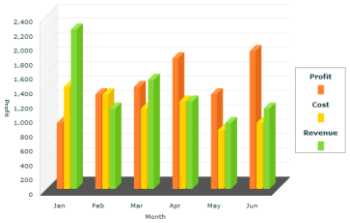
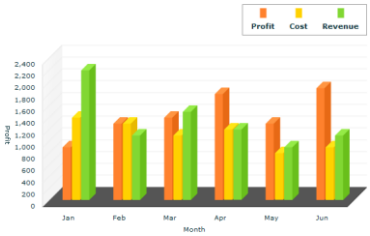
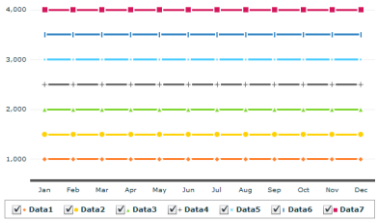
차트의 범례(Legend)는 보통 멀티시리즈(다중데이터 표현)인 경우 각 시리즈가 표현하는 데이터의 대표자를 표시하는 역할을 합니다.
범례(Legend) 작성시에는 차트시리즈(즉, 칼럼 3D 차트인 경우 <Column3DSeries> 태그)의 속성인 displayName 을 정의 하여야 합니다. 여기서 정의한 문자열이 범례의 텍스트로 출력됩니다.

아래 표는 범례의 속성을 나타냅니다.

속성 명	유효 값	설명
position	bottom | left | right | top (기본값 : bottom)	차트를 기준으로 범례가 위치할 곳을 정합니다.
direction	horizontal | vertical (기본값:horizontal)	범례의 방향을 정합니다. 예를 들어 3개의 시리즈가 존재하여 3개의 범례 아이템이 존재한다면 이 3개 아이템들의 방향을 정합니다.
hAlign	center | left | right (기본값 : center)	position 에 의해 정해진 범례의 위치에서 구체적으로 수평방향 정렬방식을

		정합니다.
vAlign	middle | top | bottom (기본값 : middle)	position에 의해 정해진 범례의 위치에서 구체적으로 수직방향 정렬방식을 정합니다.
labelPlacement	right | left | top | bottom (기본값:right)	범례에서 색을 나타내는 도형을 기준으로 텍스트 글자의 구체적인 위치를 정합니다.
markerWidth	숫자 (기본값 : 10)	범례의 마커 폭
markerHeight	숫자 (기본값 : 15)	범례의 마커 높이
backgroundColor	RGB 컬러(기본값:0xFFFFFF)	배경색입니다.
color	RGB 컬러(기본값:0x000000)	글자색을 나타냅니다.
fontSize	숫자 (기본값 : 12, 부제목 : 10)	텍스트의 사이즈를 나타냅니다.
fontWeight	bold | normal (기본값: bold)	볼드를 결정합니다.
fontStyle	normal | italic(기본값:normal)	이텔릭 를 결정합니다.
textDecoration	none | underline(기본값: none)	Underline을 결정합니다.
useVisibleCheck	true | false (기본값:true)	범례 옆에 체크 박스 표시를 지정합니다. 체크박스는 시리즈 visible on/off 기능을 합니다.
paddingLeft	숫자 (기본값 : 0)	왼쪽 여백
paddingRight	숫자 (기본값 : 0)	오른쪽 여백
paddingTop	숫자 (기본값 : 0)	윗쪽 여백
paddingBottom	숫자 (기본값 : 0)	아래쪽 여백
verticalGap	숫자 (기본값 : 6)	범례 아이템들의 세로 간격 사이 여백을 나타냅니다.
horizontalGap	숫자 (기본값 : 8)	범례 아이템들의 가로 간격 사이 여백을 나타냅니다.
defaultMouseOverAction	Boolean(기본값:false)	디폴트 범례 아이템 마우스 오버(Mouse Over) 액션을 취할지 여부를 나타냅니다. 디폴트 범례 마우스 오버(Mouse Over) 액션은 시리즈의 범례에 마우스 오버 시 해당 시리즈만 표시합니다.
defaultCheckBoxAction	Boolean(기본값:false)	useVisibleCheck 속성을 활성화 한 경우 디폴트 범례 체크 박스 아이템 클릭 액션을 취합니다. 디폴트 범례 체크 박스 아이템 클릭 액션은 해당 시리즈의 범례 체크 박스 selected 상태와 해당 시리즈의 visible 을 일치시킵니다.

<표 6 범례(Legend) 속성과 유효 값 설명>

출력결과			
적용속성	position="right" direction="vertical" useVisibleCheck="true" vAlign="middle"	position="top" direction="horizontal" useVisibleCheck="true" hAlign="right"	position="left" direction="vertical" useVisibleCheck="false" vAlign="bottom"

<예제 17 다양한 범례 속성을 적용한 예>

5.4.3. 데이터 에디터 사용하기

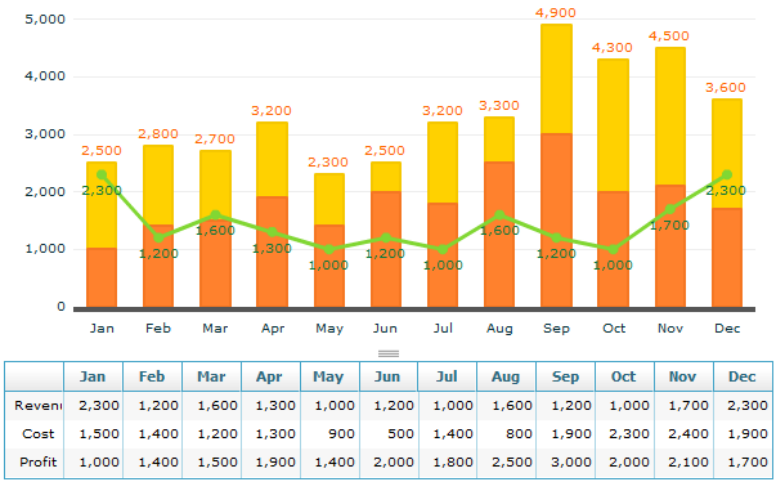
데이터 에디터는 차트 하단에 출력되며, 에디터의 셀을 클릭, 더블클릭하여 차트 데이터의 확인, 수정이 가능케 합니다.
차트에서 데이터 에디터를 사용하는 법은 간단합니다. flashVars 정의 시 useDataEditor=true 로 설정하면 바로 데이터 에디터를 사용할 수 있습니다.(flashVars 정의는 "4.1. flashVars 설정" 를 참고하십시오.)

useDataEditor=true 로 설정 후 DataEditor 태그를 통하여 데이터에디터의 구체적인 속성을 설정할 수 있습니다. 다음은 그 속성에 대한 설명 표입니다.

속성 명	유효 값	설명
text	문자열	원하는 텍스트를 입력하세요.
fontSize	숫자 (기본값 : 12, 부제목 : 10)	텍스트의 사이즈를 나타냅니다.
fontFamily	폰트명	출력할 폰트명을 나타냅니다.
fontWeight	bold | normal (기본값: bold)	볼드를 결정합니다.
fontStyle	normal | italic(기본값:normal)	이텔릭 를 결정합니다.
textDecoration	none | underline(기본값: none)	Underline을 결정합니다.
color	RGB 컬러(기본값: 0x000000)	글자색을 결정합니다.
textAlign	center | left | right (기본값:center)	글자의 위치를 결정합니다.
showOnInit	Boolean(기본값:false)	기본적으로 데이터에디터를 표시할지 여부.
reverseRow	Boolean(기본값:false)	차트 데이터의 순서와 반대로 Row를 위치시킬지 여부를 나타냅니다.
minColumnWidth	Number(픽셀단위, 기본값:40)	열(Column)의 가로 최소 사이즈를

		정합니다.
editorHeight	Number(픽셀단위, 기본값:100)	데이터에디터의 전체 높이를 나타냅니다.
headerHeight	Number(픽셀단위, 기본값:22)	헤더의 height을 지정합니다.
showHeaders	Boolean(기본값:true)	헤더 유무를 나타냅니다.
alternatingItemColors	RGB 배열	Row 단위의 배경색을 지정합니다.(반드시 2개 이상의 RGB 배열로 삽입)
headerColors	RGB 배열	헤더 배경색 지정(반드시 2개 이상의 RGB 배열로 삽입)
horizontalGridLines	Boolean(기본값:false)	가로 눈금줄 표시 조정
horizontalGridLineColor	RGB	가로 눈금줄 색을 지정합니다.
verticalAlign	top, middle, bottom(기본값:top)	셀 세로 정렬을 지정합니다.
verticalGridLines	Boolean(기본값:true)	상하 눈금 줄 표시를 지정합니다
verticalGridLineColor	RGB	상하 눈금 줄 색을 지정합니다.
borderColor	RGB	테두리색을 지정합니다.
borderStyle	none, solid, inset, outset(기본값:solid)	테두리의 모양을 지정합니다.
borderThickness	Number(픽셀단위)	테두리 두께를 지정합니다.('solid' 인 경우 유효)
selectionColor	RGB	선택된 셀 또는 행의 바탕색 지정
rollOverColor	RGB	마우스 오버된 셀 또는 행의 바탕색 지정
useCloseButton	Boolean(기본값:true)	데이터 에디터 오른쪽 상단에 에디터 닫기 버튼을 표시할지 여부를 나타냅니다.

<표 7 데이타에디터 속성 및 유효값 설명>



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Reveni	2,300	1,200	1,600	1,300	1,000	1,200	1,000	1,600	1,200	1,000	1,700	2,300
Cost	1,500	1,400	1,200	1,300	900	500	1,400	800	1,900	2,300	2,400	1,900
Profit	1,000	1,400	1,500	1,900	1,400	2,000	1,800	2,500	3,000	2,000	2,100	1,700

<그림 7 데이타에디터 사용으로 설정한 모습>

5.5. 레이아웃의 Style 노드(CSS 적용) 설정하기.

<Style> 노드는 전반적인 스타일을 미리 정의하는 스타일시트입니다. <Style/>을 선언하는 방법에는 몇 가지 규칙이 있습니다.

첫째, <Style/> 노드는 반드시 <rMateChart> 노드의 자식 위치에 정의하십시오.
둘째, 스타일네임을 정의 할 때 시작은 도트(.)을 찍고 소문자로 시작하는 영문자로 표기하십시오.
셋째, 스타일네임을 정한 후 그에 따른 구체적인 스타일을 정의 할 때는 반드시 중괄호({,})로 시작과 끝을 표시하십시오.
넷째, 속성명과 속성값(value)의 구분은 콜론(:)이며 끝은 세미콜론으로 나타냅니다.

다음은 올바른 예와 잘못된 예를 나타낸 것입니다.

올바른 예	잘못 작성 된 예

<rMateChart>
<Options>
 	
...
 	
<Style>
.rMateChartStyle
{
backgroundColor:0xFFFFFE;
borderColor:0x77EE9E;
cornerRadius:12;
borderThickness:3;
borderStyle:solid;
 	
}
<Style>
</rMateChart>
	<rMateChart>
	<Options>
	
	...
	
	<Style>
	ChartStyle
	[
	backgroundColor:0xFFFFFE;
	borderColor:"0x77EE9E";
	cornerRadius,12;
	borderThickness:3;
	borderStyle:'solid';
	
	]
	
	<Style>
	</rMateChart>

<표 8 Style 작성 예>

위와 같이 rMateChartStyle 스타일을 정의였습니다. 그렇다면 이 스타일을 적용하는 방법에 대해 보도록 하겠습니다.

```
<rMateChart styleName="rMateChartStyle">
      <Options>
            <Caption text="Annual Report"/>
      </Options>

       ......

      <Style>
      .rMateChartStyle
      {
      backgroundColor:0xFFFFFE;
      borderColor:0x77EE9E;
      cornerRadius:12;
```

```
        borderThickness:3;
        borderStyle:solid;

        }
        <Style>

</rMateChart>
```

<예제 18 Style 적용 예>

<예제 18 Style 적용 예>를 보시면 rMateChartStyle 은 <rMateChart/> 노드의 스타일을 정의 한 것이고 적용법은 styleName="정의한 스타일이름" 즉, styleName="rMateChartStyle" 과 같습니다.
<Style/>에서 정의 가능한 각각의 속성은 다음과 같습니다.

이 름	정의 가능한 속성 명	설명 참조 표	비 고
rMateChart	backgroundColor backgroundAlpha borderStyle borderColor borderThickness cornerRadius paddingTop paddingBottom paddingLeft paddingRight	<표 4 rMateChart 노드 속성과 유효값 설명>	
● Caption ● SubCaption ● 차트(예:ColumChart)	fontSize fontWeight fontStyle textDecoration color fontFamily textAlign paddingLeft paddingRight paddingTop paddingBottom	<표 5 제목과 부제목 속성과 유효값 설명>	차트는 모든 차트를 뜻합니다. 예를 들면 Column3DChart, Pie3DChart, Bubble3DChart 등등의 대표 이름입니다.

● 차트의 Series ● 차트의 axisTitleStyleName	fontFamily fontSize fontStyle fontWeight labelAlign(칼럼, 바시리즈만 해당) labelPosition		axisTitleStyleName 의 속성 및 유효값은 Caption 과 동일합니다.

<표 9 Style 에서 정의 가능한 속성>

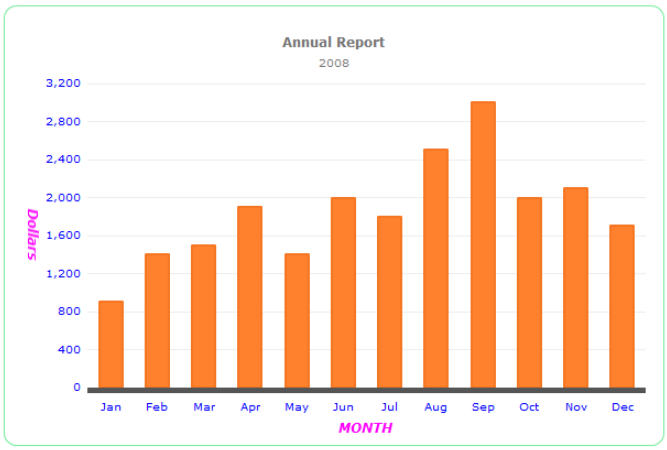
시리즈의 labelAlign 은 칼럼시리즈와 바시리즈에만 해당되는 속성입니다. 각각의 유효 속성값으로 칼럼시리즈인 경우 "top | middle | bottom" 바시리즈인 경우 "left | center | right" 입니다.

스타일을 활용한 예제와 출력 모습은 아래와 같습니다.

```
<rMateChart styleName="rMateChartStyle">
        <Options>
                <Caption text="Annual Report" styleName="captionStyle"/>
                <SubCaption text="RiaMore Soft" styleName="subCaptionStyle"/>
        </Options>
        <DateFormatter id="dateFmt" formatString="M/D"/>
        <NumberFormatter id="numFmt"/>
        <Line2DChart showDataTips="true" styleName="chartStyle" axisTitleStyleName="chartAxisStyle">
                <horizontalAxis>
                        <DateTimeAxis id="hAxis" dataUnits="days" labelUnits="days" formatter="{dateFmt}"
title="DateTime Axis"
                        interval="2" displayName="Date" alignLabelsToUnits="false" displayLocalTime="true"/>
                </horizontalAxis>
                <verticalAxis>
        <LogAxis id="vAxis" title="Log Axis" formatter="{numFmt}" interval="10" minimum="10"
maximum="10000"/>
                </verticalAxis>
                <series>
                        <Line2DSeries xField="Date" yField="Revenue" displayName="Revenue">
                                <showDataEffect>
                                        <SeriesInterpolate/>
                                </showDataEffect>
                        </Line2DSeries>
                </series>
        </Line2DChart>
        <Style>
                .rMateChartStyle    //rMateChart 스타일 정의
                {
```

```
                backgroundColor:0xFFFFFE;
                borderColor:0x77EE9E;
                cornerRadius:12;
                borderThickness:3;
                borderStyle:solid;
        }

        .captionStyle   //제목 스타일 정의
        {
                fontSize:12;
                fontFamily:Tahoma;
                fontWeight:bold;
                color:0x777777;
        }

        .subCaptionStyle   //부제목 스타일 정의
        {
                fontSize:11;
                fontStyle:italic;
                color:0x777777;
        }


        .chartStyle   //차트 스타일 정의
        {
                fontSize:11;
                fontStyle:italic;
                color:0x0000FF;
        }

        .chartAxisStyle   //차트 축의 제목(대표문자) 스타일 정의
        {
                fontSize:13;
                fontWeight:bold;
                color:0xFF00FF;
        }
    </Style>
</rMateChart>
```

<예제 19 Style 적용한 예제>



<그림 8 Style 적용 결과>

6. 차트 유형 별 레이아웃 설정하기.

6.1. 차트의 공통적인 속성 알아보기.

차트는 크게 축이 존재하는 Cartesian 차트와 축이 없는 Polar 차트로 나뉩니다. 축이 있는 차트의 기본적인 속성에 대한 설명은 아래와 같습니다.

속성 명	유효 값	설명
horizontalAxis	CategoryAxis, LinearAxis, DateTimeAxis, LogAxis 객체	수평축(X축)을 정의합니다.
verticalAxis	CategoryAxis, LinearAxis, DateTimeAxis, LogAxis 객체	수직축(Y축)을 정의합니다.
horizontalAxisRenderers	Axis3DRenderer, Axis2DRenderer	수평축의 렌더러를 정의합니다.
verticalAxisRenderers	Axis3DRenderer, Axis2DRenderer	수직축의 렌더러를 정의합니다.
series	차트의 시리즈 (예:Column3DSeries, Line2DSeries 등등)	차트 시리즈를 정의합니다.
backgroundElements	GridLines	차트 축을 기준으로 안쪽의 뒷배경을 정의합니다.
showDataTips	true | false	마우스오버시 데이터팁(툴팁)의 유무를 나타냅니다.
paddingTop	Number	위쪽 여백
paddingBottom	Number	아래 여백
paddingLeft	Number	왼쪽 여백
paddingRight	Number	오른쪽 여백
itemClickJsFunction	자바스크립트 함수명	아이템 클릭 시 차트가 호출할 함수를 나타냅니다.
dataTipJsFunction	자바스크립트 함수명	데이터팁(툴팁)을 사용자 정의할 함수를 나타냅니다.
width	Number | 퍼센티지	가로 사이즈
height	Number | 퍼센티지	세로 사이즈
gutterLeft	Number	차트 축을 기준으로 왼쪽 여백
gutterRight	Number	차트 축을 기준으로 오른쪽 여백
gutterTop	Number	차트 축을 기준으로 윗쪽 여백
gutterBottom	Number	차트 축을 기준으로 아래 여백

<예제 20 Cartesian 차트의 공통적인 속성 설명>

아래는 도넛, 파이차트와 같이 축이 없는 Polar 차트의 속성에 대한 설명입니다.

속성 명	유효 값	설명
innerRadius	0.0 ~ 1.0(기본값:0.0)	도넛차트와 같이 가운데 공간의 정도를 표현합니다. 1.0 에 가까울수록 가운데 공간이 커집니다.
showDataTips	true | false(기본값:false)	마우스오버시 데이터팁(툴팁)의 유무를 나타냅니다.
explodable	true | false(기본값:true)	마우스 클릭 시 도넛, 파이차트의 조각이 떨어져 나오는 효과를 나타냅니다.
series	Pie2DSeries, Pie3DSeries	차트의 시리즈를 정의합니다.
width	Number | 퍼센티지	가로 사이즈
height	Number | 퍼센티지	세로 사이즈
itemClickJsFunction	자바스크립트 함수명	아이템 클릭 시 차트가 호출할 함수를 나타냅니다.
dataTipJsFunction	자바스크립트 함수명	데이터팁(툴팁)을 사용자 정의할 함수를 나타냅니다.

<예제 21 Polar 차트의 공통적인 속성 설명>

6.2. Cartesian 차트의 축에 대하여.

파이차트, 도넛차트 외의 차트에는 X 축(수평축)과 Y 축(수직축)이 존재합니다. 축의 종류에는 크게 카테고리 유형과 수치 유형 2 개가 있습니다. 스트링 형태를 정의하는 축이 CategoryAxis 입니다. 이는 보통 차트에서 수치화 할 수 없지만 그룹화가 가능한 계열을 위한 것입니다. 예를 들면 경영부, 연구부, 회계부와 같이 수치화 될 수 없지만 일개 그룹으로 묶어 축 필드로 사용하고자 할 때 CategoryAxis 를 사용합니다.
수치 유형으로는 LinearAxis, LogAxis, DateTimeAxis 3 가지 Type 이 있습니다. LinearAxis 는 연속적인 데이터(즉, 일반적인 수치)를 위한 것이고, LogAxis 는 Log 함수에 의한 정의입니다. 마지막으로 DateTimeAxis 는 날짜, 시간과 관련이 있습니다.

아래 표는 각각의 축에 대한 속성 명과 유효 값에 대한 설명입니다.

Axis 명	속성 명	유효 값	설명
CategoryAxis	categoryField	데이터의 특정 필드 명 (예:Month)	카테고리축이 참조할 데이터의 특정 필드명을 입력하세요. 카테고리축을 정의하였을 때 반드시 입력하셔야 합니다.
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는

			문자열입니다.
	title	Stirng(문자열)	축의 제목(대표 문자) 입니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.
LinearAxis	interval	Number	축에 나타나는 라벨의 간격을 나타냅니다.
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.
	minimum	Number	축 라벨의 최소치를 나타냅니다.
	maximum	Number	축 라벨의 최대치를 나타냅니다.
	title	Stirng(문자열)	축의 제목(대표 문자) 입니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.
DateTimeAxis	dataUnits	milliseconds, seconds, minutes, hours, days, weeks, months, years	차트가 출력할 데이터의 표시에 사용하는 단위를 지정합니다
	labelUnits	milliseconds, seconds, minutes, hours, days, weeks, months, years	라벨에 나타날 단위를 지정합니다.
	title	Stirng(문자열)	축의 제목(대표 문자) 입니다.
	interval	Number	축에 나타나는 라벨의 시간, 날짜 간격을 나타냅니다. (라벨이 나타날 공간이 작을 경우 지정된 interval 값은 무시될 수 있습니다.)
	dataInterval	Number	dataUnits 로 지정된 그래프내의 데이터간의 간격을 지정합니다. 예를 들어 dataUnits="seconds"로 설정하고 데이터 간격은 3초 단위라면 dataInterval은 4로 줄 필요가 있습니다. 단위가 초단위이므로 차트는 그 자리를 마련합니다. dataInterval="4"는 3초 단위를 하나로 본다는 뜻으로 4번째 자리부터 그래프를 렌더링합니다. (차트 type 에 따라 무시되기도 합니다.)
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.
	displayLocalTime	false|true(기본값:false)	true 로 설정되었을 경우는, DateTimeAxis 에 의해, 모든 데이터치가 어플리케이션을 실행하는 클라이언트 머신의 타임 존에 있다고 보여집니다. false 로 설정되었을

			경우, 모든 값은 세계 표준시 (그리니지 표준시)가 됩니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.
LogAxis	interval	10의 승수	10의 승수 단위로 라벨에 표시합니다.
	minimum	Number	축 라벨의 최소치를 나타냅니다.
	maximum	Number	축 라벨의 최대치를 나타냅니다.
	title	Stirng(문자열)	축의 제목(대표 문자) 입니다.
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.

<표 10 각각의 Axis 속성 명과 유효 값 설명>

6.2.1. CategoryAxis 와 LinearAxis 예제.

아래 예제를 보면 수평축으로 CategoryAxis 를 정의하였습니다. CategoryAxis 의 categoryField 는 반드시 차트 데이터와 일치시켜야 합니다. 데이터의 Month 필드를 수평축의 카테고리로 활용한다는 뜻이 됩니다. 만약 축을 정의 하지 않으면 기본값으로 LinearAxis 가 됩니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Anual Report"/>
  </Options>
  <Column3DChart showDataTips="true">

    <horizontalAxis>  //수평축으로 카테고리축 정의
        <CategoryAxis categoryField="Month" title="Category Axis" />
    </horizontalAxis>

    <verticalAxis>  // 수직축으로 선형축 정의
        <LinearAxis maximum="3500" title="Linear Axis"/>
    </verticalAxis>
    <series>
        <Column3DSeries yField="Profit" displayName="Profit">
            <showDataEffect>
                <SeriesInterpolate />
            </showDataEffect>
        </Column3DSeries>
    </series>
  </Column3DChart>
</rMateChart>
```

<예제 22 수평축으로 카테고리축을, 수직축으로 리니어축을 정의한 예>

6.2.2. DateTimeAxis 와 LogAxis 예제.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
      <Caption text="Anual Report"/>
  </Options>
  <Line2DChart showDataTips="true">

      <horizontalAxis> // 수평축으로 DateTime축 정의
          <DateTimeAxis dataUnits="days" labelUnits="days" title="DateTime Axis" interval="3"
          displayName="Date" displayLocalTime="true"/>
      </horizontalAxis>

      <verticalAxis> //수직축으로 로그축 정의
          <LogAxis title="Log Axis" interval="10" minimum="10" maximum="10000" />
      </verticalAxis>
      <series>
          <Line2DSeries xField="Date" yField="Profit" displayName="Profit"/>
      </series>
  </Line2DChart>
</rMateChart>
```

<예제 23 수평축으로 DateTime 축을, 수직축으로 로그축을 정의한 예제>

***중요** : **DateTimeAxis 를 사용한 경우 반드시 시리즈 DateTime Field 를 명시해 줘야 합니다.** 예를 들어 칼럼차트의 수평축에 DateTimeAxis 를 정의 한 경우 xField 를 정의해야 합니다.(바차트의 수직축이 DateTimeAxis 인 경우 yField) 이 xField 는 데이터의 시간영역에 해당되는 필드명을 나타냅니다. 이것은 CategoryAxis 의 cateogoryField 와 동일한 의미이지만 DateTimeAxis 의 속성이 아닌 시리즈의 xField(또는 yField)에 해당 필드를 정의함으로써 DateTimeAxis 는 데이터 시간에 맞게 표현합니다.

6.3. 칼럼 2D 차트

칼럼 2D 차트는 <Column2DChart> 노드로 시작과 끝을 나타냅니다. <Column2DChart/> 노드의 자식 노드로 수평 축(horizontalAxis), 수직 축(verticalAxis), 시리즈(series), 배경엘레멘츠(backgroundElements) 등을 정의합니다.

아래 예제는 싱글 데이터(싱글 시리즈) 칼럼 2D 차트 예제입니다.
<Column2DChart>의 <series> 자식 노드로 <Column2DSeries>를 정의 한 것을 볼 수 있습니다. 이는 데이터 표현을 칼럼시리즈로 한다는 뜻입니다. **모든 차트의 데이터 표현은 시리즈가 담당하고 있습니다.**

그래서 시리즈의 속성을 잘 정의 하는 것이 중요합니다. 시리즈의 부모 노드 겪인 Chart(예를 들면 Column2DChart, Bar2DChart 등등)는 시리즈의 자리배치와 축의 위치, 배경 등 외형을 담당합니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Anual Report"/>
        </Options>
        <Column2DChart showDataTips="true">
                <horizontalAxis>
                        <CategoryAxis categoryField="Month"/>
                </horizontalAxis>
                <series>
                        <Column2DSeries yField="Profit" itemRenderer="SemiCircleColumnItemRenderer">
                                <showDataEffect>
                                        <SeriesInterpolate/>
                                </showDataEffect>
                            <fill>
                                <SolidColor color="0xFF0000" alpha="0.5"/>
                            </fill>
                            <stroke>
                                <Stroke color="0xFFFF00" weight="1"/>
                            </stroke>
                        </Column2DSeries>
                </series>
        </Column2DChart>
</rMateChart>
```

<예제 24 칼럼 차트 예제>

속성 명	유효 값	설명
type	clustered | stacked | overlaid | 100% (기본값:clustered)	차트 데이터 표현 유형을 지정합니다.(3D 칼럼차트 type 그림 참고)

<표 11 칼럼 차트의 속성 설명>

<Column2DSeries> 속성에 대한 설명입니다.

속성 명	유효 값	설명
xField	차트 데이터의 특정 필드	X축의 위치를 결정하는 차트데이터의 필드를 지정합니다.
yField	차트 데이터의 특정 필드	Y축의 위치를 결정하는 차트데이터의 필드를 지정합니다.

itemRenderer	아래 설명 참고	
showDataEffect	SeriesZoom, SeriesSlide, SeriesInterpolate	차트 데이터가 표현될 때 나타나는 효과를 정의합니다.
fill	SolidColor	시리즈의 컬러를 결정합니다.
fills	RGB 컬러의 배열 | SolidColor 의 배열	시리즈 아이템 각각의 컬러를 결정합니다.
stroke	Stroke	시리즈의 선을 나타냅니다.
labelPosition	inside | outside | both	수치 표시를 나타낼 곳을 지정 합니다.
insideLabelJsFunction	자바스크립트 함수명	labelPosition이 inside인 경우의 수치표시 사용자 정의 함수 입니다.
outsideLabelJsFunction	자바스크립트 함수명	labelPosition이 outside인 경우의 수치표시 사용자 정의 함수 입니다.
fillJsfunction	자바스크립트 함수명	칼럼 시리즈 아이템의 채우기 색 사용자 정의 함수입니다.
columnWidthRatio (Bar 차트라면 barWidthRatio)	0 ~ 1	컬럼의 그리는 영역의 비율을 지정합니다. 0이면 컬럼을 그리지 않고 1이면 컬럼이 그릴 수 있는 영역 전체를 그리게 됩니다.
maxColumnWidth (Bar 차트라면 maxBarWidth)	Number	이 속성은 columnWidthRatio의 영향을 받습니다. columnWidthRatio의 비율 영역안에서 이 속성에 정의한 px만큼 그리게 됩니다.

<표 12 칼럼차트 시리즈의 속성 설명>

<Column2DSeries> 노드의 itemRenderer 는 차트의 시리즈가 어떻게 데이터를 그래픽적으로 표현할지를 결정합니다. ColumnSeries 노드의 itemRenderer 의 유효 속성값으로는 두개가 존재합니다.

- SemiCircleColumnItemRenderer 
- BoxItemRenderer 

6.4. 칼럼 3D 차트

칼럼 3D 차트는 <Column3DChart> 노드로 시작과 끝을 나타냅니다. 칼럼 3D 차트의 시리즈는 <Column3DSeries>입니다.

속성 명	유효 값	설명
type	clustered | stacked | overlaid | 100% (기본값:clustered)	차트 데이터 표현 유형을 지정합니다.(3D 칼럼차트 type 그림 참고)
cubeDepth	0 ~ 30	가상 Z축의 길이
cubeAngleRatio	0 ~ 9	시선의 높낮이

<표 13 칼럼 3D 차트의 속성 설명>

속성 명	유효 값	설명
xField	차트 데이터의 특정 필드	X축의 위치를 결정하는 차트데이터의 필드를 지정합니다.
yField	차트 데이터의 특정 필드	Y축의 위치를 결정하는 차트데이터의 필드를 지정합니다.
showDataEffect	SeriesZoom, SeriesSlide, SeriesInterpolate	차트 데이터가 표현될 때 나타나는 효과를 정의합니다.
fill	SolidColor	시리즈의 컬러를 결정합니다.
fills	RGB 컬러의 배열 | SolidColor 의 배열	시리즈 아이템 각각의 컬러를 결정합니다.
stroke	Stroke	시리즈의 선을 나타냅니다.
labelPosition	inside | outside | both	수치 표시를 나타낼 곳을 지정 합니다.
insideLabelJsFunction	자바스크립트 함수명	labelPosition이 inside인 경우의 수치표시 사용자 정의 함수 입니다.
outsideLabelJsFunction	자바스크립트 함수명	labelPosition이 outside인 경우의 수치표시 사용자 정의 함수 입니다.
fillJsfunction	자바스크립트 함수명	칼럼 시리즈 아이템의 채우기 색 사용자 정의 함수입니다.

<표 14 칼럼 3D 차트 시리즈의 속성 설명>

아래는 rMate 멀티시리즈 칼럼 3D 차트 레이아웃 예제입니다.

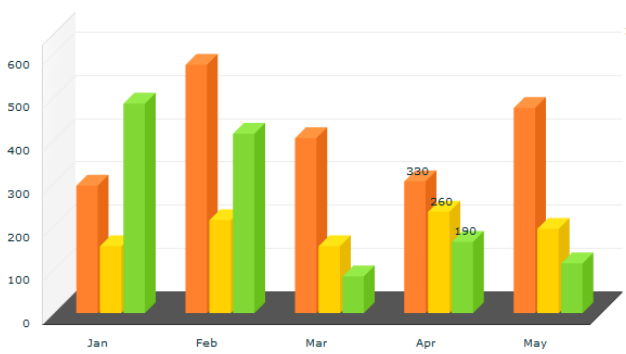
```
<rMateChart>
    <Column3DChart showDataTips="true" type="clustered">
            //칼럼차트 생성, 툴팁 보이기 true
    <horizontalAxis//X축 설정, X축은 카테고리이며 기준 필드는 Month
        <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series> //시리즈는 데이터의 표현을 말합니다.
        // 칼럼시리즈가 2개 반복 → Profit, Cost 데이터의 표현이 한쌍임.
       <Column3DSeries yField="Profit">
                        <showDataEffect> //차트 생성시 이펙트 정의
                        <SeriesInterpolate/>
                        </showDataEffect>
        </Column3DSeries>
        <Column3DSeries yField="Cost"> // 차트데이터 값 중 Cost를 y축필드에 대응시킵니다.
                        <showDataEffect>
                          <SeriesInterpolate/>
                        </showDataEffect>
        </Column3DSeries>
  </series></Column3DChart></rMateChart>
```

<표 15 칼럼 3D 차트 레이아웃 예제>

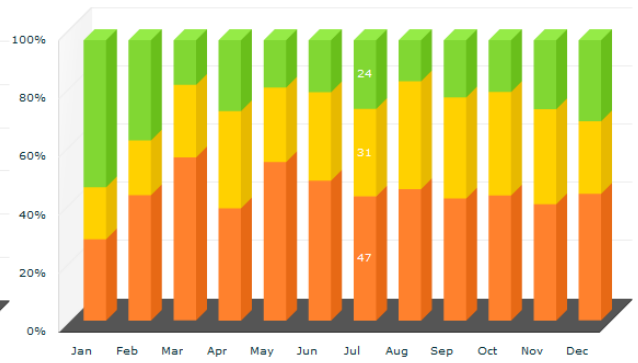
칼럼 차트의 프로퍼티 중 type 은 차트의 데이터 출력 방식을 결정합니다. type 에는 4 가지가 존재합니다.

- clustered : 일반적인 다중데이터(차트의 멀티시리즈) 방식으로 데이터를 표현합니다. (기본값)
- stacked : 데이터를 위에 쌓아 올린 방식으로 표현합니다.
- overlaid : 수치 데이터 값을 겹쳐서 표현 합니다. 주로 목표 위치와 현재 위치를 나타낼 때 많이 쓰입니다.
- 100% : 차트의 수치 데이터를 퍼센티지로 계산 후 값을 퍼센티지로 나타냅니다.

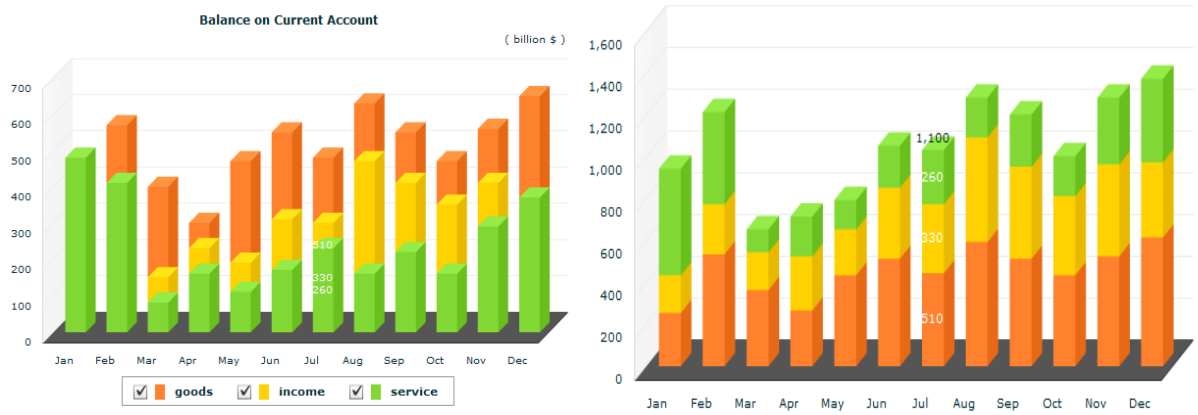
다음은 이에 대한 출력 화면입니다.



<type="clustered">



<type="100%">



<type="overlaid"> <type="stacked">

<그림 9 칼럼차트 Type 별 출력 결과>

6.5. 실린더 3D 차트

6.5.1. 실린더 3D 칼럼 차트

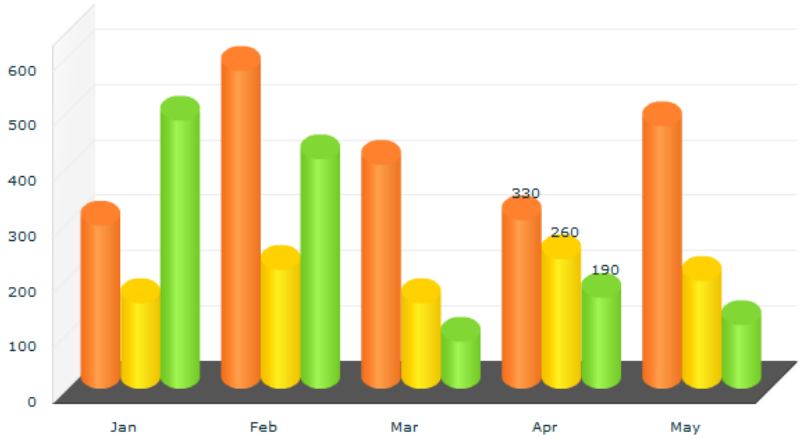
실린더 3D 칼럼 차트는 칼럼 3D 차트와 같습니다. 다만, 큐브 형태의 칼럼이 아닌 3D 원통 모양으로 데이터를 표현합니다.
모든 속성 및 레이아웃 작성은 3D 칼럼 차트와 동일 합니다.
작성된 3D 칼럼 차트에서 시리즈의 속성인 itemRenderer 속성에 CylinderItemRenderer 를 값으로 할당하면 실린더 3D 차트가 생성됩니다.

다음은 <표 15 칼럼 3D 차트 레이아웃 예제> 레이아웃에 아이템렌더러 속성을 정의한 예제입니다.

```
<rMateChart>
    <Column3DChart showDataTips="true" type="clustered">
            //칼럼차트 생성, 툴팁 보이기 true
    <horizontalAxis//X축 설정, X축은 카테고리이며 기준 필드는 Month
        <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series> //시리즈는 데이터의 표현을 말합니다.
        // 칼럼시리즈가 2개 반복 → Profit, Cost 데이터의 표현이 한쌍임.
        <Column3DSeries yField="Profit" itemRenderer="CylinderItemRenderer">
                    <showDataEffect> //차트 생성시 이펙트 정의
                    <SeriesInterpolate/>
                    </showDataEffect>
        </Column3DSeries>
        <Column3DSeries yField="Cost"// 차트데이터 값 중 Cost를 y축필드에 대응시킵니다.
itemRenderer="CylinderItemRenderer">
                    <showDataEffect>
```

```
                    <SeriesInterpolate/>
                </showDataEffect>
            </Column3DSeries>
        </series></Column3DChart></rMateChart>
```

<예제 25 실린더 3D 차트 레이아웃 예제>



<그림 10 실린더 3D 차트 결과 모습>

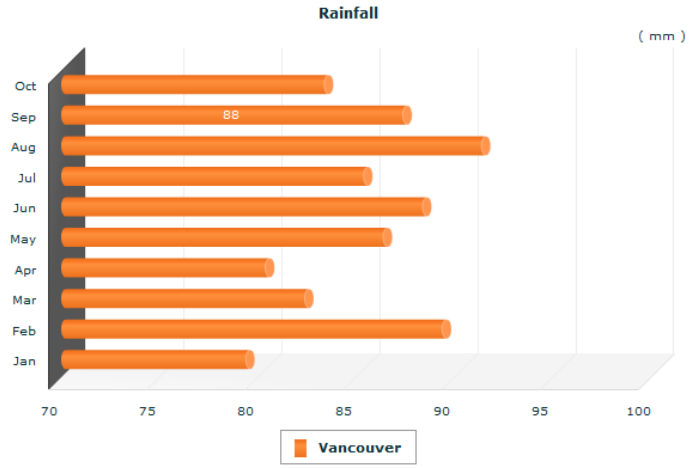
6.5.2. 실린더 3D 바차트

실린더 3D 바 차트는 바 3D 차트와 같습니다. 다만, 큐브 형태의 칼럼이 아닌 3D 원통 모양으로 데이터를 표현합니다.

모든 속성 및 레이아웃 작성은 3D 바 차트와 동일 합니다.

작성된 3D 바 차트에서 시리즈의 속성인 itemRenderer 속성에 **BarCylinderItemRenderer** 를 값으로 할당하면 실린더 3D 차트가 생성됩니다

레이아웃 작성은 “6.5.1 실린더 3D 칼럼 차트” 를 참고하여 주십시오.



<그림 11 실린더 3D 바차트 출력 모습>

6.6. 바 2D 차트.

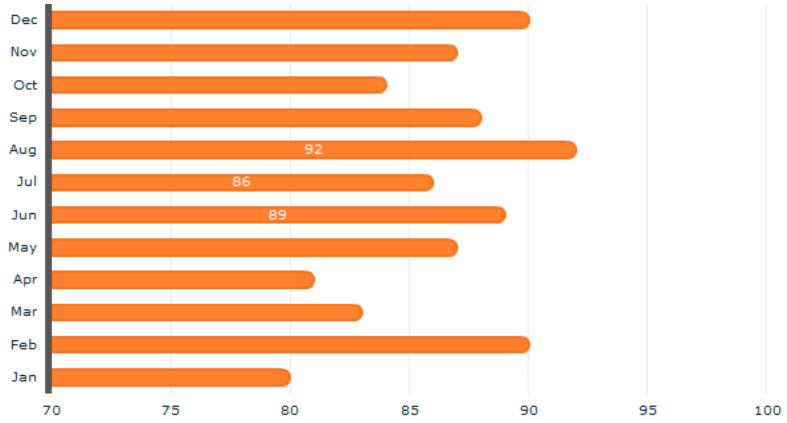
바 차트는 칼럼차트와 비슷합니다. 다만 수치 데이터를 X 축(수평축)에 할당해야 합니다.
반드시 verticalAxis(수직축)을 정의 하여 그룹핑에 해당되는 데이터를 수직축 필드로 입력하고, 수치데이터에 해당되는 값은 수평축 필드에 할당합니다.
바 2D 차트는 <Bar2DChart> 노드로 시작과 끝을 나타냅니다. Bar 2D 차트의 itemRenderer 속성값으로 2 개의 유효값이 존재합니다.

- SemiCircleBarItemRenderer
- BoxItemRenderer

그 외 속성과 속성값 "6.3 칼럼 2D 차트" 를 참고하십시오.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
	<Options>
		<Caption text="Anual Report"/>
	</Options>
	<Bar2DChart showDataTips="true">
		<verticalAxis>
			<CategoryAxis categoryField="Month"/>
		</verticalAxis>
		<series>
			<Bar2DSeries xField="Profit" itemRenderer="SemiCircleBarItemRenderer">
				<showDataEffect>
					<SeriesInterpolate/>
				</showDataEffect>
			</Bar2DSeries>
		</series>
	</Bar2DChart>
</rMateChart>
```

<예제 26 바차트 예제>



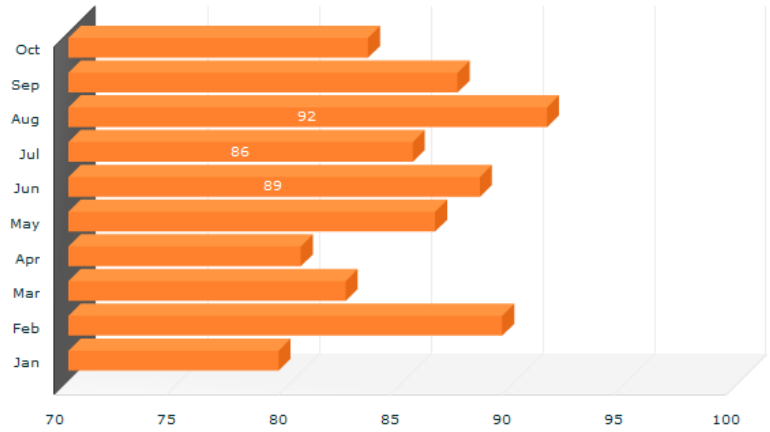
<그림 12 바차트 출력 결과>

6.7. 바 3D 차트.

바 3D 차트는 <Bar3DChart> 노드로 시작과 끝을 나타냅니다.
속성 및 유효값은 <표 13 칼럼 3D 차트의 속성 설명> 과 <표 14 칼럼 3D 차트 시리즈의 속성 설명> 를 참고하십시오.

```
<rMateChart>
    <Options>
        <Caption text="Annual Report"/>
    </Options>
    <Bar3DChart showDataTips="true">
        <verticalAxis>      //수직축을 설정하고 CateogryAxis 로 그룹핑에 해당되는 Month정의
            <CategoryAxis categoryField="Month" />
        </verticalAxis>
        <series>
            <Bar3DSeries xField="Profit">   //xField에 수치데이터 Profit
                <showDataEffect>
                    <SeriesInterpolate />
                </showDataEffect>
            </Bar3DSeries>
        </series>
    </Bar3DChart>
</rMateChart>
```

<예제 27 바 3D 차트 예제>



<그림 13 바 3D 차트 출력 결과>

6.8. 파이, 도넛차트,

2D 는 <Pie2DChart> 노드로 시작과 끝을 나타내고, 3D 는 <Pie3DChart>노드로 시작과 끝을 나타냅니다. 2D 와 3D 차트 각각의 시리즈는 <Pie2DSeries/> 와 <Pie3DSeries/> 노드로 데이터를 표현합니다.

파이차트와 도넛차트의 레이아웃은 단 한가지를 제외하고 모두 같습니다.
아래 레이아웃의 innerRadius 프로퍼티 값에 의해 파이차트와 도넛차트를 표현합니다. innerRadius 의 범위는 0 ~ 1 까지이며 0 값이 파이차트이며 1 로 가까이 갈수록 도넛차트의 가운데 공간이 커집니다.

속성 명	유효 값	설 명	비 고
innerRadius	0 ~ 1 (기본값:0)	3D 파이차트 중앙의 hole 크기	2D & 3D 공용
depth	Number (기본값:20)	원통의 깊이(높이) 입니다.	3D 전용
elevation	-90 ~ 90 (기본값:70)	원통의 기울기 각도입니다.	3D 전용
itemClickJsFunction	자바스크립트 함수	파이 조각 클릭 시 호출할 자바스크립트 함수를 설정합니다.	2D & 3D 공용
explodable	Boolean(기본값:true)	파이 차트 아이템을 클릭 시 삐져나오게 할지 여부를 나타냅니다.	2D & 3D 공용
explodingAlone	Boolean(기본값:true)	explodable 을 true 로 설정한 경우 삐져나오는 아이템이 항상 1개로 설정할지 여부를 나타냅니다.	2D & 3D 공용

<표 16 파이차트 속성 설명표>

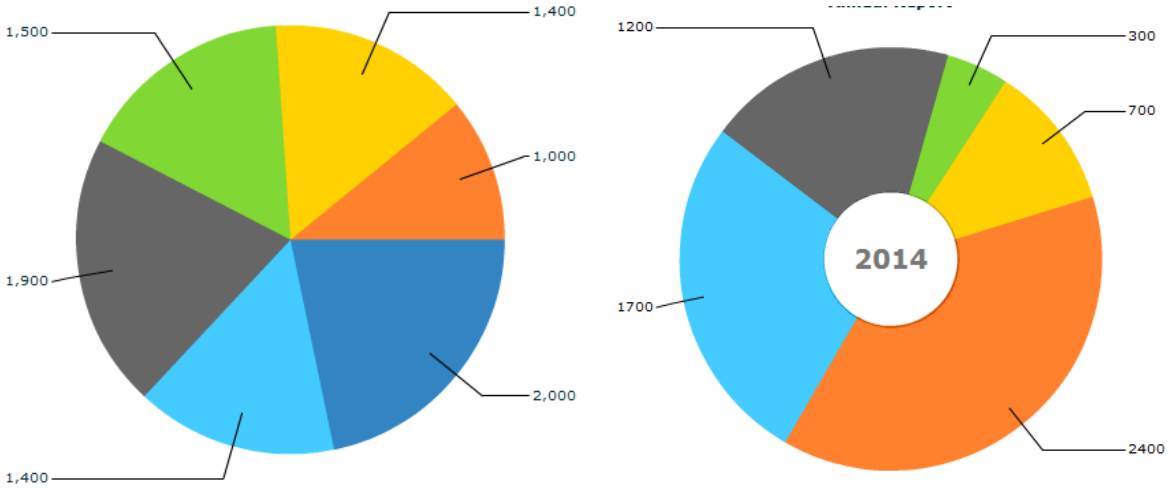
속성 명	유효 값	설명
field	차트 데이터의 특정 필드	파이 시리즈가 표현할 차트데이터의 필드를 지정합니다.
nameField	차트 데이터의 특정 필드	파이 시리즈의 이름에 해당되는 차트데이터의 필드를 지정합니다.
showDataEffect	SeriesZoom, SeriesSlide, SeriesInterpolate	차트 데이터가 표현될 때 나타나는 효과를 정의합니다.
fills	SolidColor 들의 배열	파이조각들의 컬러를 결정합니다.
startAngle	시작 각도(0 – 360)	파이 시리즈를 그리기 시작하는 각도를 지정합니다.
stroke	Stroke	시리즈의 선을 나타냅니다.
labelPosition	none | inside | outside | callout | insideWithCallout	수치 필드의 위치를 결정합니다.
labelJsFunction	자바스크립트 함수명	수치필드 표시 텍스트 사용자 정의 함수입니다.
fillJsfunction	자바스크립트 함수명	칼럼 시리즈 아이템의 채우기 색 사용자 정의 함수입니다.

<표 17 파이시리즈 속성 설명표>

```
<rMateChart>
  <Pie2DChart innerRadius="0" showDataTips="true">
    <series>
        <Pie2DSeries
            nameField="Month" //툴팁에 표시할 필드 이름. <그림 14 파이차트와 도넛차트 출력 결과> 툴팁의
November에 해당
            field="Profit" // 파이차트가 참고할 데이터 영역.
            depth="0.05" //2D 차트의 그라데이션 깊이(범위 0 ~ 0.2)
             labelPosition="callout">

            <showDataEffect>  <!—데이터 출력시 SeriesInterpolate 효과 표시→
                <SeriesInterpolate/>
            </showDataEffect>

            <fills> <!—시리즈에 2가지 색을 지정하여 2가지색이 반복되고 있습니다. →
                <SolidColor color="0xff0000"/>
                <SolidColor color="0xffffff"/>
            </fills>
        </Pie2DSeries>
    </series>
  </Pie2DChart>
```

<rMateChart>

<예제 28 파이차트 예제>



<innerRadius = 0 인 경우> <innerRadius = 0.33 인 경우>

<그림 14 파이차트와 도넛차트 출력 결과>

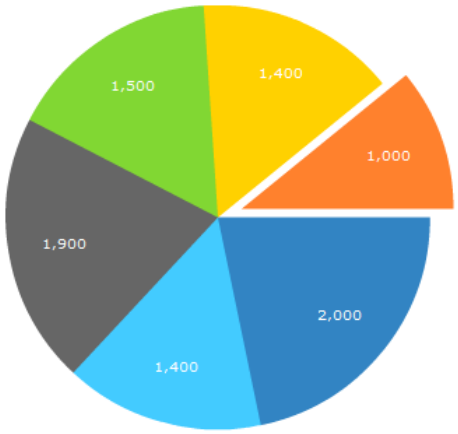
위와 같이 파이차트와 도넛차트는 innerRadius 프로퍼티 값에 따라 달라집니다.

다음으로 labelPosition 에 대한 설명입니다.
labelPosition 은 차트 데이터의 값을 따로 표시할 때 그 위치를 정의합니다. 할당 가능한 값과 설명은 아래와 같습니다.

- none : 데이터 값을 표시하지 않습니다.
- inside : 차트 안에 표시합니다.
- outside : 차트 밖에 표시합니다.
- callout : 위 예제와 같이 따로 선을 긋고 표시합니다.
- insideWithCallout : 기본적으로 안쪽에 표시하나 크기가 작아 충분한 공간이 없을 때 따로 선을 긋고 callout 형태로 표시합니다.

파이, 도넛차트의 데이터영역을 클릭한 경우 해당 영역을 다음과 같이 데이터 영역을 빠져 나오게 할 수 있습니다.(explodable 은 true 가 기본값입니다.)

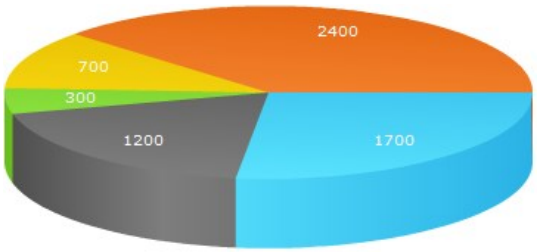
```
<rMateChart>
        <Options>
                <Caption text="Annual Report"/>
                <SubCaption text="2008"/>
        </Options>
        <Pie2DChart        explodable="true"        innerRadius="0"
showDataTips="true">
        ......
<rMateChart>
```



<예제 29 파이차트 해당 영역 클릭한 경우 예제와 출력모습>

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Annual Report"/>
                <SubCaption text="2008"/>
        </Options>
        <Pie3DChart showDataTips="true" explodable="false">
                <series>
                        <Pie3DSeries nameField="Month" field="Profit" labelPosition="inside">
                                <showDataEffect>
                                        <SeriesZoom/>
                                </showDataEffect>
                        </Pie3DSeries>
                </series>
        </Pie3DChart>
</rMateChart>
```

<예제 30 파이 3D 차트 예제>

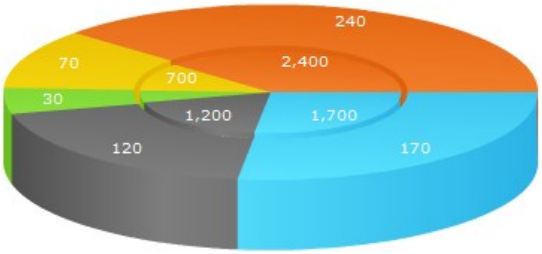


<그림 15 파이 3D 차트 출력 결과>

파이 Stacked 3D 차트는 시리즈 2 개로 표현이 가능합니다. 아래는 그 예제와 실행화면입니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
    <Options>
        <Caption text="Annual Report" />
        <SubCaption text="2008" />
    </Options>
    <Pie3DChart showDataTips="true" explodable="false">
        <series>
            <Pie3DSeries nameField="Month" field="Profit" displayName="Profit" labelPosition="inside">
                <showDataEffect>
                    <SeriesZoom />
                </showDataEffect>
            </Pie3DSeries>
            <Pie3DSeries nameField="Month" field="Cost" displayName="Cost" labelPosition="inside">
                <showDataEffect>
                    <SeriesZoom />
                </showDataEffect>
            </Pie3DSeries>
        </series>
    </Pie3DChart>
</rMateChart>
```

<예제 31 파이 Stacked 3D 차트 예>



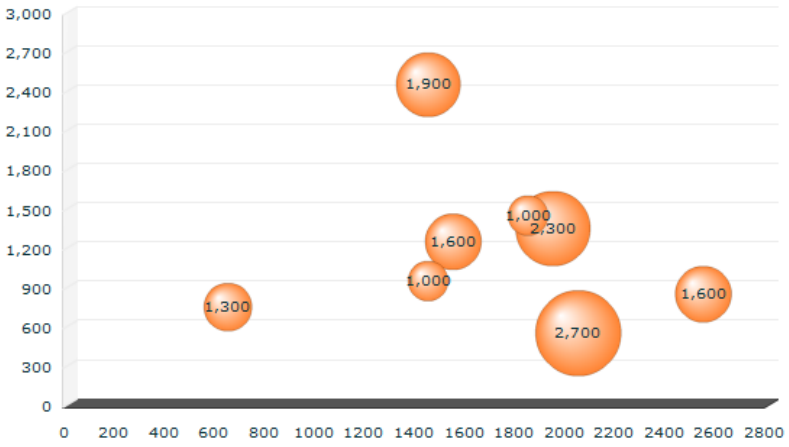
<그림 16 파이 3D Stacked 차트 출력 결과>

6.9. 버블 3D 차트.

버블 3D 차트는 <Bubble3DChart> 노드로 시작과 끝을 나타내고 데이터 표현이라 할 수 있는 시리즈는 <Bubble3DSeries> 노드로 표현합니다. 버블 3D 차트는 수치데이터를 X, Y, Radius 3 개의 필드에 표현합니다. X 는 X 축에 대응하는 값, Y 는 Y 축, 그리고 Radius 는 버블의 크기를 결정하는 값입니다.
실질적으로 데이터를 표현하는 것은 시리즈이기 때문에 <Bubble3DSeries> 노드의 속성으로 xField, yField, RadiusField 에 각각의 수치데이터를 입력하십시오. 다음은 예제와 실행화면입니다.

```
<rMateChart>
  <Bubble3DChart showDataTips="true">
    <series>
        <Bubble3DSeries xField="Profit" yField="Cost" radiusField="Revenue">

            <showDataEffect>
                <SeriesInterpolate />
            </showDataEffect>
            <fills>  <!—6가지 색을 지정하여 6가지색이 반복되고 있습니다. →
                <SolidColor color="0xFF0000"/>
                <SolidColor color="0x00FF00"/>
                <SolidColor color="0x0000FF"/>
                <SolidColor color="0xFF00FF"/>
                <SolidColor color="0xFFFF00"/>
                <SolidColor color="0xFFFFFF"/>
            </fills>
        </Bubble3DSeries>
</series></Bubble3DChart></rMateChart>
```

<예제 32 버블 3D 차트 예제>

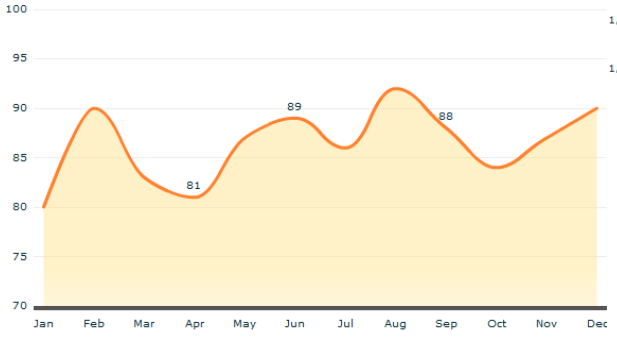


<그림 17 버블 3D 차트 출력 결과>

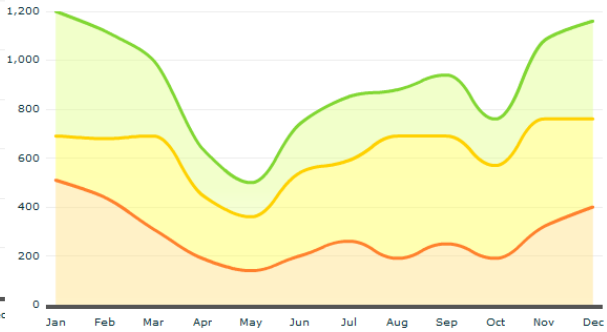
6.10. 영역(Area) 차트.

영역차트의 type 에는 overlaid, stacked, 100% 3 가지가 있습니다. 이에 대한 기능은 칼럼차트와 같고 영역차트 type 의 기본값은 overlaid 입니다

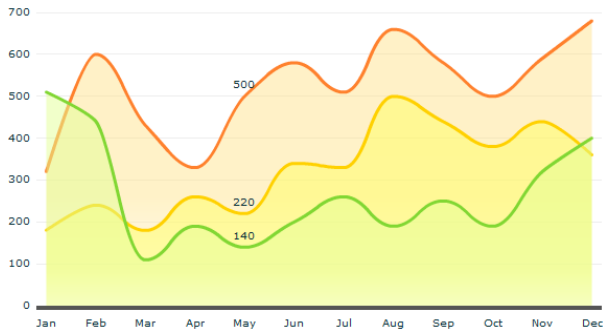
Area2DSeries 프로퍼티 중 form 은 데이터를 어떻게 표현할지를 명시합니다. 아래 그림을 참고하십시오.



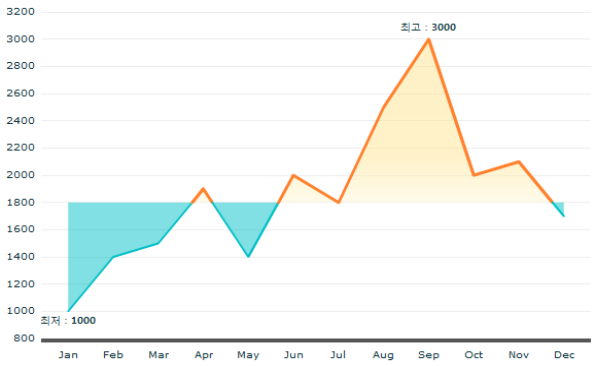
<form="segment" 기본값>



<form="curve">



<form="step">



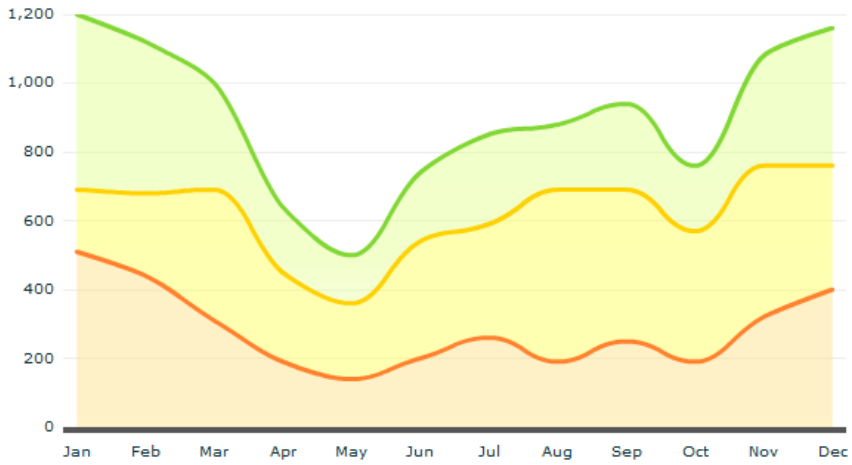
<form="reverseStep">

<그림 18 영역차트 form 별 출력 결과>

```
<rMateChart>
    <Area2DChart showDataTips="true" type="stacked">
        <horizontalAxis>
            <CategoryAxis categoryField="Month" />
        </horizontalAxis>
        <series>
            <Area2DSeries yField="Profit" form="curve" displayName="Profit">
                <showDataEffect>
                    <SeriesInterpolate />
                </showDataEffect>
            </Area2DSeries>
            <Area2DSeries yField="Cost" form="curve" displayName="Cost">
                <showDataEffect>
                    <SeriesInterpolate />
```

```
                    </showDataEffect>
                </Area2DSeries>
                <Area2DSeries yField="Revenue" form="curve" displayName="Revenue">
                    <showDataEffect>
                        <SeriesInterpolate />
                    </showDataEffect>
                </Area2DSeries>
            </series>
    </Area2DChart>
</rMateChart>
```

<예제 33 영역차트 예제>



<그림 19 영역차트 출력 결과>

6.11. 플롯차트

플롯차트는 수평 축 상의 위치를 나타내는 값과 수직 축 상의 위치를 나타내는 값을 가지는 데이터 포인트를 직교좌표로 나타내는 경우에 사용합니다.
플롯차트를 작성하려면 <Plot2DChart>로 시작하여 </Plot2DChart>로 끝납니다. 시리즈는 <Plot2DSeries>입니다.

속성 명	설 명
yField	각 데이터 포인트의 수직 축 위치를 결정하는 데이터 의 필드를 지정합니다.
xField	각 데이터 포인트의 수평 축 위치를 결정하는 데이터 의 필드를 지정합니다.
radius	각 데이터 포인트에 표시하는 심볼의 반경을 픽셀 단위로 지정합니다. 기본값은 5 픽셀입니다.

<표 18 Plot2DSeries 속성 설명>

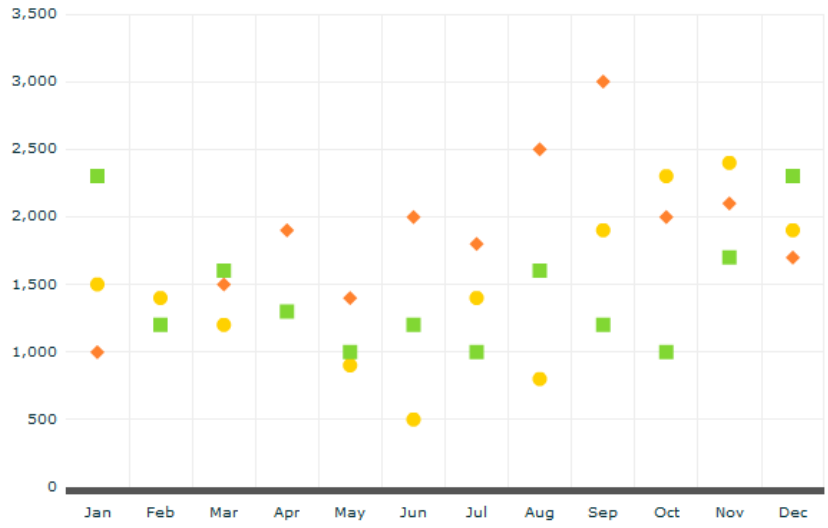
주의: Plot2DChart 에서는, 각각의 Plot2DSeries 에 xField 프로퍼티와 yField 프로퍼티의 양쪽 모두를 지정

하셔야합니다.

예제를 보도록 하겠습니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
	<Options>
		<Legend useVisibleCheck="true" />
	</Options>
	<Plot2DChart showDataTips="true"> //플롯차트 생성
		<verticalAxis>
			<LinearAxis maximum="3500" />
		</verticalAxis>
		<horizontalAxis>
			<LinearAxis maximum="2800" />
		</horizontalAxis>
		<series>
			<Plot2DSeries xField="Cost" yField="Profit" radius="5" displayName="Cost/Profit">
			</PlotSeries>
			<Plot2DSeries xField="Revenue" yField="Profit" radius="5" displayName="Revenue/Profit">
			</Plot2DSeries>
			<Plot2DSeries xField="Cost" yField="Revenue" radius="5" displayName="Cost/Revenue">
			</Plot2DSeries>
		</series>
</Plot2DChart>
</rMateChart>
```

<예제 34 플롯차트 예제>



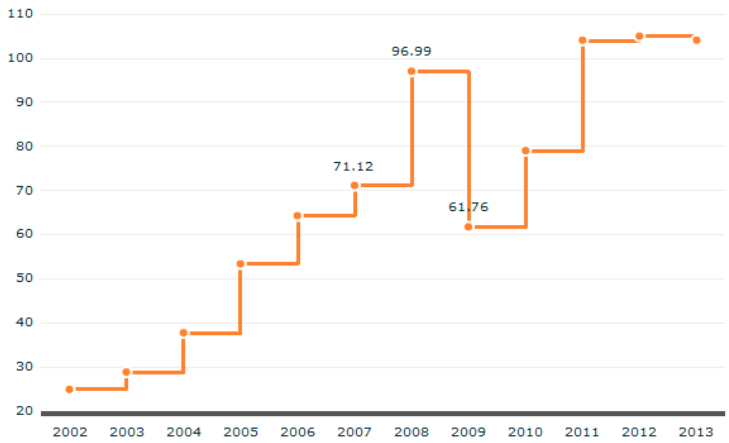
<그림 20 플롯차트 출력 결과>

6.12. 라인차트.

라인차트는 다른 Cartesian 차트와 비교하여 type 속성이 없습니다. 기본적으로 overlaid 속성입니다. 그리고 라인차트의 시리즈인 Line2DSeries 의 속성으로 form 이 존재합니다. 라인시리즈의 form 속성은 영역차트의 시리즈인 Area2DSeries 의 form 속성과 같이 segment, curve, step, reverseStep 이 존재합니다. 위 Area2DSeries 의 form 속성을 표현한 그림을 참고하십시오.

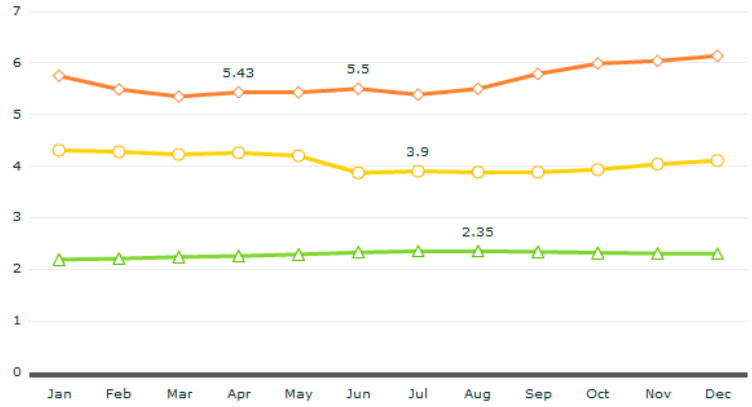
```
<rMateChart cornerRadius="12" borderStyle="solid">
	<Options>
		<Caption text="Anual Report" />
	</Options>
	<Line2DChart showDataTips="true">
		<horizontalAxis>
			<CategoryAxis categoryField="Month" />
		</horizontalAxis>
		<series>
			<Line2DSeries yField="Profit" form="step">
				<showDataEffect>
					<SeriesInterpolate />
				</showDataEffect>
			</Line2DSeries>
			<Line2DSeries yField="Cost" form="step">
			</Line2DSeries>
			<Line2DSeries yField="Revenue" form="step">
			</Line2DSeries>
		</series>
	</Line2DChart>
</rMateChart>
```

<예제 35 라인차트 예제>



<그림 21 라인차트 출력 결과>

라인차트에서 X 축과 Y 축이 만나는 지점에 itemRenderer 를 적용하여 특수하게 출력할 수 있습니다. 출력결과와 예제는 아래와 같습니다.



<그림 22 라인차트에 itemRenderer 적용한 경우 출력 결과>

```
<rMateChart cornerRadius="12" borderStyle="solid" paddingTop="10" paddingBottom="20" paddingRight="20"
paddingLeft="20">
        <Options>
            <Caption text="Anual Report" />
        </Options>
<Line2DChart showDataTips="true">
        <horizontalAxis>
        <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series>
        1 <Line2DSeries yField="Profit"  radius="10" fill="0xFF0000"  itemRenderer="
DiamondItemRenderer"/>

        2 <Line2DSeries yField="Cost"  radius="10" fill="0x00FF00" itemRenderer="CircleItemRenderer"/>

        3 <Line2DSeries yField="Revenue"  radius="10" fill="0xFFFF00"
itemRenderer="TriangleItemRenderer"/>
    </series>
</Line2DChart>
</rMateChart>
```

<예제 36 라인차트에 itemRenderer 적용한 예제>

Line2DSeries 에 itemRenderer 를 적용할 때 따르는 주요 속성은 아래와 같습니다.

- radius : X 축과 Y 축이 만나는 지점에 도형의 크기를 결정합니다.
- fill : 도형(itemRenderer)의 안쪽 컬러를 지정합니다.

- stroke : 도형의 테두리 선을 지정합니다.

라인차트 전체 라인의 색과 두께 조정은 lineStroke 속성을 이용합니다. 이에 대한 설명은 <예제 66 라인차트의 선(stroke)를 변경한 경우> 를 참고하십시오.

라인차트의 itemRenderer 속성값으로 다음 유효값이 존재합니다.

- DiamondItemRenderer – 위 예제 1 번에 해당.
- CircleItemRenderer – 위 예제 2 번에 해당.
- TriangleItemRenderer – 위 예제 3 번에 해당.
- CrossItemRenderer - 십자가 모양
- XShapeItemRenderer - X 자 모양
- IShapeItemRenderer – I 자 모양
- RectangleItemRenderer – 네모 모양

6.13. 점선(Dashed-Line) 차트

점선 차트(Dashed-Line Chart) 는 라인차트의 모든 기능을 포함하고, 레이아웃 작성법 또한 같습니다. 라인차트와 비교하여 점선 차트의 고유 속성은 다음과 같습니다.

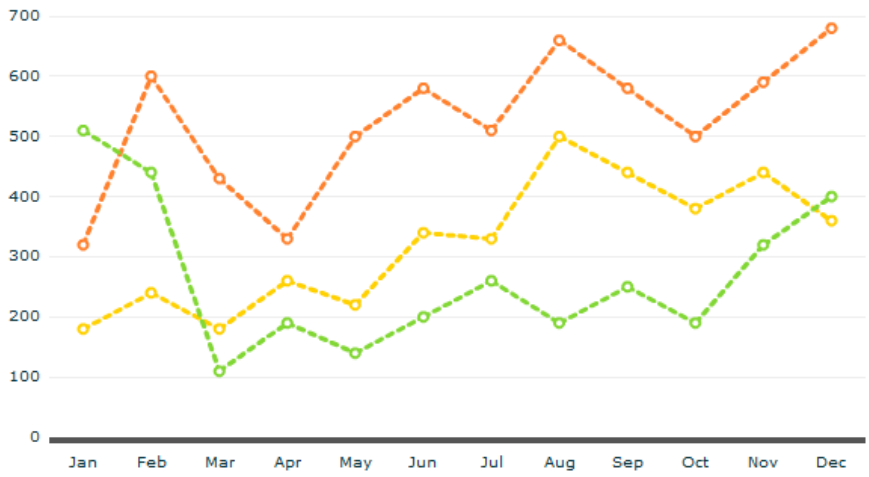
속성 명	유효 값	설 명
lineStyle	“normal”, “dashLine”	라인차트에서 라인의 type 을 나타냅니다. 점선차트를 생성하기 위해서는 dashLine 으로 값을 할당하십시오.
dashLinePlacement,	“before”, “after”	dashLineSeperatePos 을 설정한 경우 점선을 dashLineSeperatePos 기준으로 전에 나타낼지 후에 나타낼지를 결정합니다.
dashLineSeperatePos	데이터의 인덱스 (즉, uint)	점선과 실선을 같이 사용하고자 할 때의 경계점을 나타냅니다. 이 점은 데이터의 인덱스 입니다.
dashLinePattern	Number(픽셀단위) (기본값 : 10)	표현할 점선의 길이를 지정합니다. 예를 들어 5 를 지정 시 5픽셀에 해당되는 선을 그린 후 5픽셀에 해당되는 공백을 생성합니다. 이 패턴을 반복합니다. 길이를 크게 할 수록 성능은 향상됩니다. 차트 렌더링이 느리다면 패턴 길이를 길게 잡으십시오.

<표 19 점선 차트 속성 유효값 및 설명>

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFFF"  cornerRadius="12" borderStyle="solid">
```

```
        <Line2DChart showDataTips="true" >
                <horizontalAxis>
                        <CategoryAxis categoryField="Month" padding="0.5" />
                </horizontalAxis>
                <series>
                        <Line2DSeries labelPosition="up"
                yField="Profit" radius="4
                lineStyle="dashLine"          // 점선 차트로 표현
                dashLineSeperatePos="5"    // 인덱스5(즉, 6번째 데이터) 에서 점실선 나뉨
                dashLinePlacement="before" // 점선은 앞에 나타남
                itemRenderer=" RectangleItemRenderer">
                        </Line2DSeries>
                </series>
        </Line2DChart>
</rMateChart>
```

<표 20 점선 차트 예제>



<그림 23 점선 차트 출력 모습>

6.14. 콤비네이션 차트.

콤비네이션 차트는 서로 다른 유형의 차트가 함께 있는 것을 말합니다. 아래 예제는 칼럼 차트와 라인차트가 함께 데이터를 표현하는 콤비네이션 차트입니다.

- 칼럼차트라면 series 프로퍼티에 2D 인 경우 Column2DSeries, 3D 인 경우 Column3DSeries 로 데이터를 표현하였지만 콤비네이션 차트에서는 칼럼차트와 라인차트 두 차트의 시리즈(데이터 표현)가 존재하므로 칼럼시리즈를 묶을 수 있는 Column2DSet 또는 Column3DSet 을 정의하십시오.

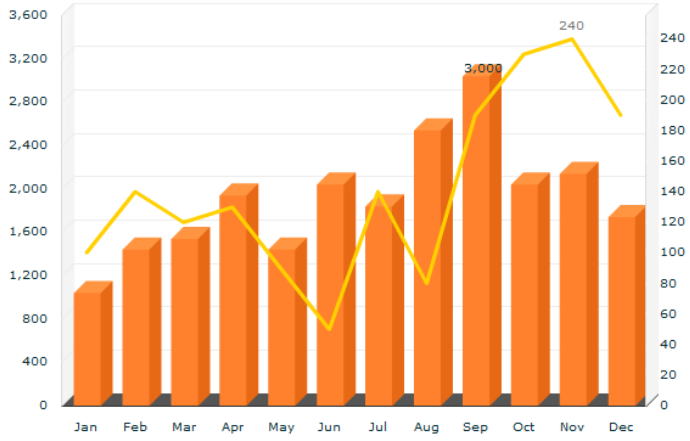
- 2D 를 표현하기 위해 Column2DSet 을 정의 한 후 그 자식으로 Column2DSeries 를 원하는 데이터 표현 만큼 정의하십시오.
- Column2DSet(또는 Column3DSet) 의 type 은 일반 칼럼차트의 type 과 동일합니다. <그림 9 칼럼차트 Type 별 출력 결과> 참고. **(그러나 기본값은 overlaid)**
- Column2DSet 과 형제인 위치에 Line2DSeries 를 정의하십시오.

위의 설명을 구체화한 코드입니다.

```
<rMateChart>
    <Combination2DChart showDataTips="true">
        <horizontalAxis>
            <CategoryAxis categoryField="Month"/>
        </horizontalAxis>

        <series>
            <Column2DSet type="clustered">
                <series>
                    <Column2DSeries yField="Profit" displayName="Profit"/>
                    <Column2DSeries yField="Revenue" displayName="Revenue"/>
                </series>
            </Column2DSet>
            <Line2DSeries  yField="Cost" displayName="Cost">
                <filters> //라인시리즈에 그림자를 넣는 효과입니다.
                    <DropShadowFilter distance="3" color="0x666666" alpha=".8"/>
                </filters>
                <lineStroke> //라인시리즈의 라인의 두께, 색을 지정합니다.
                    <Stroke color="0xFFCC00" weight="4"/>
                </lineStroke>
            </Line2DSeries>
        </series>
    </Combination2DChart>
</rMateChart>
```

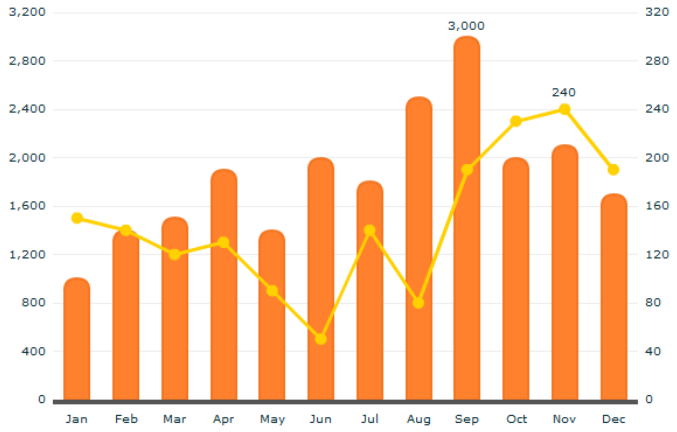
<예제 37 콤비네이션 차트 예제>

다음은 위의 예제 출력화면입니다.



<그림 24 Column2DSet type="clustered" 설정한 콤비네이션 차트 결과>

칼럼차트 type 을 스택으로 표현하는 콤비네이션을 원하다면 <ColumnSet type="stacked">를 정의하십시오.



<그림 25 Colum2DnSet type="stacked" 설정한 콤비네이션 차트 결과>

6.15. 실시간 차트

실시간 차트는 주기적으로 변하는 데이터를 그대로 표현합니다. 차트는 풀링 방식으로 데이터를 얻도록 설계되었습니다.

실시간 차트는 <RealTimeChart/> 노드로 시작과 끝을 지시합니다. 실시간 차트의 시리즈(series) 속성으로는 칼럼시리즈, 라인시리즈, Area 시리즈 등을 갖습니다. 그리고 서버에서 생성되는 데이터를 갖고 오기 위해 주기적으로 원격 호출(RPC, Remote Procedure Call)이 이루어 집니다. 그 역할을 담당하는 것이 HttpServiceRepeater 입니다. 실시간으로 데이터를 표현하기 위해서 반드시 레이아웃에 정의해야 합니다.

실시간 차트 레이아웃을 작성할 때 주의점은 아래와 같습니다.

첫째, HttpServiceRepeater 는 반드시 <rMateChart/> 노드의 자식 위치에, 그리고 <RealTimeCartesianChart/>의 아래 형제 위치에 놓여야 합니다.
둘째, target 속성의 속성값으로 차트(즉, 실시간차트)의 id 값을 지시하십시오.

셋째, HttpServiceRepeater 의 url 은 차트 데이터가 있는 XML URL 경로입니다. 이 XML 데이터 형식은 반드시 아래와 같은 구조로 작성되어야 합니다.

```
<?xml version="1.0" encoding="utf-8" ?>
<items>                          //XML의 루트는 반드시 items 로 시작
    <item>                       // 한 개의 데이터 셋은 반드시 item 으로 묶여야 합니다.
        <Time>13:8:27</Time> // 이곳은 사용자 정의하십시오.
        <Volume>5527</Volume>
        <Price>309</Price>
    </item>
</items>
```

<예제 38 실시간 차트 데이터 양식>

실시간 차트(rMateRealtimeChart.swf)를 HTML 문서에 임베딩 시 차트 초기의 데이터를 삽입할 필요는 없습니다. 그래서 아래와 같이 초기 데이터를 삽입하는 과정을 삭제하거나 주석처리 하십시오.

```
<html>
<head>
<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>
<script src="rMateChartLicense.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ---------- flashVars 설정 시작 ----------

// 차트 레이아웃 URL 경로를 설정하십시오.
var layoutURL = encodeURIComponent("RealTme_Layout.xml");
var flashVars = "layoutURL="+layoutURL;


// 차트 데이타 URL 경로 설정하는 부분을 삭제하거나 주석처리하십시오.

<!-- 사용자 정의 설정 끝  -->
</script>
</head>

<body onload="rMateChartInit()"> //동기화를 위한 함수입니다.
<table>
    <tr>
```

```
        <td>
            <script>
            <!—
             // 차트를 생성하고자 하는 위치에서 다음과 같이 함수 호출하십시오.
             // 파라메터 설명(순서대로)
             // 1. ID (임의로 정의하십시오)
             // 2. swf 파일 URL(확장자는 생략)
             // 3. URL 경로 등을 정의한 변수명.
             // 4. 차트 가로 사이즈.
             // 5. 차트 세로 사이즈.
             // 6. 차트 배경색.

                    rMateChartCreate("chart1"," rMateRealtimeChart",flashVars, 500, 500, "#FFFFFF");
            // -->
            </script>
        </td>
    </tr>
</table>
</body>
</html>
```

<예제 39 실시간 차트 HTML 문서 작성법>

***참고 : 예를 들어 차트가 데이터를 갱신하는 주기가 5 초라고 한다면, 차트는 매 5 초마다 서버에서 풀링 방식으로 데이터를 가져와 표현합니다. 그러나 실제적으로 여러 가지 요인(선로 상태, 통신 상태, 트래픽 수 등등)에 따라 서버와의 통신에서 오차는 발생할 수 있습니다. 서버사이드는 이 오차를 고려하여 차트가 가져갈 데이터를 갱신 할 필요가 있습니다.**

속성 명	유효 값	설 명
dataDisplayType	dataSize | time (기본값 : dataSize)	차트 표현 유형을 나타냅니다.
displayDataSize	Number(기본값 : 20)	한 화면에 표현할 데이터의 양입니다.
interval	Number(단위:초)	차트 표현 유형이 time 인 경우 HttpServiceRepeater의 interval 값
timePeriod	Number(단위:초)	차트 유형이 time 인 경우 데이터를 차트에 나타낼 시간단위.

<표 21 RealTimeChart 속성 및 유효 값, 설명>

HttpServiceRepeater 속성 및 유효 값, 설명입니다.

속성 명	유효 값	설 명
url	String	데이터가 있는 URL 경로입니다.

interval	Number(단위:초) (기본값:30)	매초 단위로 RPC를 실행할 것인지를 나타냅니다. 예를 들어 60이라면 1분 단위로 RPC를 실행합니다.
target	차트의 id 값	RPC 성공 후 가져온 데이터를 넘겨줄 객체를 말합니다. 실시간 차트의 id값을 넣어주십시오.
method	get | post (기본값 :get)	HTTP 프로토콜 메소드입니다. Get방식인지 Post 방식인지를 결정.
useFaultAlert	true | false (기본값:true)	RPC 실패 시 실패했다는 경고창을 나타낼지를 나타내는 플래그입니다.

<표 22 HttpServiceRepeater 속성 및 유효 값, 설명>

실시간 차트는 두 가지 유형으로 데이터를 표현합니다.

1. 데이터 개수를 기준으로 표현합니다. 서버나 기타 경로를 통하여 데이터를 받으면 차트는 즉시 데이터를 표현합니다. displayDataSize 의 값을 20 으로 설정하였다면 한 화면에 20 개의 데이터를 표현한 후 다음 데이터부터 차례로 화면을 갱신하며 데이터를 출력합니다.

2. 시간을 기준으로 데이터를 표현합니다. 예를 들어 timePeriod 값을 3600 초(1 시간)으로 설정하고 interval 값을 3 으로 설정했다면 매 3 초마다 차트는 1 시간동안 데이터를 현재 화면에 출력하고 1 시간 이후의 다음 데이터부터 차례로 화면을 갱신하며 데이터를 출력합니다.(interval 값은 HttpServiceRepeater 의 interval 과 같아야 정상적입니다.)

6.15.1. 데이터 개수를 기준으로 실시간 차트 표현(CategoryAxis 활용)

현재 차트가 표현하는 데이터의 개수를 생각한다면 CategoryAxis 를 활용하여 차트 레이아웃을 작성하여야 합니다. CategoryAxis 에 대한 속성 설명은 <표 10 각각의 Axis 속성 명과 유효 값 설명> 를 참고하세요.
CategoryAxis 는 연속적이지 않고, 데이터의 카테고리 필드를 개별적으로 판단합니다. 예를 들어 데이터 필드 중 시간이나 날짜와 관련된 필드가 있을 때 이것은 형식상 시간, 날짜이지만 차트의 카테고리축은 그렇게 판단하지 않고 단순 문자열(String)로 판단합니다. 연속적이지 않기 때문에 중복데이터를 그대로 표현하는 단점이 있습니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
        <RealTimeChart id="chart" dataDisplayType="dataSize" displayDataSize="15" showDataTips="true">
                <horizontalAxis>
```

```
                    <CategoryAxis id="hAxis" categoryField="Time"/>
                </horizontalAxis>
                <series>
                    <Column2DSeries yField="Volume" displayName="Trading Volume"
itemRenderer="net.riamore.rmate.charts.renderers.GradientColumnItemRenderer">
                        <filters> //그림자 효과를 내는 필터 정의
                            <DropShadowFilter distance="3" color="0x666666" alpha=".8" />
                        </filters>
                        <fill> //채우기 색 정의
                            <SolidColor color="0xB0C759" />
                        </fill>
                    </Column2DSeries>
                </series>
        </RealTimeChart>
        <HttpServiceRepeater url="http://demo.riamore.net/chartTest/data.jsp" target="{chart}" interval="3"
method="get" /> //3초 단위로 풀링방식으로 데이터 가져옴
</rMateChart>
```

<예제 40 CategoryAxis 활용한 실시간 차트 예제>

6.15.2. 정해진 시간을 기준으로 실시간 차트 표현(DateTimeAxis 활용)

최근 데이터를 기준으로 정해진 시간 동안 들어온 데이터를 표현하고자 할 때 DateTimeAxis 를 활용하여 레이아웃을 작성하십시오. DateTimeAxis 는 연속적인 시간을 표현하기 때문에 반복적인 데이터는 무시합니다. 그래서 어떠한 오차로 인해 차트가 서버의 갱신된 데이터를 가져오지 못했을 경우 즉, 이전의 중복된 데이터를 가져왔을 경우 무시하게 됩니다. 또한 오차로 인해 다음 데이터를 얻지 못하고 다다음 데이터를 얻었을 경우 얻지 못한 데이터에 대해 그 시간 영역에 해당되는 공백을 남겨 표시합니다.
일반적인 실시간 차트는 DateTimeAxis 를 활용하십시오.

```
<rMateChart cornerRadius="12" borderStyle="solid">
        <RealTimeChart id="chart" dataDisplayType="time" timePeriod="60" interval="3" showDataTips="true">
//60초동안 3초 간격으로 데이터 표현(시간 기준)
            <horizontalAxis>
            //데이터와 축라벨은 모두 초단위, 데이터가 3초단위로 들어오므로 dataInterval=3, 축라벨 간격은
            9(초), GMT가 아닌 Local시간 표시 true
                <DateTimeAxis dataUnits="seconds" labelUnits="seconds" dataInterval="3" interval="9"
displayLocalTime="true"/>
            </horizontalAxis>
            <series> //DateTimeAxis 이므로 xField 정의 필수.
                <Column2DSeries xField="Time" yField="Volume" displayName="Trading Volume"
itemRenderer="net.riamore.rmate.charts.renderers.GradientColumnItemRenderer">
```

```
                                <filters>
                                        <DropShadowFilter distance="3" color="0x666666" alpha=".8" />
                                </filters>
                                <fill>
                                        <SolidColor color="0xB0C759" />
                                </fill>
                        </Column2DSeries>
                </series>
        </RealTimeChart>
        <HttpServiceRepeater url="http://demo.riamore.net/chartTest/data4.jsp" target="{chart}" interval="3"
method="get" />
</rMateChart>
```

<예제 41 DateTimeAxis 활용한 실시간 차트 예제>

***참고 : RealTimeChart 의 interval, DateTimeAxis 의 interval, HttpServiceRepeater 의 interval, 이 3 개의 interval 을 똑같이 맞춤으로써 원하는 모양의 차트가 나옵니다.**

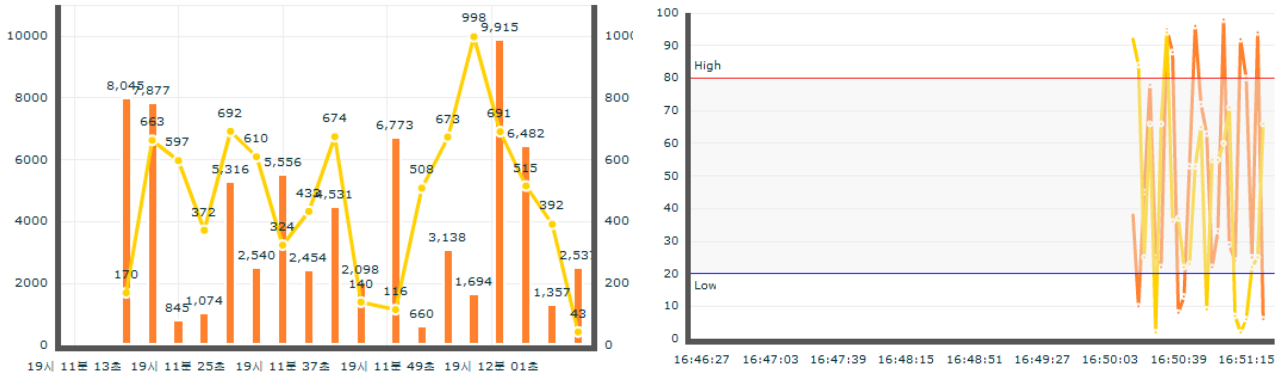


그림 26 Cateogory 축 실시간차트(좌), DateTime 축 실시간 차트(우)

6.15.3. HttpServiceRepeater 로 일반 차트의 데이터를 실시간으로 바꾸기.

HttpServiceRepeater 는 반드시 실시간 차트에만 국한적이지 않습니다.

예를 들어 칼럼 3D 차트를 생성하였습니다. 이 때 매 1 분 마다 칼럼 3D 차트의 전체 데이터를 갱신하고자 할 때 HttpServiceRepeater 는 유용합니다.

칼럼 3D 차트 레이아웃을 그대로 활용하고 아래와 같이 HttpServiceRepeater 를 레이아웃에 넣어주시면 됩니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Anual Report" />
```

```
        </Options>
        <Column3DChart id="chart" showDataTips="true">
                <horizontalAxis>
                        <CategoryAxis categoryField="Month" />
                </horizontalAxis>
                <series>
                        <Column3DSeries yField="Profit" displayName="Profit">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                        </Column3DSeries>
                </series>
        </Column3DChart>

        <HttpServiceRepeater url="http://demo.riamore.net/chartTest/singleData.jsp" target="{chart}"
        interval="60"/>
</rMateChart>
```

<예제 42 칼럼차트에 HttpServiceRepeater 적용>

6.16. 레이더(방사형) 차트

레이더(방사형) 차트는 <RadarChart/> 노드로 시작과 끝을 지시합니다. 레이더 차트의 데이터 표현을 나타내는 레이더 라인 즉, 차트 시리즈는 <RadarSeries> 입니다.
레이더 차트는 radialAxis 와 angularAxis 두 개의 축이 존재합니다. radialAxis 는 레이더 차트 원점에서 테두리까지를 잇는 축으로 각 데이터의 수치를 표현하는 역할을 합니다.
angularAxis 는 레이더 차트 테두리에 출력되는 축으로 레이더 차트 데이터의 카테고리에 해당되는 데이터를 표현합니다.

<RadarChart/> 의 속성 명 및 유효 값에 대한 설명은 아래와 같습니다.

속성 명	유효 값	설 명
type	polygon, circle	레이더 차트를 다각형 모양 또는 원 모양으로 표현합니다.
isSeriesOnAxis	true | false	차트 시리즈가 radialAxis 위에 표현될지 여부를 나타냅니다.
radialAxis	LinearAxis, LogAxis 객체	레이더 차트 원점에서 테두리까지의 반지름에 각 수치를 표시하는 축을 정의합니다.
angularAxis	CategoryAxis, LinearAxis, DateTimeAxis, LogAxis 객체	레이더 차트를 항목 데이터 개수만큼 쪼개 해당 항목을 출력하는 축입니다.

radialAxisRenderers	Axis2DRenderer 객체 배열	radialAxis의 렌더러를 정의합니다.
angularAxisRenderers	AngularAxisRenderers 객체 배열	angularAxis의 렌더러를 정의합니다.
series	레이더 차트의 시리즈 (즉, RadarSeries)	차트 시리즈를 정의합니다.
backgroundElements	RadarGridLines	레이더 차트 뒷 배경을 정의합니다.
showDataTips	true | false	마우스오버시 데이터팁(툴팁)의 유무를 나타냅니다.
paddingTop	Number	위쪽 여백
paddingBottom	Number	아래 여백
paddingLeft	Number	왼쪽 여백
paddingRight	Number	오른쪽 여백
startAnlge	Number	레이더시리즈의 첫번째 아이템을 렌더링하는 시작 각도를 지정합니다. 레이더시리즈의 가운데 점을 기준으로 오른쪽 가로방향(즉, 3시 방향)이 0 도로 기본값입니다.

<표 23 RadarChart 속성 설명표>

<RadarSeries/> 의 속성 명 및 유효 값에 대한 설명은 아래와 같습니다.

속성 명	유효 값	설 명
fillLineArea	true | false	레이더 차트 시리즈가 표현한 라인 안쪽에 채우기 색을 넣을지 여부를 정의합니다.
field	시리즈가 표현할 데이터의 필드	많은 데이터 중 현재 시리즈가 표현할 데이터의 필드입니다.
displayName	String	데이터팁(툴팁), 범례(Legend)에 표시될 시리즈의 이름입니다.
stroke	Stroke 객체	아이템렌더러의 테두리 그리기 선 스타일을 정의합니다.
lineStroke	Stroke 객체	레이더 차트 시리즈의 실제 라인 스타일을 정의합니다.
fill	SolidColor 객체	fillLineArea = 'true' 인 경우 라인 안쪽 공간에 채우기 색을 정의합니다.
radius	Number	아이템렌더러의 크기를 정의합니다.
itemRenderer	아이템렌더러(예:mx.charts.renderers.CircleItemRender)	데이터 포인트에 표시할 도형을 정의합니다.

<표 24 RadarSeries 속성 설명표>

연간 가계 지출 현황이라는 데이터를 갖고 레이더 차트로 표현하기 위해 일단 데이터를 정의해 보도록 하겠습니다.
다음은 일반적인 데이터를 갖고 차례로 XML, 배열 형태로 데이터를 바꾼 형태입니다.

1. 일반적인 데이터

	2005년	2006년	2007년	2008년
Food	100	100	180	150
Health Care	80	120	200	210
Transportation	70	115	100	240
Clothing	80	120	140	210
Education	90	160	130	200
Shelter	100	180	165	140
Leisure	76	120	130	170
Others	80	140	140	190

2. 일반적인 데이터를 배열 형태로 바꾼 결과.

```
var chartData = [{"catName":"Food", "year2005":100, "year2006":100, "year2007":180, "year2008":150}
                ,{"catName":"Health Care", "year2005":80, "year2006":120, "year2007":200, "year2008":210}
                ,{"catName":"Transportation", "year2005":70, "year2006":115, "year2007":100,
"year2008":240}
                ,{"catName":"Clothing", "year2005":80, "year2006":120, "year2007":140, "year2008":210}
                ,{"catName":"Education", "year2005":90, "year2006":160, "year2007":130, "year2008":200}
                ,{"catName":"Shelter", "year2005":100, "year2006":180, "year2007":165, "year2008":140}
                ,{"catName":"Leisure", "year2005":76, "year2006":120, "year2007":130, "year2008":170}
                ,{"catName":"Others", "year2005":80, "year2006":140, "year2007":140, "year2008":190}];
```

3. 일반적인 데이터를 XML 형태로 바꾼 결과.

```
<items>
        <item>
                <catName>Food</catName>
                <year2005>100</year2005>
                <year2006>100</year2006>
                <year2007>180</year2007>
                <year2008>150</year2008>
        </item>
        <item>
                <catName>Health Care</catName>
                <year2005>80</year2005>
                <year2006>120</year2006>
                <year2007>200</year2007>
                <year2008>210</year2008>
```

```
        </item>
          .
          .
          .
        <item>
                <catName>Others</catName>
                <year2005>80</year2005>
                <year2006>140</year2006>
                <year2007>140</year2007>
                <year2008>190</year2008>
        </item>
</items>
```

위 데이터를 바탕으로 작성한 레이더 차트 레이아웃 예제입니다.

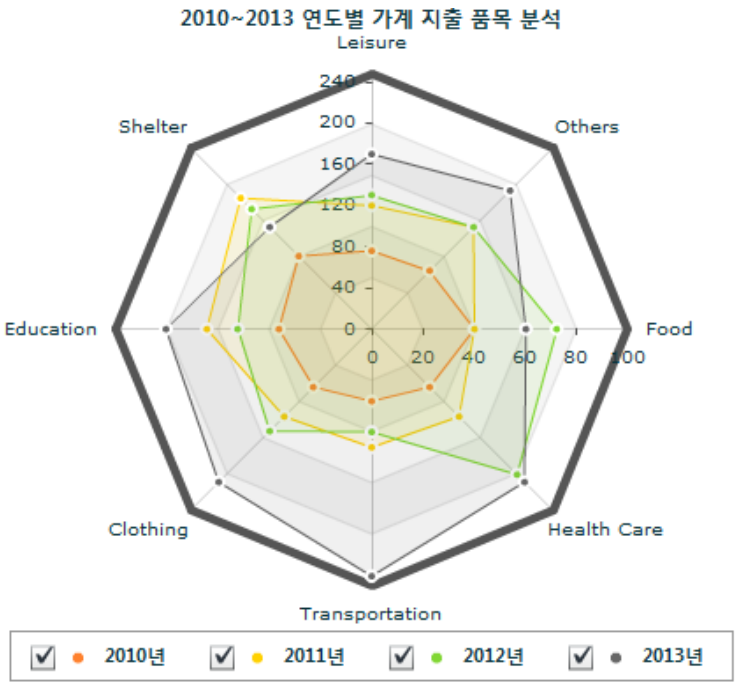
```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="연도별 가계 지출 품목 분석"/>
                <SubCaption text="(2005~2008)"/>
                <Legend useVisibleCheck="true"/>
        </Options>
        <RadarChart id="chart1"
                isSeriesOnAxis="false"
                type="polygon"
                showAllDataTips="false"
                showDataTips="true">
                <radialAxisRenderers>
                        <Axis2DRenderer horizontal="true" visible="true" styleName="hangingCategoryAxis" >
                        </Axis2DRenderer>
                        <Axis2DRenderer horizontal="false" visible="false" styleName="hangingCategoryAxis" >
                        </Axis2DRenderer>
                </radialAxisRenderers>
                <radialAxis>
                        <LinearAxis/>
                </radialAxis>
                <angularAxis>
                                        <CategoryAxis id="aAxis" categoryField="catName"
displayName="Category" />
                                </angularAxis>
                <angularAxisRenderers>
                        <AngularAxisRenderer axis="{aAxis}" axisTitleStyleName="axisTitle"
styleName="hangingCategoryAxis"/>
                </angularAxisRenderers>
              <series>
```

```
<RadarSeries field="year2005"
        displayName="2005년">
        <showDataEffect>
            <SeriesInterpolate/>
        </showDataEffect>
        <stroke> <!-- 데이터 포인트에 찍힌 도형의 외곽 테두리 선 스타일  -->
<Stroke color="0xE48701" weight="2"/>
        </stroke>
        <lineStroke> <!-- 레이더 라인 스타일 -->
            <Stroke color="0xE48701" weight="2"/>
        </lineStroke>
        <areaFill> <!--  레이더 안쪽 영역 채우기 색  -->
            <SolidColor color="0xE48701" alpha="0.1"/>
        </areaFill>
        <fill> <!--  데이터 포인트에 찍힌 도형의 안쪽 채우기 색 -->
            <SolidColor color="0xFFFFFF"/>
        </fill>
</RadarSeries>

<RadarSeries field="year2006"
        displayName="2006년">
        <showDataEffect>
            <SeriesInterpolate/>
        </showDataEffect>
        <stroke>
            <Stroke color="0xB0C759" weight="2"/>
        </stroke>
        <lineStroke>
            <Stroke color="0xB0C759" weight="2"/>
        </lineStroke>
        <areaFill>
            <SolidColor color="0xB0C759" alpha="0.1"/>
        </areaFill>
        <fill>
            <SolidColor color="0xFFFFFF"/>
        </fill>
</RadarSeries>
<RadarSeries field="year2007"
        displayName="2007년">
        <showDataEffect>
            <SeriesInterpolate/>
        </showDataEffect>
        <stroke>
            <Stroke color="0x0A80C4" weight="2"/>
```

```
                                </stroke>
                                <lineStroke>
                                        <Stroke color="0x0A80C4" weight="2"/>
                                </lineStroke>
                                <areaFill>
                                        <SolidColor color="0x0A80C4" alpha="0.1"/>
                                </areaFill>
                                <fill>
                                        <SolidColor color="0xFFFFFF"/>
                                </fill>
                        </RadarSeries>
                        <RadarSeries field="year2008"
                                displayName="2008년">
                                <showDataEffect>
                                        <SeriesInterpolate/>
                                </showDataEffect>
                                <stroke>
                                        <Stroke color="0xCC99CC" weight="2"/>
                                </stroke>
                                <lineStroke>
                                        <Stroke color="0xCC99CC" weight="2"/>
                                </lineStroke>
                                <areaFill>
                                        <SolidColor color="0xCC99CC" alpha="0.1"/>
                                </areaFill>
                                <fill>
                                        <SolidColor color="0xFFFFFF"/>
                                </fill>
                        </RadarSeries>
                </series>
        </RadarChart>
</rMateChart>
```

<예제 43 레이더 차트 레이아웃>

정의한 데이터와 위 레이아웃으로 출력한 차트는 다음과 같습니다.



<그림 27 레이더 차트 출력 화면>

6.17. 목표 대비 실적 차트.

목표 대비 실적 차트는 목표치와 달성치 두 데이터를 표현하는 차트로 2D 리니어 유형과 3D 실린더 유형이 있습니다.
목표 대비 실적 차트는 2D 의 경우 <Combination2DChart/> 노드로, 3D 의 경우 <Combination3DChart/> 노드로, 차트의 시작과 끝을 지시합니다. 목표 대비 실적 차트 시리즈(series) 속성 값으로는 단 2 개의 시리즈만 넣을 수 있습니다. 목표에 해당되는 시리즈와, 실적에 해당되는 시리즈입니다.
기본적으로 모든 속성은 Cartesian 공통 속성과 같습니다. 이에 대한 설명은 <예제 20 Cartesian 차트의 공통적인 속성 설명> 을 참고하십시오.

시리즈를 정의 할 때 반드시 지켜야 할 점은 실적에 해당되는 시리즈를 먼저 정의 한 후 목표에 해당되는 시리즈를 정의 해야 합니다.

	2D Linear Type	3D Cylinder Type	3D Cylinder-Bar Type
차트 노드	Combination2DChart	Combination3DChart	Combination3DChart
실적에 해당되는 시리즈 노드	Target2DResultSeries	Target3DResultSeries	HTarget3DResultSeries
목표에 해당되	Target2DGoalSeries	Target3DGoalSeries	HTarget3DResultSeries

는 시리즈 노드			

<표 25 목표 대비 실적 차트 노드 정의표>

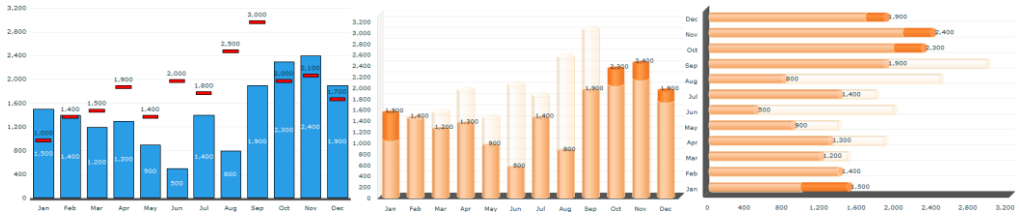
아래는 2D 목표 대비 실적 차트 레이아웃 예제입니다.

```
<rMateChart backgroundColor="0xFFFFEE"  cornerRadius="12" borderStyle="solid">
	<Options>
		<Caption text="목표 대비 실적 차트"/>
		<SubCaption text="Linear Type"/>
	</Options>
	<Combination2DChart showDataTips="true">
		<horizontalAxis>
			<CategoryAxis categoryField="Month" />
		</horizontalAxis>
		<series>
			<!-- 실적에 해당 필히 순서 준수-->
			<Target2DResultSeries yField="Result" displayName="Result">
				<showDataEffect>
					<SeriesInterpolate />
				</showDataEffect>
			</Target2DResultSeries>
			<!-- 목표에 해당 -->
			<Target2DGoalSeries yField="Goal" displayName="Goal">
				<showDataEffect>
					<SeriesInterpolate />
				</showDataEffect>
			</Target2DGoalSeries>

		</series>
	</Combination2DChart>
</rMateChart>
```

<예제 44 2D 목표 대비 실적 차트(Linear Type) 레이아웃 예제>

3D 실린더 형 목표 대비 실적 차트 레이아웃은 위 2D 목표 대비 실적 차트 레이아웃에서 2D 라는 글자를 3D 라는 글자로 바꿔주기만 하면 됩니다.

목표 대비 실적 차트 출력 모습은 다음과 같습니다.



<그림 28 목표 대비 실적 차트 출력 결과(2D, 3D)>

6.18. 스크롤 차트

스크롤 차트는 데이터양이 많은 경우 지정된 데이터 개수만을 표시하고 다음 또는 이전 데이터는 스크롤을 통해 데이터를 표시하게끔 하는 차트입니다. 스크롤 차트는 현재 2D 차트에 한해 지원하고 있습니다. 스크롤 차트를 생성하기 위해서는 기존 2D 차트 레이아웃에서 <ScrollableAxisRenderer/> 와 <CategoryLinearAxis/> 를 추가 작성할 필요가 있습니다.

CategoryLinearAxis 는 LinearAxis 를 상속받아 만든 클래스이므로 LinearAxis 의 모든 속성을 상속 받았습니다. 다음은 CategoryLinearAxis 추가 속성에 대한 설명입니다.

속성 명	유효 값	설 명
categoryField	데이터의 특정 필드	스크롤이 있는 해당 축에 나타낼 축 라벨에 표시할 데이터 필드입니다.

<표 26 CategoryLinearAxis 속성 설명표>

ScrollableAxisRenderer 의 속성 및 유효 값에 대한 설명입니다.

속성 명	유효 값	설 명
visibleItemSize	Number	스크롤 차트가 한번에 표시할 데이터의 개수입니다. 예를 들어 10 이라면 10개의 데이터만 표시하고 다음 데이터는 스크롤이 생성됩니다.
scrollSensitivity	1~10 (기본값:4)	스크롤의 민감도를 나타냅니다. 값이 작아질수록 스크롤의 이동 폭이 커집니다.

<표 27 ScrollableAxisRenderer 속성 설명표>

스크롤 차트 레이아웃을 작성하기 위해 다음 단계를 차례로 밟으십시오.

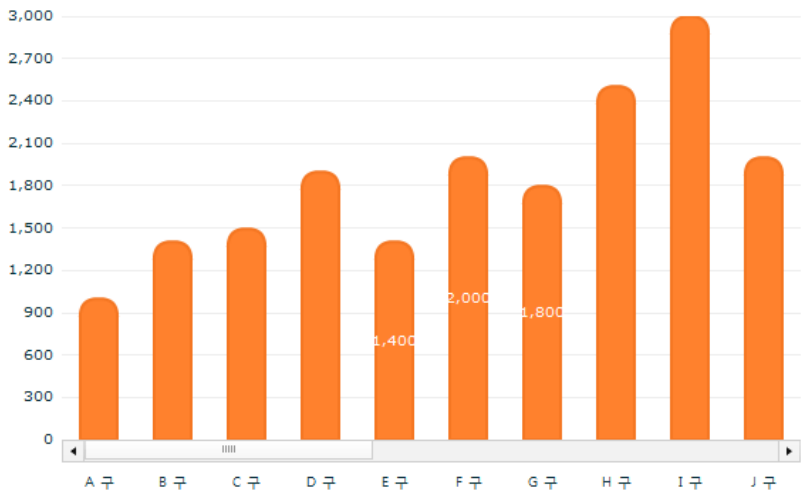
- 기존 2D 차트 레이아웃 작성법 대로 레이아웃을 작성합니다. 예를 들어 칼럼 2D 차트를 스크롤 가능한 차트로 만들어 보겠습니다.

- 칼럼 차트는 스크롤 방향이 가로 방향이므로 가로축(horizontalAxis) 속성에 CategoryLinearAxis 를 value 로 할당합니다.(바차트의 경우엔 세로축 속성에 할당)
- 가로축렌더러(horizontalAxisRenderers) 속성 값으로 ScrollableAxisRenderer 를 할당합니다.

위 방법처럼 기존 차트를 스크롤 차트로 변경하는 방법은 해당 축과 해당 축 렌더러에 할당하는 값을 조금만 수정하면 됩니다.

```
<rMateChart backgroundColor="0xFFFFEE"  cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="스크롤 2D 칼럼 차트"/>
        </Options>
        <Column2DChart showDataTips="true" gutterRight="10">
            <series>
                <Column2DSeries id="cs1" yField="Data1" displayName="Data1"
itemRenderer="net.riamore.rmate.charts.renderers.SemiCircleColumnItemRenderer"/>
            </series>
            <horizontalAxis>
                <CategoryLinearAxis id="hAxis" categoryField="Gu"/>
            </horizontalAxis>
            <horizontalAxisRenderers>
                <ScrollableAxisRenderer axis="{hAxis}" visibleItemSize="10" scrollSensitivity="10"/>
            </horizontalAxisRenderers>
        </Column2DChart>
</rMateChart>
```

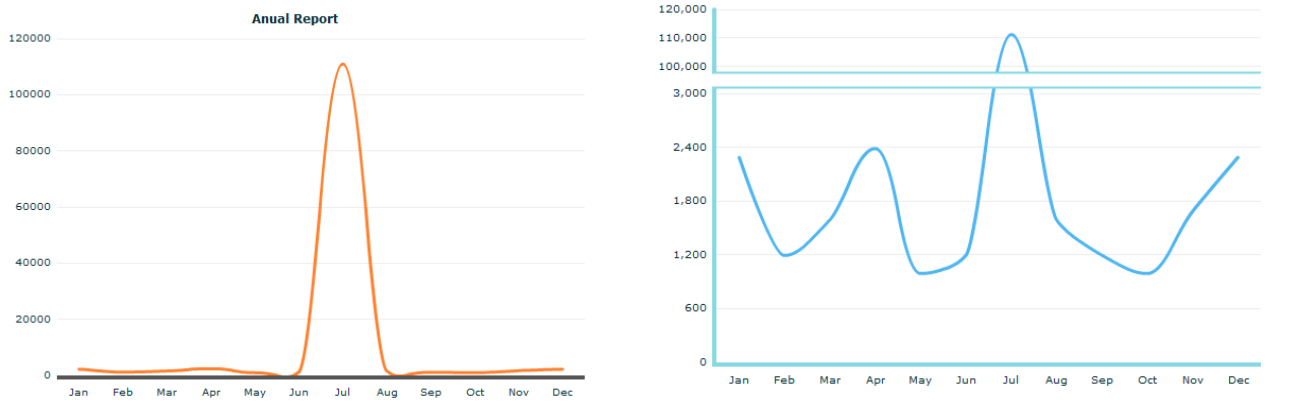
<예제 45 스크롤 2D 칼럼 차트 예제>



<그림 29 스크롤 2D 칼럼 차트 출력 모습>

6.19. 브로큰 축(BrokenAxis) 차트

브로큰 축 차트는 데이터간의 값 편차가 심할 경우 사용합니다. 예를 들어 대부분의 데이터가 3000 미만인데 특정 데이터는 10 만이 넘어가는 데이터와 같은 경우 차트는 이를 모두 표현하기 위하여 y 축에대한 값을 10 만을 넘는 값을 설정하게 됩니다. 이로 인하여 3000 미만의 데이터들의 변동에 대한 것을 한눈에 알아 볼 수 없습니다.



일반 차트 (데이터 3000미만들과 10만의 데이터)　　브로큰 축 차트(데이터 3000미만들과 10만의 데이터)

< 그림 30 브로큰 축 차트 출력 모습 >

브로큰 축 차트는 위와 같은 데이터 값 편차가 클 경우 사용하면 데이터 변동에 대한 것을 한눈에 보기 좋게 됩니다.

브로큰 축 차트 레이아웃은 아래와 같이 설정 합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFFF" borderStyle="solid" cornerRadius="5">
        <Options>
                <Caption text="Anual Report"/>
        </Options>
        <Column2DChart showDataTips="true">
                <horizontalAxis>
                        <CategoryAxis categoryField="Month"/>
                </horizontalAxis>
                <verticalAxis>
                <!-- BrokenAxis를 설정 하시려면 해당 축이 설정될 곳에 BrokenAxis를 설정하십시오. -->
                        <BrokenAxis id="vAxis" brokenMinimum="3000" brokenMaximum="110000"
brokenRatio="0.8"/>
<!-- brokenMinimum - Broken축이 시작될 값 입니다. -->
<!-- brokenMaximum - Broken축이 끝날 값 입니다. -->
```

```
<!-- brokenRatio - Broken축이 그려질 위치 값입니다. 0 ~ 1이 유효값이며 0에 가까울수록 축의 최소값에 -->
<!-- 가까워지며 1에 가까워질수록 축의 최대값에 가까워집니다. -->
                </verticalAxis>
                <verticalAxisRenderers>
                        <BrokenAxis2DRenderer axis="{vAxis}"/>
                        <!-- BrokenAxis를 사용할 경우에 BrokenAxis3DRenderer를 설정 합니다. -->
                        <!-- 이 외의 렌더러를 설정할 경우 올바르게 표현이 되지 않습니다. -->
                </verticalAxisRenderers>
                <series>
                        <Column2DSeries yField="Profit" displayName="Profit">
                                <showDataEffect>
                                        <SeriesInterpolate/>
                                </showDataEffect>
                        </Column2DSeries>
                </series>
                <annotationElements>
                        <CrossRangeZoomer enableZooming="false"/>
                        <!-- BrokenAxis는 확대/축소를 지원하지 않습니다. -->
                </annotationElements>
        </Column2DChart>
</rMateChart>
```

< 예제 46 브로큰 축 차트 예제 >

주의 사항 –

브로큰 축 차트는 Line,Area,Column,Bar 차트 이외의 차트는 지원하지 않습니다.

브로큰 축 차트는 CrossRangeZoomer 의 확대 / 축소, HistoryChart 를 지원하지 않습니다.

6.20. 히스토리 차트

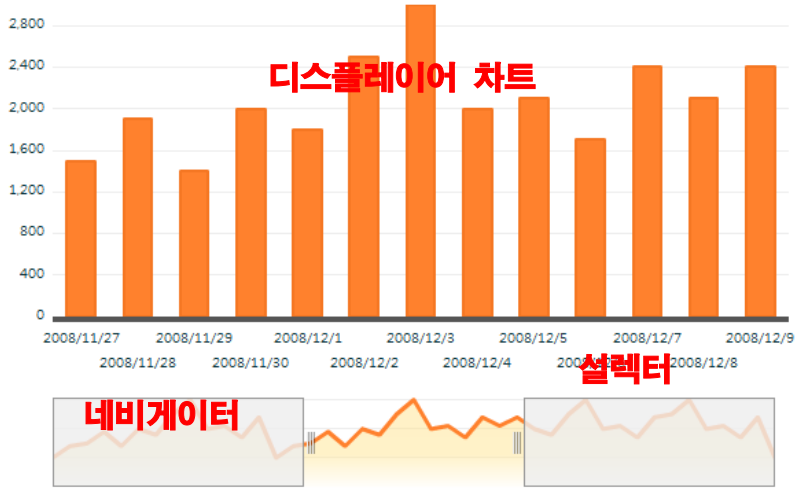
히스토리 차트의 기본적인 사상은 스크롤 차트와 같습니다. 그러나 히스토리 차트는 더 많은 데이터의 경우 전체 데이터의 흐름을 나타내는 네비게이터(또는 overview)가 있어 특정 데이터 영역을 자유롭게 넘나들며 특정 데이터의 개수가 아닌 유동적으로 데이터의 개수를 사용자가 조정하여 표현할 수 있습니다. 히스토리 차트는 <HistoryChart/> 노드로 시작과 끝을 지시합니다. 히스토리 차트의 구성은 디스플레이어, 네비게이터, 설렉터로 구성되어 있습니다.

- 디스플레이어(displayer)
  디스플레이어 차트는 사용자가 선택된 영역만이 표시하는 차트입니다. 실제로 사용자가 보고자 하는 부분의 데이터를 표시합니다.

- 네비게이터(navigator)
  네비게이터는 전체 데이터를 표현하여 데이터의 흐름을 전체적으로 볼 수 있습니다. 이는 곧 오

버뷰 역할을 합니다.

- 설렉터(selector)
  설렉터는 네비게이터에서 사용자가 특정 영역을 선택하게끔 하는 역할을 합니다. 셀렉터에서 선택된 영역이 곧 디스플레이어 차트에 표현됩니다.



<그림 31 히스토리 차트 구성도>

HistoryChart 의 속성 및 유효 값 설명에 대한 표입니다.

속성 명	유효 값	설 명
displayerChart	Displayer	히스토리 차트의 디스플레이어차트를 정의합니다.
navigator	Navigator	네비게이터를 정의합니다.
selector	HistoryRangeSelector	설렉터를 정의합니다.

<표 28 히스토리 차트 속성 및 유효값 설명>

<HistoryChart/>의 속성으로 각각 displayerChart 와 navigator 의 유효 값인 Displayer 와 Navigator 는 각각 히스토리에서만 적용되는 차트입니다. Displayer 는 Cartesian 공통 속성을 모두 갖고 있으며 Navigator 는 Area2DChart 의 모든 속성을 갖고 있습니다.

HistoryRangeSelector 의 속성 및 유효 값에 대한 설명입니다.

속성 명	유효 값	설 명
startingRange	centr | left | right	히스토리 차트가 처음 로딩 시 표시할 데이터 영역 부분입니다.
visibleItemSize	1~100	처음 로딩 시 디스플레이어 차트가 표시할 데이터의 양을 나타냅니다. 데이터의 양은 전체 데이터의 퍼센티지 수치입니다.

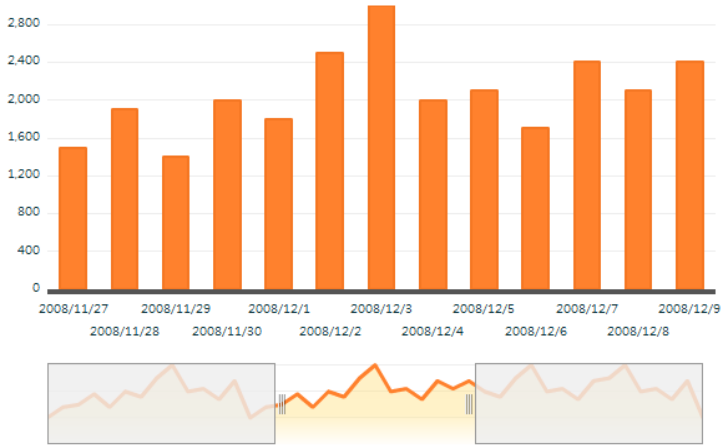
<표 29 HistoryRangeSelector 속성 설명표>

히스토리 차트 레이아웃 작성 예입니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
    <Options>
        <Caption text="히스토리 2D 차트"/>
    </Options>
    <HistoryChart>
        <displayerChart> <!-- 디스플레이 차트 정의 -->
            <Displayer id="chart1" showDataTips="true" width="100%" height="100%">
                <horizontalAxis>
                    <CategoryAxis id="mainHAxis" categoryField="Date"/>
                </horizontalAxis>
                <series>
                    <Column2DSeries id="columnSeries"
                        yField="Data1" fill="0xB0C759"
                        displayName="Data1">
                        <showDataEffect>
                            <SeriesInterpolate duration="1000"/>
                        </showDataEffect>
                    </Column2DSeries>
                </series>
            </Displayer>
        </displayerChart>
        <navigator>
            <Navigator id="navi"  width="100%" height="100">
                <horizontalAxis>
                    <CategoryAxis categoryField="Date" id="naviHAxis"/>
                </horizontalAxis>
                <horizontalAxisRenderers>
                    <Axis2DRenderer axis="{naviHAxis}" visible="false"/>
                </horizontalAxisRenderers>
                <verticalAxis>
                <LinearAxis id="vAxis"/>
                </verticalAxis>
                <verticalAxisRenderers>
                <Axis2DRenderer axis="{vAxis}" visible="false"/>
                </verticalAxisRenderers>
                    <backgroundElements>
                <GridLines direction="horizontal"/>
                </backgroundElements>
                    <series>
```

```
                        <Area2DSeries name="A" yField="Data1"/>
                    </series>
                </Navigator>
            </navigator>
            <selector><!-- 시작 시 보여지는 차트는 총 데이터의 약 30%이며 센터 영역입니다. -->
                <HistoryRangeSelector width="100%" startingRange="center" visibleItemSize="30"/>
            </selector>
    </HistoryChart>
</rMateChart>
```

<예제 47 히스토리 차트 레이아웃 예제>



<그림 32 히스토리 차트 출력 모습>

6.21. From-To 차트

From-To 차트는 칼럼 차트 계열과 바 차트 계열에 시작점 데이터와 끝 점 데이터를 삽입하여 데이터 영역을 표현하는 차트입니다.
이 데이터 영역을 활용하여 waterfall 차트, step 차트 생성이 가능합니다.
칼럼 시리즈(Column2DSeries, Column3DSeries) 와 바 시리즈(Bar2DSeries, Bar3DSeries) 그리고 영역 시리즈(Area2DSeries) 의 추가 속성으로 minField 가 시작점 데이터에 해당됩니다.
From-To 차트의 모든 속성은 각각 칼럼, 바, 영역 차트와 동일합니다.

이를 활용하여 칼럼 2D 차트로 waterfall 차트를 생성해 보도록 하겠습니다.
먼저 minField 에 해당하는 수치필드를 포함한 데이터를 정의합니다.

배열 형태로 정의하면 아래와 같습니다.

```
var data= [
            {"cat":"1월", "from":1000, "to":5000},
           {"cat":"2월", "from":5000, "to":12000},
           {"cat":"3월", "from":12000, "to":16000},
           {"cat":"4월","from":16000, "to":10000},
            {"cat":"5월", "from":10000, "to":100},
            {"cat":"6월", "from":100, "to":-1000},
            {"cat":"7월", "from":-1000, "to":3000},
            {"cat":"8월", "from":3000, "to":8000},
            {"cat":"9월", "from":8000, "to":12000},
            {"cat":"10월", "from":12000, "to":14000},
            {"cat":"11월", "from":14000, "to":7500},
            {"cat":"12월", "from":7500, "to":2500},
            {"cat":"내년 목표", "from":0, "to":20000}
            ];
```

<예제 48 From-To 차트 데이터 정의 예제>

위 데이터는 한해 매출 증감을 나타낸 예시 데이터입니다.
이 데이터를 차트로 표현하기 위해 작성된 레이아웃은 아래와 같습니다.

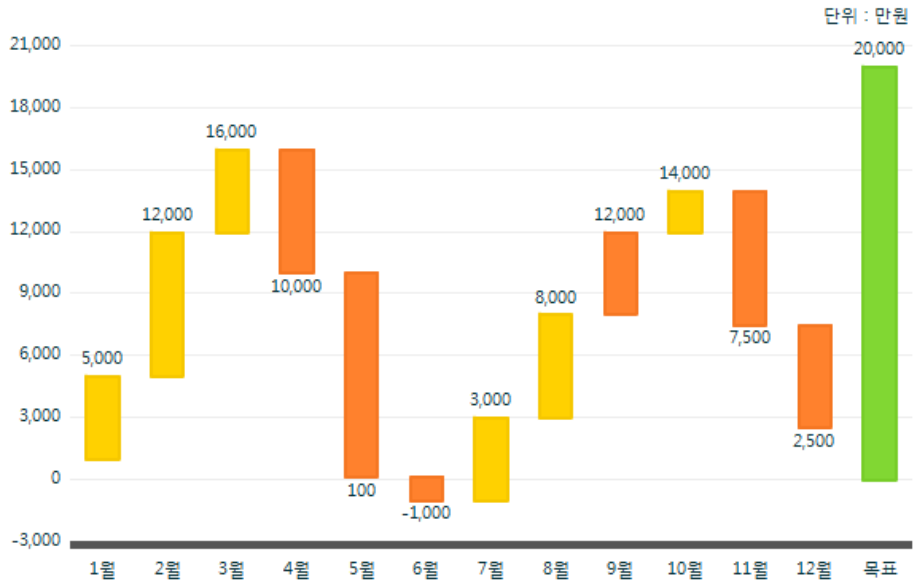
```
<rMateChart backgroundColor='0xFFFFEE'  cornerRadius='12' borderStyle='solid'>
        <Options>
            <Caption text='2009 월별 매출액 증감 비교 (Waterfall)'/>
            <SubCaption text='단위 : 만원' textAlign='right' fontSize='11' paddingRight='20'/>
        </Options>
        <NumberFormatter id='fmt'/>
        <Column2DChart showDataTips='true' dataTipJsFunction='dataTipFunc' fontSize='11'>
            <seriesFilters>
                <DropShadowFilter alpha='0'/>
            </seriesFilters>
            <horizontalAxis>
                <CategoryAxis categoryField='cat'/>
            </horizontalAxis>
            <verticalAxis>
                <LinearAxis formatter='{fmt}'/>
            </verticalAxis>
            <series>
                <Column2DSeries id='series1' minField='from' yField='to' fillJsFunction='fillFunc'
itemRenderer='net.riamore.rmate.charts.renderers.BoxItemRenderer'>
                    <showDataEffect>
                        <SeriesSlide direction='up' duration='1000'/>
                    </showDataEffect>
```

```
                        <stroke>
                            <Stroke weight='1' color='0x999999' alpha='0.7'/>
                        </stroke>
                    </Column2DSeries>
                </series>
            </Column2DChart>
</rMateChart>
```

<예제 49 From-To 차트 레이아웃 예제>

위 레이아웃 예제에서 minField 속성에 데이터의 from 필드를 삽입한 것을 볼 수 있습니다. 따라서 from 필드는 시리즈 아이템들의 시작점이 됩니다.

또한 fillJsFunction 과 dataTipJsFunction 을 정의하여 데이터팁과 채우기색을 사용자 정의 한 것을 볼 수 있습니다. 데이터팁과 채우기색의 사용자 정의 함수에 대한 설명은 '8.11 사용자 정의 함수 설정하기' 를 참고하여 주십시오.
위 예제를 출력한 결과는 아래와 같습니다.

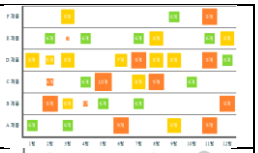
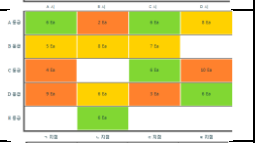


<그림 33 From-To 차트(waterfall) 출력화면>

6.22. 매트릭스 차트

매트릭스 차트는 x,y 값으로 위치를 정하고 z 값으로 크기를 정하여 표현하는 차트입니다.
매트릭스 차트의 표현 유형으로는 아래와 같습니다.

매트릭스 차트의 유형(type)	설 명	예제 모습

renderer	x,y로 위치를 잡고 z로 도형 크기를 정하여 표현합니다.	
image	x,y로 위치를 잡고 z로 이미지 크기를 정하여 표현합니다.	
fill	x,y 로 위치를 잡으며 z값은 무시되어 해당 칸을 색채움으로 표시합니다.	
plot	x,y로 위치를 잡으며 z값은 무시되어 해당칸에 plot형태로 표현합니다.	

<표 30 매트릭스 차트의 유형 표>

매트릭스 차트 유형을 결정하는 방법은 아래와 같습니다.

```
<Matrix2DChart showDataTips="true" type="renderer" selectionMode="single">
```

Matrix2DChart 의 속성으로는 아래와 같습니다.

속성 명	유효 값	설 명
type	renderer, image, fill, plot (default : renderer)	매트릭스가 그려질 때의 표현 방법입니다.
drawType	radius, area (default : radius)	매트릭스 차트가 그려질 때 어떤 기준으로 그릴지 정합니다. radius : 반지름 기준, area : 면적 기준 (fill, plot은 무시됩니다)

<표 31 매트릭스 차트 속성 표>

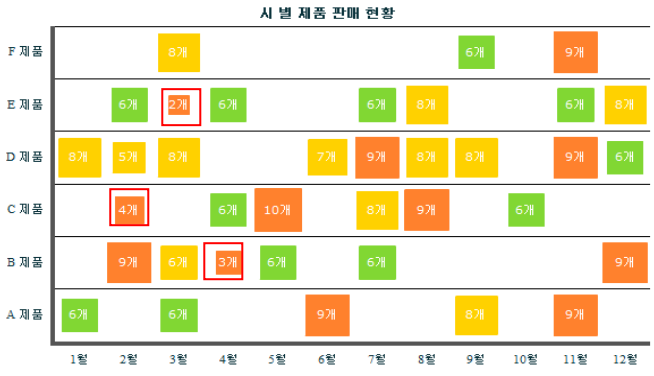
Matrix2DChart 의 type 이 renderer, image 일 때 어떤 기준으로 그릴지 정하는 방법은 아래와 같습니다.

```
<Matrix2DChart showDataTips="true" type="renderer" drawType="radius" selectionMode="single">
```

아래 그림은 drawType="radius" 와 "area"의 비교 모습입니다.

drawType = “radius”

drawType = “area”



<Matrix2DSeries /> 매트릭스 차트 시리즈의 속성으로는 아래와 같습니다.

속 성	유효 값	설 명
renderer	rectangle, circle, diamond, upTriangle, downTriangle, star default : rectangle	도형 종류를 변경하고 싶으실 때 정의하십시오.
imageSource	해당 이미지의 주소	이미지로 표현하고 싶으실 때 해당 주소를 넣어주십시오.

<표 32 매트릭스 시리즈의 속성 및 유효값 설명>

매트릭스 차트의 기본도형(renderer)을 변경하고 싶으시다면 아래와 같이 하십시오.

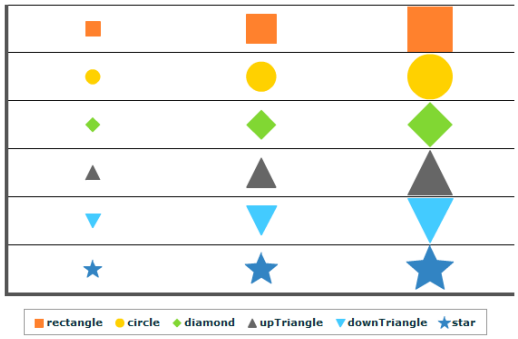
```
<Matrix2DSeries ….. renderer=”circle” // renderer=”star” …..>
```

매트릭스 차트를 이미지로 표현하고 싶으시다면 imageSource 에 해당 이미지 주소를 넣어주십시오.

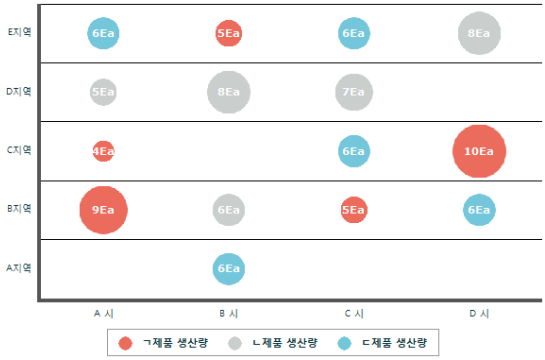
```
<Matrix2DSeries ….. imageSource=”./Images/icon1.png” …..>
```

아래 그림은 위 속성들을 변경 후의 모습입니다.

renderer 변경



이미지

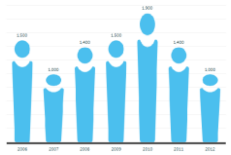
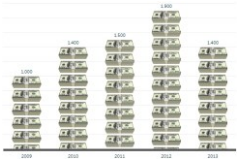


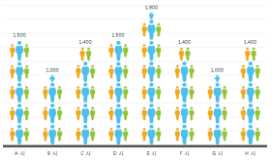
6.23. 이미지 차트

이미지 차트는 칼럼 형태의 차트를 특정 이미지로 표현하는 차트입니다. 이미지 차트의 유형을 결정하는 요소는 크게 두가지로 나뉩니다.

- 첫째, 이미지 고유의 비율에 따라 이미지를 표현할지의 여부
- 둘째, 데이터 하나를 표현할 때 단일 이미지 또는 다중 이미지를 사용할지의 여부

위 두가지 옵션에 따라 결정되는 이미지 차트의 유형은 아래와 같습니다.

이미지 차트 유형 (imageDisplayType)	이미지 고유 비율 (true,false)	설 명	예제 모습
single	True(정비율)	데이터를 단일 이미지로 표현하되 이미지 고유 사이즈 유지, 남는 여백은 막대를 세웁니다.	
	False(차등비율)	데이터를 단일 이미지로 표현하되 이미지 고유 사이즈가 아닌 차트가 결정한 사이즈대로 표현합니다.	
singleRepeat	True(정비율)	데이터를 단일 이미지로 표현하되 이미지 고유 사이즈 유지, 남는 여백은 **이미지의 반복**으로 처리	
	False(차등비율)	singleRepeat 유형에서의 차등비율은 존재하지 않습니다.	X
multiple	True(정비율)	Multiple 유형에서의 차등비율은 존재하지 않습니다.	X

	False(차등비율)	데이터를 다중 이미지로 표현합니다. 각각의 이미지는 고유 값을 갖습니다. 이 값은 차트의 사이즈에 의해 계산되어 이미지로 전달됩니다.	

<표 33 이미지 차트 유형 표>

기본적으로 이미지 차트의 모든 속성은 칼럼차트와 동일합니다. 이미지 차트 고유 속성은 이미지 시리즈의 imgSource 입니다.
imgSource 속성은 이미지의 경로와 어떻게 표현할지를 결정하는 속성입니다.

```
<ImageSeries yField="Data1" imageDisplayType="singleRepeat" displayName="분양수" styleName="seriesStyle"
formatter="{numFmt}">
    <imgSource>
        <ImageSourceItem url="../Samples/Images/10000.png"/>
    </imgSource>
        <showDataEffect>
            <SeriesSlide duration="1000" direction="up"/>
        </showDataEffect>
</ImageSeries>
```

위 예제는 단일 시리즈를 정의한 것입니다. imageDisplayType 을 singleRepeat 로 설정하여 하나의 이미지를 반복적으로 처리하도록 하였습니다.
그리고 imgSource 노드에서 이미지의 경로를 정의합니다.
imgsource 노드의 자식 노드로 설정가능한 유효값은 ImageSourceItem 노드입니다.
ImageSourceItem 노드는 이미지의 경로와 이미지를 정비율로 표현 할지 여부를 나타냅니다.

속성명	유효값	설 명
maintainAspectRatio	True | false(기본값 : true)	이미지 고유 비율대로 표현할지 여부
url	이미지 주소(URL)	이미지 파일의 주소입니다.
Value	Number	이미지가 갖을 고유 value 입니다.(multiple 에서만 해당됩니다.)

<표 34 이미지 차트 ImageSourceItem 속성 설명>

3 개의 이미지가 각각 고유 값을 갖는 multiple 이미지 차트 레이아웃을 작성하면 아래와 같습니다.

```
<ImageChart id="chart" showDataTips="true" gutterLeft="20" gutterRight="20" showLabelVertically="true">
            <horizontalAxis> <!—X축 정의 -->
                <CategoryAxis id="hAxis" categoryField="Region"/>
            </horizontalAxis>
```

```
                    <verticalAxis><!—Y축 정의 -->
                            <LinearAxis id="vAxis"/>
                    </verticalAxis>
                    <series>
                    <ImageSeries yField="Data1" imageDisplayType="multiple" styleName="seriesStyle"
formatter="{numFmt}">
                        <imgSource>
                         <!—value 100 을 갖는 첫번째 이미지 -->
                          <ImageSourceItem maintainAspectRatio="false" url="../Samples/Images/3-1.png"
value="100"/>
                        <!—value 200 을 갖는 두번째 이미지 -->
                          <ImageSourceItem maintainAspectRatio="false" url="../Samples/Images/3-2.png"
value="200"/>
                        <!—value 300 을 갖는 세번째 이미지 -->
                          <ImageSourceItem maintainAspectRatio="false" url="../Samples/Images/3-3.png"
value="300"/>
                        </imgSource>
                    </ImageSeries>
                    </series>
                    <horizontalAxisRenderers>
                            <Axis2DRenderer axis="{hAxis}" fontSize="11"/>
                    </horizontalAxisRenderers>
                    <verticalAxisRenderers><!—Y 축은 visible off -->
                            <Axis2DRenderer axis="{vAxis}" visible="false" includeInLayout="false"/>
                    </verticalAxisRenderers>
            </ImageChart>
```

위 레이아웃을 실행한 모습은 아래와 같습니다.

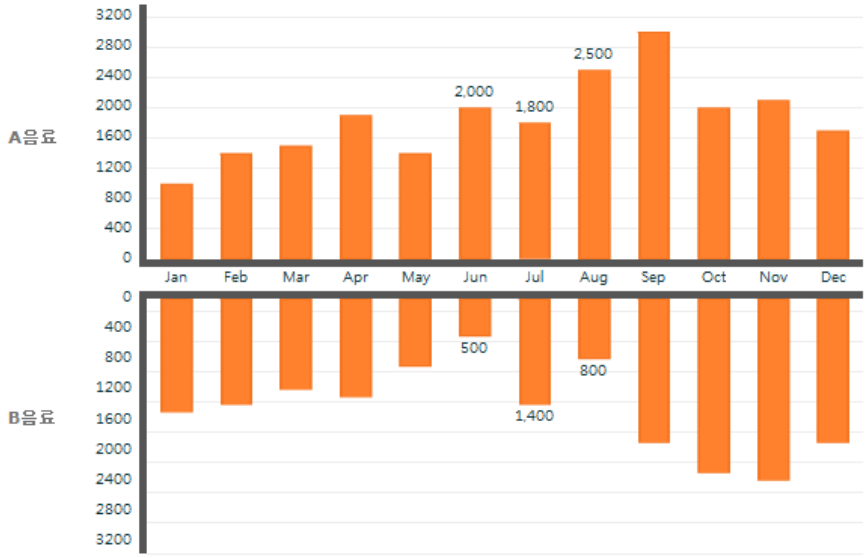
<그림 34 이미지 차트 실행화면>

6.24. 윙 차트

윙 차트는 기존 차트와는 달리 양옆 혹은 위아래로 출력되는 형태의 차트 입니다. 예를 들어 해당 차트는 특정 제품에 대하여 여성 이용자와 남성 이용자의 구매 현황 같은 것을 시각적으로 보기 쉽게 표현하는 차트 입니다.

현재 윙 차트에는 두가지의 형태만 존재합니다. 해당 형태는 Column 과 Bar 형 이 두가지만 존재합니다.

1. Column 형



2. Bar 형

위 그림들을 살펴보시면 위아래, 양옆으로 해당 월에 대한 값들을 한눈에 비교해 보실 수 있습니다.
윙 차트의 컬럼 형태를 출력하고 싶으시다면 레이아웃을 아래와 같이 작성 하십시오.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<rMateChart backgroundColor="0xFFFFFF" borderStyle="solid" cornerRadius="5">
        <Options>
                <Caption text="Anual Report"/>
        </Options>
        <!-- 윙 컬럼 2D 차트를 생성 하실 경우에는 Column2DWingChart를 정의 하십시오. -->
        <Column2DWingChart showDataTips="true">
                <horizontalAxis>
                        <CategoryAxis categoryField="Month"/>
                </horizontalAxis>
                <series>
                <!-- Column2DWingChart의 series속성으로는 Column2DWingSeries를 설정 하여야 합니다. -->
                <!-- 일반 Column2D,3DChart에서는 데이터 필드로 yField만 존재 하였지만 Wing Chart에서는-->
                <!-- yFieldOpp란 데이터 필드가 존재하므로 해당 데이터 필드도 설정 해주셔야 합니다. -->
                <!-- yField는 위쪽 영역, yFieldOpp는 아래쪽 영역 입니다. -->
                        <Column2DWingSeries yField="Profit" yFieldOpp="Cost" labelPosition="inside">
                                <showDataEffect>
                                        <WingSeriesInterpolate/>
                                </showDataEffect>
                        </Column2DWingSeries>
                </series>
        </Column2DWingChart>
</rMateChart>
```

위와 같이 작성하시면 윙차트의 Column 형태를 출력 하실 수 있습니다.

6.25. 실시간-프리미엄 차트

실시간-프리미엄 차트는 6.15 실시간 차트를 업그레이드 한 차트입니다. 기본적으로 실시간 차트와 비교하여 통신 방법에 의한 변화는 없습니다. HTTPService 를 통하여 RPC 요청을 하고 응답으로 차트에 표현하게 됩니다. 그러나 실시간-프리미엄 차트만의 특징은 아래와 같습니다.

1. 차트에 표현되는 시리즈 각각에 해당 URL 을 통하여 주기를 다르게 표현할 수 있습니다. 예를 들어 3 개의 시리즈에 각각 5 초, 10 초, 1 분 단위로 개별 주기를 줄 수 있습니다.
2. 기존 실시간 차트는 오른쪽 방향에서 왼쪽 방향으로 특정 시간(또는 데이터량) 만큼 출력을 한 반면 실시간-프리미엄 차트는 왼쪽에서 오른쪽 방향으로 서버 사이드에서 결정한 시간 범위만큼 출력할 수 있습니다. 개별 시리즈의 주기가 다르므로 개별 시리즈가 표현될 시간 범위를 개별적으로 줄 수 있습니다.
3. 초기 데이터를 클라이언트의 차트에 표현해야 할 경우 차트가 로딩 완료 된 시점에 초기 데이터를 받아와 출력할 수 있습니다.
4. 실시간-프리미엄 차트의 **X 축 정의는 항상 DateTimeAxis 로 정의**하여야 합니다. 즉, 기존 실시간 차트의 데이터량 만큼 출력시키는 CategoryAxis 는 지원하지 않습니다.

실시간-프리미엄 차트를 생성하기 위해서는 기존 실시간 차트처럼 <RealTimeChart/> 노드로 레이아웃을 작성하지 않습니다. 실시간-프리미엄 차트는 어떤 종류의 rMate 차트도 가능합니다.
단, 실시간-프리미엄 차트를 가능케 하는 <HttpMultiServiceRepeater /> 노드를 차트 아래에 정의하여야 합니다.
HttpMultiServiceRepeater 는 원격 호출 할 오퍼레이션들을 갖습니다. 이 오퍼레이션은 시리즈 개수만큼 정의되어 각 주기마다 RPC 요청하여 응답으로 해당 시리즈에 값을 넘겨 출력하게끔 합니다.

HttpMultiServiceRepeater 노드의 속성에 대한 설명과 유효값은 아래와 같습니다.

속성명	유효값	설 명
baseURL	URL 주소	RPCItem 속성 'url'의 기본 URL에 해당됩니다. 즉, RPCItem 의 url 은 "baseURL + RPCItem의 url" 이 됩니다.
method	get | post (기본값 :get)	HTTP 프로토콜 메소드입니다. Get방식인지 Post 방식인지를 결정.
requestTimeout	Second(초)	요청 후 응답 대기 시간입니다.
targetController	차트 id	RPCItem 의 target 의 컨트롤러에 해당됩니다. 즉, 언제나 차트가 됩니다.
showErrorMessage	true|false(기본값:true)	RPC 요청 시 실패 또는 에러 발생 시 Alert 메시지 창을 띄울 지를 나타냅니다. 만약 showErrorMessage 속성을 false 로 설정한다면 어떤 메시지도 출력하지 않습니다.

<표 35 HttpMultiServiceRepeater 속성 및 유효값 설명표>

다음은 HttpMultiServiceRepeater 노드의 RPCList 값에 해당되는 RPCItem 노드의 속성 및 유효값에 대한 설명입니다.

속성명	유효값	설 명
name	String	RPCItem 의 name 입니다. 반드시 정의하십시오.(임의의 스트링)
url	String	RPCItem 속성 'url'의 기본 URL에 해당됩니다. 즉, RPCItem 의 url 은 "baseURL + RPCItem의 url" 이 됩니다.
method	get | post (기본값 :get)	HTTP 프로토콜 메소드입니다. Get방식인지 Post 방식인지를 결정합니다.
target	시리즈의 id	RPCItem 해당 url 로 요청하여 받은 데이터를 표현할 시리즈를 나타냅니다. 각각의 RPCItem 은 각각

		의 시리즈가 됩니다
interval	Seconds(초)	RPC 요청 주기를 나타냅니다. 최초 차트 로딩시 요청을 보내고 주어진 interval 간격으로 다음 요청이 이루어집니다. 예를 들어 5로 설정한 경우 5초 간격으로 요청을 하게 됩니다. Interval을 설정하지 않은 경우 주기적으로 요청이 이루어지지 않습니다. 이런 경우는 비교 데이터 즉, 갱신만 하고자 할 때 사용합니다.
concurrency	multiple | single | last(기본값:multiple)	HTTP 동일 서비스 발생 시 처리 방법을 나타냅니다. multiple : 기존 요구을 취소하지 않고 모든 요청을 보냅니다. single : 한번에 1개의 요청만 가능합니다. 중복 발생 시 에러 Alert 메시지를 띄웁니다. last : 기존의 요청을 모두 취소하고, 마지막 요청만을 보냅니다.
retryCount	Number(기본값:30)	RPC 요청 시 실패 응답이 온 경우 재시도 횟수를 나타냅니다.

<표 36 RPCItem 노드 속성 및 유효값 설명표>

예로 컴비네이션 2D 차트를 통하여 실시간-프리미엄 차트를 생성해 보겠습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFFF"  cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="다른 주기를 갖는 데이터 실시간 표현"/>
                <SubCaption text="본 샘플은 랜덤 데이터입니다." textAlign="right" fontSize="11"/>
                <Legend fontSize="11" useVisibleCheck="true"/>
        </Options>

        // 각종 포메터 정의
        <DateFormatter id="dateOrgFmt" formatString="YYYY/MM/DD HH:NN:SS"/>
        <DateFormatter id="dateFmt" formatString="HH:NN:SS"/>
        <DateFormatter id="dateFmt2" formatString="HH:NN"/>
        <NumberFormatter id="numFmt"/>

// 칼럼시리즈와 라인 시리즈 2개를 갖는 컴비네이션 2D 차트 생성
        <Combination2DChart id="chart" showDataTips="true" dataTipMode="multiple">
                <series>
```

```
                        <!-- 5 초 주기 라인 시리즈  -->
                        <Line2DSeries id="lineSeries" xField="date" yField="data5"  displayName="Data(5
Sec)">
                                <lineStroke>
                                        <Stroke color="0xFF6666" weight="2" alpha="1"/>
                                </lineStroke>
                                <horizontalAxis>
                                        <DateTimeAxis id="hAxis2" displayLocalTime="true"
labelUnits="minutes" dataUnits="seconds" interval="1" formatter="{dateOrgFmt}" displayName="Time"/>
                                </horizontalAxis>
                                <verticalAxis>
                                        <LinearAxis id="vAxis2" minimum="0" maximum="150"/>
                                </verticalAxis>
                        </Line2DSeries>

                        <!-- 3 초 주기 라인 시리즈  -->
                        <Line2DSeries id="lineSeries2" xField="date" yField="data3" displayName="Data(3 Sec)"
verticalAxis="{vAxis2}" horizontalAxis="{hAxis2}">
                                <lineStroke>
                                        <Stroke color="0x339966" weight="1" alpha="1"/>
                                </lineStroke>
                        </Line2DSeries>

                        <!-- 누적량 -->
                        <Column2DSeries id="columnSeries" xField="date" yField="data60"
displayName="누적량" itemRenderer="BoxItemRenderer">
                                <horizontalAxis>
                                        <DateTimeAxis id="hAxis" displayLocalTime="true"
labelUnits="hours" dataUnits="minutes" interval="3" dataInterval="10" formatter="{dateOrgFmt}"
displayName="Time"/>
                                </horizontalAxis>
                                <verticalAxis>
                                        <LinearAxis id="vAxis" minimum="0" maximum="999"/>
                                </verticalAxis>
                                <fill>
                                        <SolidColor color="0x6666FF"/>
                                </fill>
                        </Column2DSeries>

                </series>

                <horizontalAxisRenderers>
                        <Axis2DRenderer axis="{hAxis}" placement="bottom" formatter="{dateFmt2}"
tickLength="30" minorTickLength="0" tickPlacement="inside" showLine="false">
```

```
                                        <axisStroke>
                                                <Stroke weight="1" color="0x999999"/>
                                        </axisStroke>
                                        <tickStroke>
                                                <Stroke weight="1" color="0x6666FF" alpha="0.5"/>
                                        </tickStroke>
                                </Axis2DRenderer>
                                <Axis2DRenderer axis="{hAxis2}" placement="bottom" formatter="{dateFmt}">
                                        <axisStroke>
                                                <Stroke weight="1" color="0x999999"/>
                                        </axisStroke>
                                </Axis2DRenderer>
                        </horizontalAxisRenderers>

                        <verticalAxisRenderers>
                                <Axis2DRenderer axis="{vAxis}" placement="right" formatter="{numFmt}"/>
                                <Axis2DRenderer axis="{vAxis2}" placement="left" formatter="{numFmt}"/>
                        </verticalAxisRenderers>
                </Combination2DChart>


// 실시간-프리미엄 차트 생성을 위한 HttpMultiServiceRepeater 정의.
                <HttpMultiServiceRepeater baseURL="http://demo.riamore.net/realtimeSample/" targetController="{chart}"
requestTimeout="30">
                        <RPCList>
                                <RPCItem name="rpc1" url="data3Interval.jsp" target="{lineSeries2}" interval="3"
concurrency="last" retryCount="30"/>
                                <RPCItem name="rpc2" url="data5Interval.jsp" target="{lineSeries}" interval="5"
concurrency="last" retryCount="30"/>
                                <RPCItem name="rpc3" url="data23ToCurrent2.jsp" target="{columnSeries}"
interval="600" concurrency="last" retryCount="30"/>
                        </RPCList>
                </HttpMultiServiceRepeater>
</rMateChart>
```

<예제 50 실시간-프리미엄 차트 레이아웃 예제>

위 샘플은 라인 2 개와 칼럼을 실시간으로 표현합니다. 각각의 주기는 3 초, 5 초, 10 분 입니다.
처음 차트 구동 시 초기 데이터를 위해 RPC 호출을 하여 출력한 후 주기마다 반복적으로 RPC 호출을 하여 데이터를 표현합니다.
라인 차트 2 개의 초기 데이터는 없는 상태(서버 사이드에서 이와 같이 작업함)로 차트가 구동 시부터 10 분 영역의 데이터를 실시간으로 뿌린 후 10 분이 지나면 뿌려진 데이터를 모두 삭제하고 다시 10 분 영역의 데이터를 각 주기 간격으로 출력합니다.
아래 칼럼 차트는 10 분동안 쌓여진 데이터의 누적량을 표현하는 예제로 10 분마다 RPC 호출하여 칼럼

하나씩을 더하게 됩니다.
칼럼 차트의 경우 설정된 알람 시간(매일 23 시 59 분)이 되면 출력된 데이터를 모두 갱신하게 됩니다

각 서버 스크립트 예제 파일은 제공된 시디의 다음 폴더 경로에서 확인해 볼 수 있습니다.

시디/Samples/RealtimeServerSamples/

위 예제를 위한 서버사이드 샘플을 살펴보도록 하겠습니다.

```
 리얼 타임 차트 서버사이드 예제.

 http://demo.riamore.net/realtimeSample/hourDataToday.jsp

--------------------------------------------------------------------------------
 예제 작동 방식 설명
--------------------------------------------------------------------------------

// 본 예제는 DB 데이터가 아닌 임의의 랜덤 데이터를 기반으로 합니다.
 requestAllData = true 인 경우 현재 시(hour):00 부터 현재 시:분 까지 5초 간격 데이터를 생성합니다.
 requestAllData = false 인 경우 현재 시:분에 해당하는 데이터 하나를 생성하게 됩니다.
 차트 레이아웃에서 <RPCItem> 의 속성으로 interval 을 5로 주어 5초마다 하나의 데이터를 서버에서
 가져와 기존 데이터와 adding 을 하게 됩니다.
 <nextInitDate> 값에 설정된 시간이 되면 모든 데이터를 삭제하고 다시 요청하게 됩니다.(refresh)


  아래 URL을 각각 복사하여 브라우저에 출력해 보십시오.

  http:demo.riamore.net/realtimeSample/hourDataToday.jsp?requestAllData=true

  http:demo.riamore.net/realtimeSample/hourDataToday.jsp?requestAllData=false


--------------------------------------------------------------------------------
 파라메터 설명.
--------------------------------------------------------------------------------

 requestAllData : 차트가 모든 데이터를 요청하는 플래그입니다.
                  즉, 현재 차트에 뿌려진 데이터를 갱신하고자 할 때 사용합니다.
                  requestAllData=true 로 설정되어 호출하는 순간은 다음과 같습니다.
```

1. 처음 차트가 로딩되어 처음 RPC 요청할 때.
2. 현재 시간이 <nextInitDate> 시간을 넘은 경우 true 로 설정되어 RPC 요청합니다.
requestAllData=true 로 설정된 경우 서버사이드는 다음과 같이 작업을 할 필요가 있습니다.
이는 요청 후 응답으로 받은 데이터가 정확하다는 서로간의 통신법입니다.
<infoMsg> 태그 밑에 <isInitData>true</isInitData> 로 설정.
requestAllData=false 인 경우 <isInitData>false</isInitData> 여야 합니다.

index : 차트가 최근에 받은 자료의 index를 나타냅니다.
이는 서버사이드에서 응답으로 전달해준 자료입니다.
index 를 통하여 서버와 클라이언트 간 자료 중복을 줄일 수 있습니다.
다시 말해 차트가 이전에 받은 자료의 index를 다음 호출 때 파라메터로 넘겨줌으로써
서버는 index 다음 자료를 쉽게 넘겨줄 수 있습니다.

dummy : 이 값은 IE 의 캐시 문제를 피하기 위해 넣은 더미값입니다.
경우에 따라서는 유용하게 쓰일 수 있습니다.
1970년 1월 1일 00시 부터 현재 차트가 RPC 를 보낸 순간까지의 밀리섹컨드를 나타냅니다.
이 값은 서버의 시간 기준이 아닌 클라이언트 시간 기준입니다.

XML 작성 규칙.

클라이언트가 응답으로 가져갈 XML 은 다음 규칙을 따라야 합니다.

규칙 1. 루트 노드 다음으로 <infoMsg> 노드 정보 및 서브 노드 정보는 반드시 필요합니다. (이름 변경 불가)
규칙 2. 차트가 뿌릴 데이터에 해당되는 노드는 반드시 <item> 이여야 합니다. (이름 변경 불가)

예제 XML 설명.

현재 예제는 랜덤으로 값을 나타냅니다. (DB 데이터 값이 아님)

requestAllData 파라메터의 true/false 에 따라 다르게 XML 이 출력됩니다.

1. requestAllData = false 인 경우
데이터 한개에 해당되는 XML 를 생성합니다.
이 데이터는 차트가 기존 데이터에 adding 을 합니다.

2. requestAllData = true 인 경우.

차트가 처음 로딩 시 requestAllData=true 파라메터로 요청을 합니다. 이 때 서버사이드에서 적당한 초기 데이터를 차트에 넘기십시오.

현재 샘플에선 현재 시간의 00분 부터 현재 분(minutes)까지 누적 데이터를 생성합니다.

requestAllData = true 는 초기 데이터이므로 차트가 뿌려줄 누적 데이터를 모두 select 하는 것이 좋습니다.

infoMsg 노드의 <nextInitDate> 값 시간에 다시 true 로 설정되어 요청합니다.

그 외는 모두 false로 요청.

3. <infoMsg> 노드에 대한 설명.

1. <index> 노드.

현재 샘플에선 index 파라메터를 사용하여 샘플이 작성되지 않았습니다.

그러나 이 노드는 현재 서버가 작성한 데이터의 고유값을 삽입하면 유리합니다.

차트는 항상 RPC 요청 시 차트가 최근에 응답으로 받아간 index를 갖고 요청을 합니다.

그렇기 때문에 이 파라메터를 이용하면 다음 데이터를 쉽게 전달해 줄 수 있습니다.

처음 차트 로딩 시(즉, 이전에 받아간 데이터 없을 경우)엔 init 임.

2. <timeNow> 노드.

서버의 현재 시간을 나타냅니다. 이는 데이터를 갱신(초기화)할 때 사용됩니다.

모든 클라이언트의 데이터를 일괄적으로 갱신(초기화) 하기를 원할 때 서버의 현재 시간을 삽입하여 주십시오.

이 노드를 생략한다면 모든 클라이언트의 로컬 시간을 기준으로 endDate 와 비교하여 갱신됩니다.

즉, <timeNow> 가 없다면 갱신되는 시간은 클라이언트마다 다를 수 있습니다.

3. <isInitData> 노드.

requestAllData = true 로 파라메터를 받은 경우 isInitData 노드를 true 로 설정하세요.

이는 서버와 클라이언트간 규약입니다.

4. <startDate> 노드.

차트에 표시되는 시간의 초기 시간입니다. 이 노드가 없다면 처음 데이터를 기준으로 처음 시간이 뿌려집니다.

5. <endDate> 노드.

차트에 표시할 시간의 마지막입니다. 이는 데이터가 앞으로 표현될 시간입니다.(현재 시간과 무관, 단순 데이터의 시간임.)

6. <nextInitDate> 노드.

이 값보다 현재 시간이 커진다면 차트는 갱신을 시도합니다.

이 말은 현재 차트에 뿌려진 데이터를 모두 지우고

새로 서버에 초기 데이터를 요청하는 requestAllData=true 파라메터로 요청을 합니다.

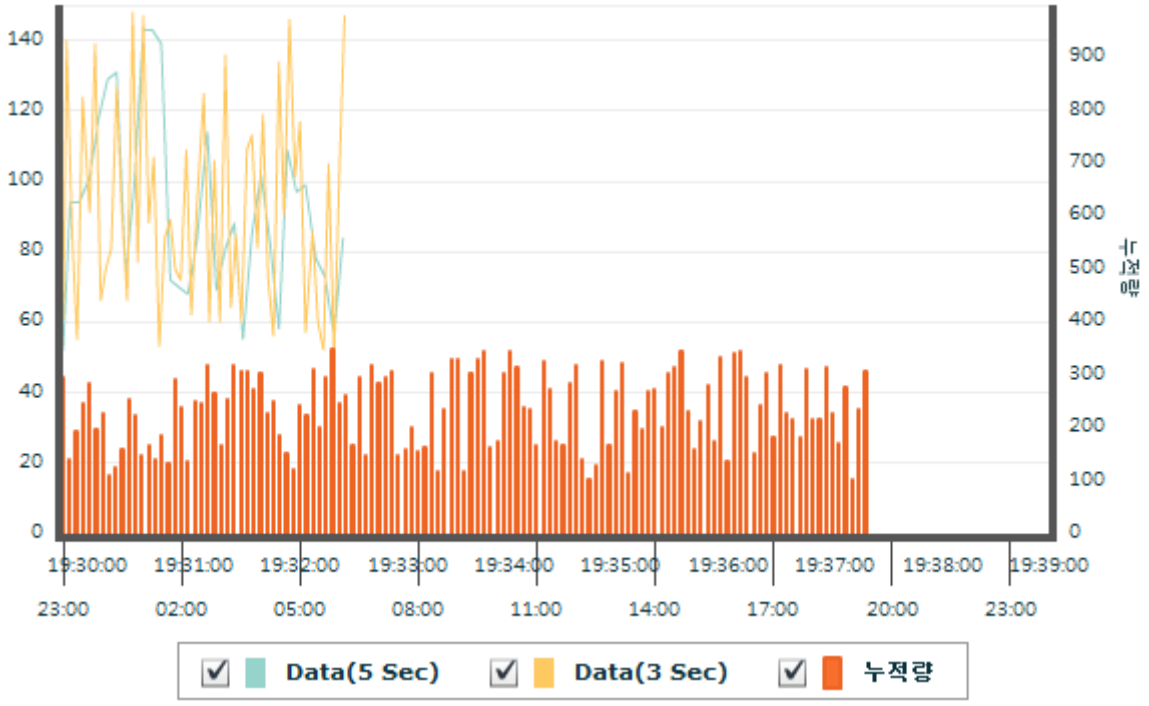
이 값은 실제 현재 시간과 관련이 있습니다. 또한 timeNow 노드 값과 깊은 관련이 있습니다.(timeNow 참고)

예를 들어 한달전 데이터를 현재 10분 주기로 보여주고 있는데 특정 시점에서 차트의 <endDate> 값까지 출력한 경우

이 시간을 참조하여 초기화를 요청하게 됩니다. (초기화는 requestAllData=true 파라메터 설정임).

<예제 51 실시간-프리미엄 서버사이드 샘플 설명>

위 레이아웃과 서버 샘플을 실행한 화면은 아래와 같습니다.



<그림 35 실시간-프리미엄 차트 실행화면>

6.26. 캔들스틱 차트

캔들스틱 차트는 전문적인 주식차트의 축소 버전입니다. 일반적인 주식차트와 같이 시가, 종가, 고가, 저가 거래량의 데이터를 설정하여 출력 할 수 있습니다. 그러나 전문적인 주식차트와 같이 마우스로 특정 지점에 선을 긋거나 특정 기호를 삽입하는 기능은 지원하고 있지 않습니다.

아래는 캔들차트에서 지원가능한 기능들입니다.

- 캔들차트 안에서의 최소, 최대값을 표현합니다.
- 최소, 최대값을 사용자가 원하는 형태로 출력할 수 있습니다.
- 특정 데이터에 공시에 대한 데이터가 존재한다면 공시기호를 표현할 수 있습니다.
- 공시기호를 일반 itemRenderer로 표현가능하며 div객체를 이용하여 특정 문자, 이미지를

표현할 수 있습니다.

- div 로 된 공시기호는 마우스 클릭 이벤트를 받을 수 있습니다.
- 시가와 종가에따라 선 색상, 막대 테두리, 막대 배경색상등을 변경 할 수 있습니다.

- 차트에 보여지는 아이템의 갯수를 변경 할 수 있습니다.

<그림 36 캔들차트 기본화면>

```
<rMateChart backgroundColor="0xFFFFFF"  cornerRadius="12" borderStyle="solid">
        <Options>
            <Caption text="Riamore CandleChart"/>
        </Options>
        // 각종 포메터 정의
        <NumberFormatter id="nft" precision="0"/>
// 한 화면에 두개의 차트를 설정 할 수 있는 DualChart 설정
        <DualChart leftGutterSyncEnable="true" rightGutterSyncEnable="true">
        // mainChart 와 subChart 의 왼쪽, 오른쪽 여백을 동기화 시킵니다.
            <mainChart>
                <Candlestick2DChart showDataTips="true" paddingBottom="0">
                    <horizontalAxis>
                        <CategoryAxis id="hAxis" categoryField="date"/>
                    </horizontalAxis>
                    <verticalAxis>
                        <LinearAxis baseAtZero="false"/>
                    </ verticalAxis>
                    <series>
// openField : 시가 closeField : 종가 highField : 고가 lowField : 저가
// showMaxValueLabel : 가장 큰 값 수치를 표현합니다.
// showMinValueLabel : 가장 작은 값 수치를 표현합니다.
                        <Candlestick2DSeries  openField="openprc"  closeField="closeprc"
highField="high"  lowField="low"  showMinValueLabel="true"  showMaxValueLabel="true"
maxLabelJsFunction="maxLabelFunc" minLabelJsFunction="minLabelFunc">
```

```
                                        // 시가 보다 종가가 높을 경우 막대의 색상 입니다.
                                        <fill>
                                                <SolidColor color="#ff0000" alpha="0.5"/>
                                        </fill>
                                        // 시가 보다 종가가 높을 경우 고가와 저가를 잇는 선의
색상 입니다.
                                        <stroke>
                                                <Stroke color="#ff0000"/>
                                        </stroke>
                                        // 시가 보다 종가가 높을 경우 막대의 테두리 색상입니다.
                                        <boxStroke>
                                                <Stroke color="#ff0000"/>
                                        </boxStroke>
                                        // 종가 보다 시가가 높을 경우 막대의 배경 색상입니다.
                                        <declineFill>
                                                <SolidColor color="#0000ff" alpha="0.5"/>
                                        </declineFill>
                                        // 종가 보다 시가가 높을 경우 고가와 저가를 잇는 선의
색상 입니다..
                                        <declineStroke>
                                                <Stroke color="#0000ff"/>
                                        </declineStroke>
                                        // 종가 보다 시가가 높을 경우 막대의 테두리 색상입니다.
                                        <declineBoxStroke>
                                                <Stroke color="#0000ff"/>
                                        </declineBoxStroke>
                                </Candlestick2DSeries>
                        </series/>
                        <horizontalAxisRenderers>
                                <Axis2DRenderer placement="bottom" axis="{hAxis}"
tickLength="0"/>
                        </horizontalAxisRenderers>
                        <annotationElements>
// syncCrossRangeZoomer : mainChart 의 십자선과 동기화 시킬 subChart 의 CrossRangeZoomer id 를 설정합니다.
                <CrossRangeZoomer id="candleCRZ" enableZooming="false" syncCrossRangeZoomer="{columnCRZ}"
zoomType="both" horizontalLabelFormatter="{nft}"/>
                        </annotationElements>
                </Candlestick2DChart>
        </mainChart>
        <subChart>
                <Column2DChart showDatatips="true" height="20%" paddingTop="0"
paddingBottom="0" gutterTop="4">
                        <horizontalAxis>
                                <CategoryAxis id="hAxis2" categoryField="date"/>
```

```
                                        </horizontalAxis>
                                        <verticalAxis>
                                                <LinearAxis baseAtZero="false"/>
                                        </ verticalAxis>
                                        <series>
                                                <Column2DSeries                                         yField="trdvolume"
itemRenderer="BoxItemRenderer">
                                                        <fill>
                                                                <SolidColor color="#eca614"/>
                                                        </fill>
                                                </Column2DSeries>
                                        </series>
                                        <horizontalAxisRenderers>
                                                <Axis2DRenderer          axis="{hAxis2}"          showLabels="false"
tickLength="0"/>
                                        </horizontalAxisRenderers>
                                        <annotationElements>
// syncCrossRangeZoomer : subChart 의 십자선과 동기화 시킬 mainChart 의 CrossRangeZoomer id 를 설정합니다.
                                                <CrossRangeZoomer      id="columnCRZ"      enableZooming="false"
syncCrossRangeZoomer="{candleCRZ}" horizontalLabelFormatter="{nft}" verticalLabelPlacement="top"/>
                                        </annotationElements>
                                </Column2DChart>
                        </subChart>
                        <dataSelector>
                                <DualScrollBar inverted="true" visibleItemSize="50"/>
                        </dataSelector>
                </DualChart>
</rMateChart>
```

<예제 52 캔들차트 기본 예제 >

<그림 36 캔들차트 공시 표현 화면 >

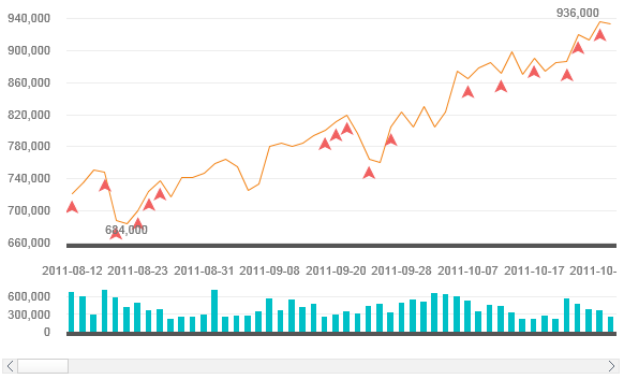
```
<rMateChart backgroundColor="0xFFFFFF"  cornerRadius="12" borderStyle="solid">
        <Options>
            <Caption text="Riamore CandleChart"/>
        </Options>
        // 각종 포메터 정의
        <NumberFormatter id="nft" precision="0"/>
// 한 화면에 두개의 차트를 설정 할 수 있는 DualChart 설정
        <DualChart leftGutterSyncEnable="true" rightGutterSyncEnable="true">
        // mainChart와 subChart의 왼쪽, 오른쪽 여백을 동기화 시킵니다.
            <mainChart>
                <Candlestick2DChart showDataTips="true" paddingBottom="0">
                    <horizontalAxis>
                        <CategoryAxis id="hAxis" categoryField="date"/>
                    </horizontalAxis>
                    <verticalAxis>
                        <LinearAxis baseAtZero="false"/>
                    </ verticalAxis>
                    <series>
// openField : 시가 closeField : 종가 highField : 고가 lowField : 저가
// showMaxValueLabel : 가장 큰 값 수치를 표현합니다.
// showMinValueLabel : 가장 작은 값 수치를 표현합니다.
// symbolField : 공시 데이터를 설정하는 데이터 필드 입니다.
// symbolRenderer : 공시 데이터의 존재유무를 알려주는 렌더러입니다.
                        <Candlestick2DSeries openField="openprc" closeField="closeprc"
highField="high"  lowField="low" showMinValueLabel="true" showMaxValueLabel="true"
maxLabelJsFunction="maxLabelFunc" minLabelJsFunction="minLabelFunc" symbolField="gongsi"
symbolRenderer="UpArrowItemRenderer">
```

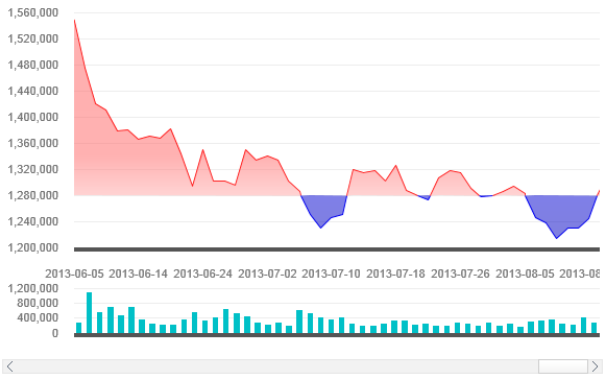
```
                                        // 시가 보다 종가가 높을 경우 막대의 색상 입니다.
                                        <fill>
                                                <SolidColor color="#ff0000" alpha="0.5"/>
                                        </fill>
                                        // 시가 보다 종가가 높을 경우 고가와 저가를 잇는 선의
색상 입니다.
                                        <stroke>
                                                <Stroke color="#ff0000"/>
                                        </stroke>
                                        // 시가 보다 종가가 높을 경우 막대의 테두리 색상입니다.
                                        <boxStroke>
                                                <Stroke color="#ff0000"/>
                                        </boxStroke>
                                        // 종가 보다 시가가 높을 경우 막대의 배경 색상입니다.
                                        <declineFill>
                                                <SolidColor color="#0000ff" alpha="0.5"/>
                                        </declineFill>
                                        // 종가 보다 시가가 높을 경우 고가와 저가를 잇는 선의
색상 입니다..
                                        <declineStroke>
                                                <Stroke color="#0000ff"/>
                                        </declineStroke>
                                        // 종가 보다 시가가 높을 경우 막대의 테두리 색상입니다.
                                        <declineBoxStroke>
                                                <Stroke color="#0000ff"/>
                                        </declineBoxStroke>
                                </Candlestick2DSeries>
                        </series/>
                        <horizontalAxisRenderers>
                                <Axis2DRenderer placement="bottom" axis="{hAxis}"
tickLength="0"/>
                        </horizontalAxisRenderers>
                        <annotationElements>
// syncCrossRangeZoomer : mainChart의 십자선과 동기화 시킬 subChart의 CrossRangeZoomer id를 설정합니다.
                <CrossRangeZoomer id="candleCRZ" enableZooming="false" syncCrossRangeZoomer="{columnCRZ}"
zoomType="both" horizontalLabelFormatter="{nft}"/>
                        </annotationElements>
                </Candlestick2DChart>
        </mainChart>
        <subChart>
                <Column2DChart showDatatips="true" height="20%" paddingTop="0"
paddingBottom="0" gutterTop="4">
                        <horizontalAxis>
                                <CategoryAxis id="hAxis2" categoryField="date"/>
```

```
                                        </horizontalAxis>
                                        <verticalAxis>
                                                <LinearAxis baseAtZero="false"/>
                                        </ verticalAxis>
                                        <series>
                                                <Column2DSeries yField="trdvolume"
itemRenderer="BoxItemRenderer">
                                                        <fill>
                                                                <SolidColor color="#eca614"/>
                                                        </fill>
                                                </Column2DSeries>
                                        </series>
                                        <horizontalAxisRenderers>
                                                <Axis2DRenderer axis="{hAxis2}" showLabels="false"
tickLength="0"/>
                                        </horizontalAxisRenderers>
                                        <annotationElements>
// syncCrossRangeZoomer : subChart의 십자선과 동기화 시킬 mainChart의 CrossRangeZoomer id를 설정합니다.
                                                <CrossRangeZoomer id="columnCRZ" enableZooming="false"
syncCrossRangeZoomer="{candleCRZ}" horizontalLabelFormatter="{nft}" verticalLabelPlacement="top"/>
                                        </annotationElements>
                                </Column2DChart>
                        </subChart>
                        <dataSelector>
                                <DualScrollBar inverted="true" visibleItemSize="50"/>
                        </dataSelector>
                </DualChart>
</rMateChart>
```

<예제 53 캔들차트 공시표현 예제 >

위 캔들차트 기본형외에 아래 그림과 같이 캔들라인차트, 캔들영역차트로 출력을 할 수 있습니다. <Candlestick2D 부분을 <CandleLine2D, <CandleArea2D 로 변경하시고 데이터 필드명만 변경하시면 아래와 같은 차트를 출력 하실 수 있습니다. 더욱 자세한 내용은 샘플페이지와 api 를 참고하십시오





<그림 37 캔들 차트 기본형외에 캔들라인, 캔들영역 차트 >

6.27. 게이지 차트

6.27.1. Circular 게이지

Circular 게이지의 시작과 끝 노드는 <CircularGauge/> 입니다. 0~360 도 범위를 갖는 원형 게이지를 생성하게 되는데 디폴트 0 도의 위치는 시계 3 시 방향이며 시계 방향으로 각도는 진행됩니다.
Circular 게이지는 크게 배경 프레임, 바늘, 바늘 커버 3 개로 구성되어 있습니다. 이 3 개를 각각 디자인함으로써 다양한 모습의 게이지 작성이 가능합니다. 게이지는 기본적으로 디폴트 속성을 갖고 있습니다.
그러나 반드시 사용자가 정의 해야 할 속성은 아래와 같습니다.

- startAngle
- minimumAngle
- maximumAngle
- value
- minimum
- maximum

다음은 Circular 게이지의 속성 및 유효 속성에 대한 설명입니다.

속성명	유효값	설 명
width	Number	게이지의 가로 사이즈 입니다.
height	Number	게이지의 세로 사이즈 입니다.
animationDuration	Number(밀리세컨드)	게이지 애니매이션 지속시간을 설정합니다.(ms 단위)
bounceAnimating	Boolean(기본값:true)	게이지 애니메이션에 바운스 효과를 추가할지 여부를 나타냅니다.
dataTipJsFunction	자바스크립트함수	데이터팁 라벨을 재정의하는 함수입니다.
editMode	Boolean(기본값:false)	게이지의 value 를 마우스로 편집할 수 있게 할지 여부를 나타냅니다.
formatter	Formatter 객체 id	valueDisplay 와 tickLabel, 데이터팁에 표시할 텍스트 포매터입니다.
hideTickLabel	"none","first","last"	차트 구간이 360도인 경우 시작 라벨과 마지막 라벨이 만나게 되는데 이 때 시작 라벨 또는 마지막 라벨 표시 여부를 나타냅니다.

		유효값은 first, last, none 입니다.
interval	Number	게이지의 tick 을 표현할 틱 사이의 간격을 나타냅니다.
labelJsFunction	자바스크립트함수	value 라벨을 재정의하는 함수입니다.
liveDragging	Boolean(기본값:true)	editMode 가 true 인 경우 마우스 드래그로 편집할 수 있게 할지 여부를 나타냅니다 editMode 가 false 인 경우 무시됩니다.
maximum	Number	게이지 라벨에 표시할 최대값입니다.
maximumAngle	0~360의 Number	게이지의 최대 각도입니다.
minimum	Number	게이지 라벨에 표시할 최소값입니다
minimumAngle	0~360의 Number	게이지의 최소각도입니다.
minorInterval	Number	게이지의 minor tick들의 간격을 나타냅니다.
startAngle	Number(Degree)	게이지의 시작각도입니다.
tickLabelJsFunction	자바스크립트함수	major Tick의 라벨을 재정의하는 함수입니다
value	Number	게이지의 현재값 입니다.
enableFilter	Boolean(기본값:true)	게이지에 필터를 활성화 할지 여부를 나타냅니다. The default value is true.
frameFill	SolidColor, RadialGradient 객체	게이지 바탕 프레임의 채우기색입니다.
frameFilter	DropshadowFilter, BevelFilter 객체	프레임에 적용할 필터를 지정합니다.
frameStroke	Stroke 객체	게이지 바탕 프레임의 테두리 선 스타일입니다.
isValueTop	Boolean(기본값:false)	value 라벨을 최상단에 표시할지 여부를 나타냅니다. The default value is false.
labelGap	Number(기본값:4)	틱과 라벨 사이의 여백(gap) 을 지정합니다. The default value is 4.
minorTickFill	SolidColor, RadialGradient 객체	게이지 작은 틱의 채우기색입니다.
minorTickRadius	Number(기본값:2)	게이지 작은 틱의 반지름을 픽셀단위로 지정합니다. The default value is 2.
minorTickStroke	Stroke 객체	게이지 작은 틱의 선 스타일입니다.
needleBackLengthRatio	Number(기본값:0)	게이지 바늘 커버 뒤쪽으로 바늘의 하단부를 표시하는 비율입니다. The default value is 0.
needleCoverFill	SolidColor, RadialGradient 객체	게이지 바늘 커버의 채우기색입니다.
needleCoverFilter	DropshadowFilter, BevelFilter 객체	바늘 커버에 적용할 필터를 지정합니다.
needleCoverRadius	Number(기본값:15)	게이지 바늘 커버의 반지름을 픽셀 단위로

		지정합니다. The default value is 15;.
needleCoverStroke	Stroke 객체	게이지 바늘 커버의 선 스타일입니다.
needleFill	SolidColor, RadialGradient 객체	게이지 바늘의 채우기색입니다.
needleFilter	DropshadowFilter, BevelFilter 객체	바늘에 적용할 필터를 지정합니다.
needleLengthRatio	Number(기본값:0.85)	게이지 바늘의 길이를 나타냅니다. 이는 게이지 반지름의 비율입니다. 예를 들어 1로 설정 시 게이지 반지름 크기의 바늘이 생성됩니다. The default value is 0.85.
needlePointStyle	"steeple","round"	게이지 바늘의 모양입니다. 뽀족한 모양과 둥근 모양으로 설정할 수 있습니다. 유효값은 steeple, round 입니다. The default value is steeple.
needleStroke	Stroke 객체	게이지 바늘의 테두리 선 스타일입니다.
needleThickness	Number(기본값:10)	게이지 바늘의 두께를 나타냅니다.(픽셀 단위) The default value is 10.
padding	Number(기본값:0)	게이지의 상하좌우 여백입니다. The default value is 0.
pointThickness	Number(기본값:0)	게이지 바늘 끝의 두께를 나타냅니다. The default value is 0.
showDataTip	Boolean(기본값:true)	게이지의 바늘에 마우스 오버 시 게이지값을 표시하는 팁을 표시할지 여부를 나타냅니다. The default value is true.
showTickLabels	Boolean(기본값:false)	게이지 틱 라벨의 표시 여부를 나타냅니다. The default value is true.
showTrackColor	Boolean(기본값:false)	게이지의 minimum, maximum 사이의 범위에 따라 게이지 트랙을 설정할지 여부를 나타냅니다. The default value is false
showValueLabel	Boolean(기본값:true)	게이지 value 라벨의 표시 여부를 나타냅니다. The default value is true.
tickFill	SolidColor, RadialGradient 객체	게이지 틱의 채우기색입니다.
tickFilter	DropshadowFilter, BevelFilter 객체	틱에 적용할 필터를 지정합니다.
tickGap	Number(기본값:4)	게이지 프레임 테두리와 tick 사이에 여백을 지정합니다. The default value is 4.
tickLabelPlacement	"inside","outside"	틱 라벨의 위치를 틱을 기준으로 안쪽에 위치

	(기본값:inside)	할지 바깥쪽에 위치할지를 지정합니다. 유효값은 inside, outside 입니다. The default value is inside.
tickLabelStyleName	스타일 이름	tick 라벨의 스타일 이름입니다.
tickRadius	Number(기본값:4)	게이지 틱의 반지름을 픽셀단위로 지정합니다. The default value is 4.
tickStroke	Stroke 객체	게이지 틱의 선 스타일입니다.
trackAlphas	0~1 요소의 배열	minimum~maximum 구간에 따라 따라 표현하는 컬러 트랙의 채우기 색 투명도를 지정합니다.
trackColors	RGB 요소의 배열	minimum~maximum 구간에 따라 따라 표현하는 컬러 트랙의 범위를 지정합니다. 예를들어, trackColors=[#FFFF00, #0000FF, #FF0000] 이면, trackValue의 범위에 맞게 색이 채워집니다.
trackInnerRadius	0~1 사이의 Number	minimum~maximum 구간에 따라 따라 표현하는 컬러 트랙의 안쪽 표현 범위를 나타냅니다. 유효값은 0~1 이며, 0 인 경우 게이지 정중앙을 나타내고, 1인 경우 게이지 테두리 전까지를 나타냅니다. The default value is 0.
trackOuterRadius	0~1 사이의 Number	minimum~maximum 구간에 따라 따라 표현하는 컬러 트랙의 바깥쪽 표현 범위를 나타냅니다. 유효값은 0~1 이며, 0 인 경우 게이지 정중앙을 나타내고, 1인 경우 게이지 테두리 전까지를 나타냅니다. The default value is 1.
trackValues	배열	minimum~maximum 구간에 따라 따라 표현하는 컬러 트랙의 범위를 지정합니다. 예를들어, minimum=0, maximnum=100 일 경우, trackValues=[0,50,80,100] 이면, 0~50, 50~80, 80~100까지 색상을 지정할 수 있습니다.
valueLabelStyleName	스타일이름	value 라벨의 스타일 이름입니다.
valueXOffset	Number	value 라벨에 추가 X Offset 을 지정하여 라벨을 이동시킵니다. The default value is 0.
valueYOffset	Number	value 라벨에 추가 Y Offset 을 지정하여 라벨을 이동시킵니다. The default value is 0.

<표 37 Circular 게이지 속성 및 유효값 설명 표>

위 속성들을 토대로 레이아웃을 작성해 보도록 하겠습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xEEEEEE" cornerRadius="12" borderStyle="solid">
	<Options>
		<Caption text="게이지 예제 - Red"/>
		<SubCaption text="게이지, 색상, 그라데이션 등을 변경 할 수 있습니다." textAlign="right"
paddingRight="10" fontSize="11" />
	</Options>
	<CurrencyFormatter id="numFmt" precision="0" currencySymbol="%" alignSymbol="right"/>
	<CircularGauge width="300" height="300"  valueChangeFunction="valueChangeFunc"
				startAngle="270" minimumAngle="0" maximumAngle="360"
				minimum="0" maximum="100" value="28"
				interval="10" minorInterval="2"
				formatter="{numFmt}"
				padding="10"
				labelGap="10"
				tickLabelStyleName="tickText"
				valueLabelStyleName="valueText"
				editMode="true" liveDragging="true"
				showDataTip="true"
				tickColor="0xCCCCCC"
				minorTickColor="0x932108"
				coverRadiusRatio="0.1"
				hideTickLabel="first"
				needleThickness="10"
				pointThickness="5"
				needleLengthRatio="0.6"
				needlePointStyle="steeple"
				needleBackLengthRatio="0"
				isValueTop="false"
				valueYOffset="50">
			<frameStroke>
				<Stroke color="0xCCCCCC" weight="10"/>
			</frameStroke>
			<frameFill>
				<SolidColor color="0x932108"/>
			</frameFill>
			<needleFill>
				<LinearGradient angle="90">
					<entries>
						<GradientEntry color="0xEEEEEE" ratio="0" alpha="1"/>
						<GradientEntry color="0x555555" ratio="1" alpha="1"/>
					</entries>
				</LinearGradient>
			</needleFill>
```

```
                        <needleCoverFill>
                                <RadialGradient>
                                        <entries>
                                                <GradientEntry color="0xFFFFFF" ratio="0" alpha="1"/>
                                                <GradientEntry color="0xB0B0B0" ratio="1" alpha="1"/>
                                        </entries>
                                </RadialGradient>
                        </needleCoverFill>
                        <needleCoverFilter>
                                <DropShadowFilter distance="1"/>
                        </needleCoverFilter>
        </CircularGauge>
        <Style>
                .valueText
                {
                        fontSize:12;
                        fontFamily:Myriad;
                        textAlign:center;
                        borderColor:0x999999;
                        backgroundColor:0xFFFFFF;
                        backgroundAlpha:1;
                        paddingTop:2;
                        borderThickness:1;
                        borderAlpha:1;
                        borderStyle:solid;
                        color:0xFF0000;
                }
                .tickText
                {
                        fontFamily:Myriad;
                        fontSize:18;
                        color:0xFFFFFF;
                }
        </Style>
</rMateChart>
```

<예제 54 Circular 게이지 레이아웃 예제>

위 레이아웃을 실행시킨 화면은 아래와 같습니다.

<그림 38 Circular 게이지 실행 화면>

6.27.2. Half-Circular 게이지

Half-Circular 게이지는 <HalfCircularGauge/> 노드로 시작과 끝을 정의 합니다.
Half-Circular 게이지에서 시작 각도(startAngle) 은 무효한 속성입니다.
minimumAngle 은 초기값이 180 도입니다. 즉, Half-Circular 게이지의 특성 상 9 시 방향이 초기점이 됩니다. 9 시방향은 180 도에 해당됩니다. 만약 180 도보다 작은 각도를 설정시 Half 게이지는 180 도로 강제 설정하게 됩니다. 또한 maximumAngle 은 360 도보다 커질 수 없습니다.
따라서 Half-Circular 게이지가 표현할 수 있는 범위는 9 시 방향에서 3 시 방향 사이입니다.
그 외 모든 속성 및 스타일은 Circular 게이지와 같습니다. "6.27.1 Circular 게이지" 를 참고하여 주십시오.
Half-Circular 게이지의 고유 속성은 다음과 같습니다.

속성명	유효값	설 명
bottomRadius	Number	반원형 게이지 밑단의 중앙 반지름을 픽셀단위로 지정합니다. frameType 을 rouding 으로 설정 시 밑단 중앙의 반지름이 되며, flat 으로 설정 시 bottomRadius 만큼의 여백을 갖습니다. The default value is 0.
frameType	"flat", "rounding"	반원형 게이지의 밑단의 중앙을 둥굴게 처리할지 평평하게 처리할지를 나타냅니다. The default value is 'flat'

<표 38 Half-Circular 게이지 속성 및 유효값 설명>

6.27.3. Cylinder 게이지

실린더 게이지는 수직형태와 수평 형태가 존재합니다. 수직 형태의 실린더 게이지를 생성하기 위해서는 <HCylinderGauge/> 노드로 시작과 끝을 결정하고, 수평 형태의 실린더 게이지를 생성하기 위해서는 <HCylinderGauge/> 노드로 시작과 끝을 정의합니다.
다음 표는 실린더 게이지의 속성 및 유효값 설명입니다.

속성명	유효값	설 명
width	Number	실린더 게이지의 가로 사이즈입니다.
height	Number	실린더 게이지의 세로 사이즈입니다.
minimum	Number	실린더 게이지가 표현할 최소값입니다.
maximum	Number	실린더 게이지가 표현할 최대값입니다.
labels	String의 집합	tick 라벨에 표현할 텍스트의 집합입니다.
showValueLabel	true/false(기본값:true)	Value 라벨을 표시할지 여부를 나타냅니다.
value	Number	현재 입력된 값(value) 입니다.
targetMark	Number	목표치에 해당되는 값입니다.
targetMarkColor	Array (기본값 : [0xeaeaea,0xeaeaea, 0xeaeaea])	실린더 바깥모양의 색상입니다
targetMarkAlpha	Array (기본값 : [0.6,0.6,0.7])	실린더 바깥모양의 투명도입니다
targetMarkRatio	Array (기본값 : [0,125,255])	실린더 바깥모양의 그리기 영역입니다
cylinderColor	Array[0xdb88db,0xffffff,0xdb88db]	안쪽 실린더의 색을 결정합니다.
innerTipJsFunction	Object (기본값 : null)	안쪽 게이지의 innerTipFunction입니다. 안쪽 게이지의 팁을 원하는 형태로 출력 하고 싶으시다면 정의하십시오
labelJsFunction	Object (기본값 : null)	실린더의 값을 표시하는 텍스트(valueLable)의 형식을 설정하는 콜백 함수입니다. 이 함수는, 단일의 수치를 인수로 받아, 형식을 설정한 string형태로 반환합니다
targetTipJsFunction	Object (기본값 : null)	목표 라인의 targetTipFunction입니다. 목표 라인의 팁을 원하는 형태로 출력 하고 싶으시다면 정의하십시오
cylinderColumnWidth	0~1(기본값:1)	안쪽 실린더의 가로 비율입니다. 이값은 전체 실린더의 퍼센티지로 할당됩니다. 1인 경우 100%, 0인 경우 0 퍼센트.

tickLabelJsFunction	Object (기본값 : null)	labels에 넣어놓은 라벨들을 사용자가 원하는 형태로 바꾸고 싶다면 정의하십시오
snapInterval	Number	실린더 썸의 이동 간격을 나타냅니다. 예를 들어 minimum=0, maximum=10 인 경우 snapInterval=2로 설정한 경우 썸 이동 간격은 2씩 증가합니다.
tickThickness	Number(기본값:1)	tick 의 두께를 나타냅니다.
tickLength	Number(기본값:3)	tick 의 길이를 나타냅니다.
tickInterval	Number	tick 간격을 나타냅니다.
liveDragging	true | false	thumb의 라이브드래깅을 value 에 반영할지를 나타냅니다. true 인 경우 마우스 snapInterval 간격으로 value에 반영하고, false 인 경우 마우스를 클릭이 완전히 이루어진 경우에만 반영됩니다.
tickColor	RGB(기본값:0x6F7777)	tick 의 색깔을 결정합니다.
valueLabelXOffset	Number(기본값:0)	값을 나타내는 라벨의 X Offset 입니다. 라벨의 X는 기본적으로 게이지 가운데 위치합니다.
valueLabelYOffset	Number(기본값:0)	값을 나타내는 라벨의 Y Offset 입니다. 라벨의 Y는 기본적으로 게이지 가운데 위치합니다.
valueLabelStyleName	String (기본값 : no default)	게이지 값이 보일 텍스트의 스타일 네임입니다
offsetOfthumb	Number (기본값 : 6)	thumb와 게이지의 여백입니다.
dataTipFontSize	Number (기본값 : 10)	데이터 팁 폰트 사이즈 입니다.
showDataTip	Boolean (기본값 : true)	데이터팁 보이기 여부입니다
thumbVisible	Boolean (기본값 : false)	LinearGauge의 thumb숨기기 입니다
padding	Number (기본값 : 0)	게이지의 여백입니다.

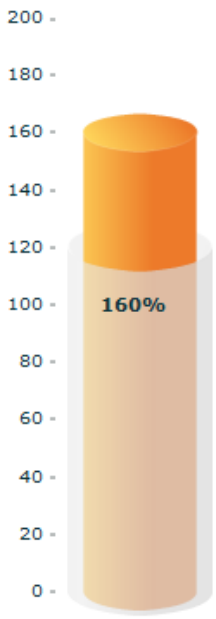
<표 39 실린더 게이지 속성 설명표>

다은은 수직 실린더 게이지의 샘플 레이아웃을 작성한 모습과 실행화면입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart   backgroundColor="0xFFFFFF"   cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="게이지"/>
        </Options>
        <VCylinderGauge id="cy1" width="30%" height="180"
                                minimum="0" maximum="160"
                                labels="[0, 20, 40, 60, 80, 100, 120, 140, 160]"
                                value="50"
```

```
                                targetMark="150"
                                snapInterval="1"
                                tickThickness="2"
                                tickLength="2"
                                tickInterval="20"
                                liveDragging="true"
                                tickColor="0x000000"
                                valueLabelXOffset="10"
                                valueLabelStylename="valueLabel"
        />
        <Style>
            .valuelabel{
                fontWeight:bold;
            }
        </Style>
</rMateChart>
```

<예제 55 VCylinderGauge 레이아웃 예제>



<그림 39 VCylinderGauge 실행화면>

6.27.4. Linear 게이지

리니어 게이지는 세로 방향의 막대 모양과 가로 방향의 막대 모양으로 나뉩니다. 세로 방향은 <VLinearGauge/> 로 시작과 끝을 정의하고, 가로 방향은 <HLinearGauge/> 노드로 정의합니다.

리니어 게이지는 크게 값(value) 를 표현하는 막대와 목표치를 나타내는 목표선(target line)으로 구성되어 있습니다.

다음은 리니어 게이지의 속성 및 유효값에 대한 설명입니다.

속성명	유효값	설 명
width	Number	리니어 게이지의 가로 사이즈입니다.
height	Number	리니어 게이지의 세로 사이즈입니다.
minimum	Number	리니어 게이지가 표현할 최소값입니다.
maximum	Number	리니어 게이지가 표현할 최대값입니다.
labels	String의 집합	tick 라벨에 표현할 텍스트의 집합입니다.
showValueLabel	true/false(기본값:true)	Value 라벨을 표시할지 여부를 나타냅니다.
value	Number	현재 입력된 값(value) 입니다.
targetMark	Number	목표치에 해당되는 값입니다.
targetMarkLength	Number(기본값:10)	목표치에 해당되는 선의 길이를 나타냅니다.
targetMarkAlpha	Array [1,1,1]	목표치에 해당되는 선의 투명도를 나타냅니다.
targetMarkColor	Array (기본값 : [0xff0000,0xff0000,0xff0000])	목표치에 해당되는 선의 색상을 나타냅니다.
targetMarkRatio	Array (기본값 : [255,255,255])	목표치를 나타내는 선의 그리기 영역입니다
targetMarkThickness	Number(기본값:3)	목표치에 해당되는 선의 두께를 나타냅니다.
targetMarkBorderColor	Uint(RGB색상)	목표치의 테두리 색상을 지정합니다.
targetMarkBorderAlpha	Number(0~1)	목표치의 테두리 투명도를 지정합니다.
linearColor	Array (기본값: [0xff66ff, 0xff66ff, 0xff66ff])	실적치의 채우기색을 결정합니다.
linearAlpha	Array (기본값: [1,1,1,])	실적치의 채우기색 투명도를 나타냅니다.
linearRatio	Array (기본값: [255,255,255])	실적치의 채우기색 비율입니다
linearBorderColor	Uint(RGB색상)	실적치의 테두리 색상을 지정합니다.
linearBorderAlpha	Number(0~1)	실적치의의 테두리 투명도를 지정합니다.
linearBgColor	Array (기본값: [0xEAEAEA,0xFFFFFF,0xEAEAEA])	게이지의 배경색을 나타냅니다.
linearBgAlpha	Array (기본값 : [1,1,1])	게이지 배경색의 투명값입니다.
linearBgRatio	Array (기본값 : [255,255,255])	게이지 배경색의 그리기 영역입니다
linearBgBorderColor	Uint(RGB색상)	게이지 배경색의 테두리 색상을 지정합니다.
linearBgBorderAlpha	Number(0~1)	게이지 배경색의 테두리 투명도를 지정합니다
linearColumnWidth	0~1(기본값:1)	안쪽 막대의 가로 비율입니다. 이값은 전체 게이지가로의 퍼센티지로 할당됩니다. 1인 경우 100%, 0인 경우 0 퍼센트.
snapInterval	Number	실린더 썸의 이동 간격을 나타냅니다. 예를 들어 minimum=0, maximum=10 인 경우

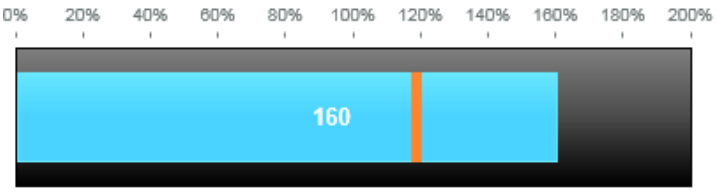
		snapInterval=2로 설정한 경우 썸 이동 간격은 2씩 증가합니다.
tickThickness	Number(기본값:1)	tick 의 두께를 나타냅니다.
tickLength	Number(기본값:3)	tick 의 길이를 나타냅니다.
tickInterval	Number	tick 간격을 나타냅니다.
liveDragging	true|false	썸의 라이브드래깅을 value 에 반영할지를 나타냅니다. true 인 경우 마우스 snapInterval 간격으로 value에 반영하고, false 인 경우 마우스를 클릭이 완전히 이루어진 경우에만 반영됩니다.
tickColor	RGB(기본값:0x6F7777)	tick 의 색깔을 결정합니다.
valueLabelXOffset	Number(기본값:0)	값을 나타내는 라벨의 X Offset 입니다. 라벨의 X는 기본적으로 게이지 가운데 위치합니다.
valueLabelYOffset	Number(기본값:0)	값을 나타내는 라벨의 Y Offset 입니다. 라벨의 Y는 기본적으로 게이지 가운데 위치합니다.
valueLabelStyleName	String (기본값 : No Default)	게이지 값이 보일 텍스트의 스타일 네임입니다
offsetOfthumb	Number (기본값 : 6)	thumb와 게이지의 여백입니다.
dataTipFontSize	Number (기본값 : 10)	데이터 팁 폰트 사이즈 입니다.
showDataTip	Boolean (기본값 : true)	데이터팁 보이기 여부입니다
thumbVisible	Boolean (기본값 : false)	LinearGauge의 thumb숨기기 입니다
Padding	Number (기본값 : 0)	게이지의 여백입니다.
showTargetMark	Boolean (기본값 : true)	목표치 기둥 보이기 여부입니다
innerTipJsFunction	Object (기본값 : null)	안쪽 게이지의 innerTipFunction입니다. 안쪽 게이지의 팁을 원하는 형태로 출력 하고 싶으시다면 정의하십시오
labelJsFunction	Object (기본값 : null)	실린더의 값을 표시하는 텍스트(valueLable)의 형식을 설정하는 콜백 함수입니다. 이 함수는, 단일의 수치를 인수로 받아, 형식을 설정한 string형태로 반환합니다
innerTipJsFunction	Object (기본값 : null)	목표 라인의 targetTipFunction입니다. 목표 라인의 팁을 원하는 형태로 출력 하고 싶으시다면 정의 하십시오
tickLabelJsFunction	Object (기본값 : null)	labels에 넣어놓은 라벨들을 사용자가 원하는 형태로 바꾸고 싶다면 정의하십시오

<표 40 리니어 게이지 속성 설명 표>

다음은 수평 리니어 게이지의 샘플 레이아웃과 실행 화면입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFFF"  cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="게이지"/>
        </Options>
        <HLinearGauge id="hl" width="300" height="100"
                                minimum="0" maximum="100"
                                liveDragging="true"
                                labels="[0, 25, 50, 75, 100]"
                                value="50"
                                targetMark="90"
                                tickThickness="1"
                                tickLength="4"
                                tickInterval="5"
                                snapInterval="1"
                                linearColumnWidth=".6"
                                tickColor="#000000"
                                valueLabelFontWeight="bold"
        />
        <Style>
            .valuelabel{
                fontWeight:bold;
             }
        </Style>
</rMateChart>
```

<예제 56 Horizontal Linear Gauge 레이아웃 예제>



<그림 40 Horizontal Linear Gauge 실행 화면>

7. 차트의 부가기능 사용하기.

7.1. 사이드바 사용하기

알메이트 차트에 마우스 포인트가 활성화되면 차트의 우측 상단에 사이드바가 표시됩니다. 이 사이드바는 자주 사용하는 메뉴들을 모아놓은 것입니다.
사이드바는 차트컴포넌트(예:rMateChart.swf, rMateRadarChart.swf 등)에 내장된 기능이 아닙니다. 사이드바는 함께 제공된 ChartSideBar.swf 입니다. 따라서 사이드바를 사용하기 위해서는 다음과 같은 절차를 밟아야 합니다.

- 첫째, 차트 컴포넌트(예:rMateChart.swf 등)이 위치한 곳에 ChartSideBar.swf 가 위치해야 합니다.
  즉, 차트 컴포넌트와 사이드바 컴포넌트는 같은 디렉토리상에 위치해야 합니다.
- 둘째, flashVars 설정 시 useSideBar=true 로 설정합니다.
  true 로 설정 시 차트컴포넌트에서 ChartSideBar.swf 를 로딩하여 차트에 적용시킵니다.
  (기본값은 false 입니다. useSidBar=false 인 경우 차트는 사이드바를 로딩하지 않습니다.)

사이드바에 설정할 수 있는 메뉴 아이템은 다음과 같습니다.

1. 마우스 이동에 따른 십자가 표시 기능 ON/OFF 토글 – 레이아웃에서 구체적인 속성 必
2. 데이터 에디터 표시 ON/OFF 토글 – 데이터 에디터 표시 ON/OFF 토글은 flashVars 에서 데이터 에디터를 사용으로 설정한 경우에만 사용할 수 있는 기능입니다. 즉, 데이터에디터 사용을 true 로 설정하지 않은 경우 이 메뉴 아이템은 사용 불가입니다. 이 메뉴 아이템을 사용하고자 하는 경우 데이터 에디터 사용이 먼저 선행되어야 합니다.(“5.4.3. 데이터 에디터 사용하기” 참고)
3. 그리기 도구 사용 ON/OFF 토글
4. 확대/축소 기능 ON/OFF 토글 – 레이아웃에서 구체적인 속성 必
5. 차트 이미지로 저장하기
6. 차트 인쇄하기

이 사이드바의 표시 여부와 사이드 바 메뉴 아이템 설정은 전적으로 차트 생성 시 정의 하는 flashVars 에서 이루어 집니다. 이에 대한 자세한 설명은 “4.1. flashVars 설정” 을 참고하여 주십시오.

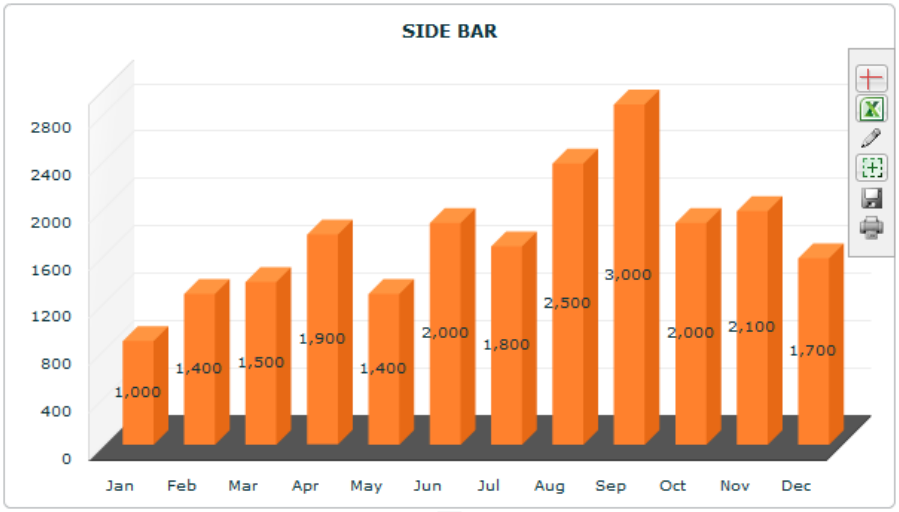
마우스 이동에 따른 십자가 표시와 확대/축소 기능은 반드시 레이아웃에서 해당 기능을 정의하여야 정상적인 기능을 발휘합니다. 사이드 바는 십자가 표시와 확대 축소 기능으로 접근하는 메뉴만 정의 한 것이지 해당 기능을 정의한 것이 아닙니다. 해당 기능은 레이아웃에서 정의하여야 합니다. 이에 대한 설명은 “확대/축소와 마우스 이동에 따른 십자가 표시하기” 를 참고하여 주십시오.

다음은 사이드 바를 표시하고 사이드 바에 표시할 수 있는 모든 사이드바 메뉴 아이템을 설정한 예제 및 결과 모습입니다.

```
<script src="./JS/AC_OETags.js" language="javascript"> </script>
<script src="./JS/rMateChart.js" language="javascript"> </script>
<script src="./JS/rMateChartLicense.js" language="javascript"> </script>
```

```
<link rel="StyleSheet" href="cssStyle.css" type="text/css" />

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">
//-----------------------1번 차트 flashVars 설정 시작----------------------
//차트 레이아웃 URL 경로. 반드시 설정하십시오.
var layoutURL = encodeURIComponent("./LayoutXml/SideBar_EditItem.xml");
var flashVars = "layoutURL="+layoutURL;

//데이타를 URL 경로를 통해 가져올 경우 설정하십시오.
//배열형태로 데이터를 삽입할 경우 주석처리나 삭제하십시오.
//배열형태와 같이 사용할 경우, 우선순위는 배열형태 데이터 삽입입니다.
var dataURL =encodeURIComponent("./DataXml/multiData2.xml");
flashVars += "&dataURL="+dataURL;

//사이드 바를 사용으로 설정(default:false)
var flashVars = "&useSideBar=true";

//사이드 바에 데이타 에디터 메뉴 추가.(defalut:false)
flashVars += "&addEditorToSide=false";

//사이드 바에 십자가 메뉴 추가(default:false)
flashVars += "&addCrossHairToSide=true";

//사이드 바에 줌 메뉴 추가(default:false)
flashVars += "&addZoomToSide=true";

//사이드 바에 드로잉 툴 메뉴 추가(default:true)
flashVars += "&addDrawingToSide=true";

//사이드 바에 차트 이미지 저장 메뉴 숨기기(default:true)
flashVars += "&addSaveAsImgToSide=true";

//사이드 바에 차트 인쇄 메뉴 숨기기(default:true)
flashVars += "&addPrintToSide=true";
<!-- 사용자 정의 설정 끝  -->
</script>

.....
```

<예제 57 사이드바 적용 예제>



<그림 41 사이드바 적용 모습>

7.2. 슬라이드 차트 기능 사용하기

슬라이드 차트 기능은 여러 페이지의 차트를 생성하여 프리젠테이션 형식과 같이 차트를 보여주는 기능입니다.

원활한 슬라이드 차트 기능을 사용하기 위해서는 rMateIntegration.swf 컴포넌트를 사용하여 주십시오. 예를 들어 칼럼 3D 차트, 레이다 차트, 이미지 차트 를 각각의 페이지로 구성된 슬라이드를 만들 경우 다른 차트 컴포넌트로는 생성이 불가능합니다. 통합버전인 rMateIntegration.swf 컴포넌트는 이 모두를 가능케 합니다.

슬라이드 차트에 삽입가능한 차트에는 제한이 없습니다. 알메이트 차트에서 작성 가능한 차트는 모두 슬라이드 차트로 표현 가능합니다.
슬라이드 차트는 반드시 다음 규칙을 따라야 합니다.

1. 레이아웃은 setSlideLayoutSet() 함수로 삽입해야 합니다.
2. 데이터는 setSlideDataSet() 함수로 삽입해야 합니다.

위 규칙대로 레이아웃과 데이터를 삽입하면 슬라이드 차트가 생성됩니다.
예를 들어 총 5 페이지의 차트 슬라이드를 구성한다면 데이터와 레이아웃은 각각 5 개의 쌍으로 이루어져야 합니다. 따라서 레이아웃과 데이터는 배열 형태가 되며 배열 요소는 각각 스트링 형태의 레이아웃과 배열 형태의 데이터가 됩니다.

다음은 그에 대한 예제입니다.

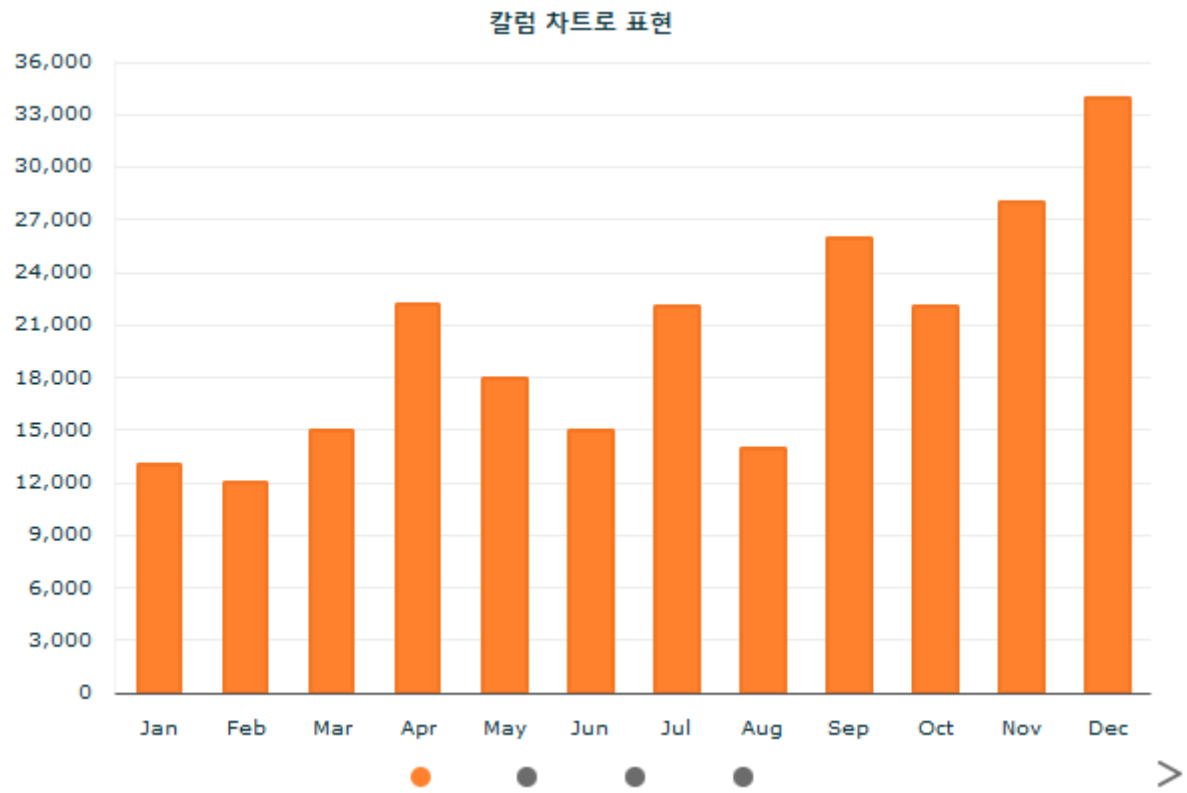
```
<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">


// ------------------------차트  flashVars 설정 시작-----------------------
// rMate 차트와 스크립트 간의 동기화가 완료된 후 호출되는 함수를 지시하여
// 이 함수를 통하여 레이아웃과 데이터를 차트에 입력합니다.
// 따라서 레이아웃과 데이터의 XML 경로를 입력하지 않습니다.

var flashVars = "";

//동기화가 완료 돼었을 때 호출할 함수를 정의하십시오.

var rMateOnLoadCallFunction = "rMateChartOnLoad";
flashVars += "rMateOnLoadCallFunction="+rMateOnLoadCallFunction;

// ------------------------차트  flashVars 설정 끝-----------------------


//동기화가 완료 돼었을 때 호출할
function rMateChartOnLoad()
{
        var layout1 = getCartesianLayout("Column2D","칼럼 차트로 표현",["Profit"]);
        var layout2 = getCartesianLayout("Line2D","라인 차트로 표현",["Profit"]);
        var layout3 = getCartesianLayout("Column3D","칼럼 3D 멀티 데이터 표현",["Profit","Cost"]);
```

- 매뉴얼은 같은 레이아웃과 같은 데이터로 슬라이드 차트를 표현하고 있습니다.

```
        // 슬라이드에 넣을 데이터와 레이아웃들.
        layoutSet = [layout1, layout2, layout3, layout1, layout2];
        dataSet = [chartData, chartData,chartData, chartData, chartData];

        // 슬라이드에서 표현할 레이아웃들 삽입.
        document.getElementById("chart").setSlideLayoutSet(layoutSet);

        // 슬라이드에서 표현할 데이터들 삽입.
        document.getElementById("chart").setSlideDataSet(dataSet);
}



//레이아웃을 반환하는 함수를 작성한 것입니다.
//해당 레이아웃은 스트링 형태의 조합이므로 개발자에 의해 다음과 같은 함수 작성이 가능합니다.
```

```
//파라메터 설명
//type : 차트 type
//title : 차트 Caption
//dataField : 시리즈가 표현할 실데이터의 필드명 배열
function getCartesianLayout(type, title, dataField)
{
	var layout="<rMateChart borderStyle='solid'>"
				+"<Options><Caption text='" + title +"'/></Options>"
				+"<" + type + "Chart showDataTips='true'>"
				+"<series>";

		for(var i=0; i<dataField.length; ++i)
		{
			layout += "<" + type +"Series yField='" + dataField[i] + "' displayName='" + dataField[i]
+ "'/>"
		}

		layout +="</series>"
				+"<horizontalAxis>"
				+	"<CategoryAxis categoryField='Month'/>"
				+"</horizontalAxis>"
				+"</" + type + "Chart>"
				+"</rMateChart>";
	return layout;
}
//배열 데이터로 정의
var chartData = [{"Month":"Jan", "Profit":13000},
	{"Month":"Feb", "Profit":12000},
	{"Month":"Mar", "Profit":15000},
	{"Month":"Apr", "Profit":22200},
	{"Month":"May", "Profit":18000},
	{"Month":"Jun", "Profit":15000},
	{"Month":"Jul", "Profit":22000},
	{"Month":"Aug", "Profit":14000},
	{"Month":"Sep", "Profit":26000},
	{"Month":"Oct", "Profit":22000},
	{"Month":"Nov", "Profit":28000},
	{"Month":"Dec", "Profit":34000}];
<!-- 사용자 정의 설정 끝  -->
```

<예제 58 슬라이드 차트 작성 예제>



<그림 42 슬라이드 차트 출력 모습>

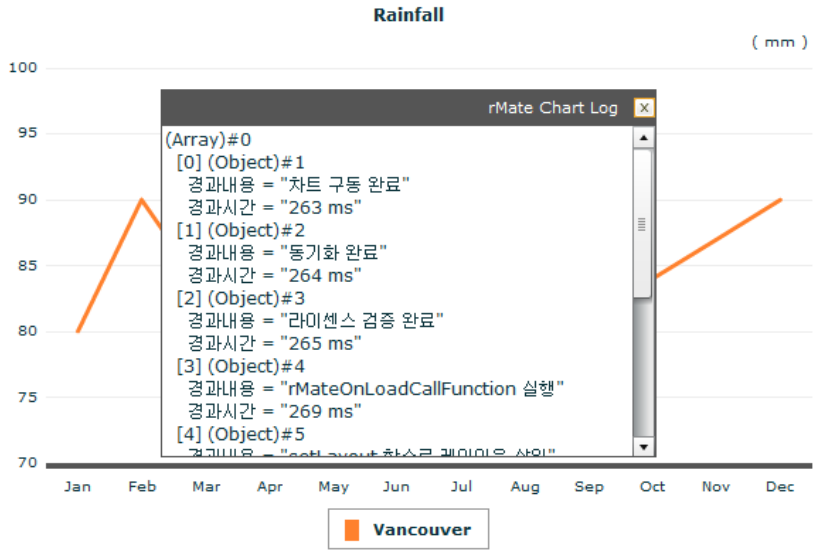
7.3. 디버그 모드 사용하기.

디버그 모드는 플래시 플레이어에 의해 알메이트 차트가 인지되는 시점부터 데이터가 정상적으로 차트에 출력되는 종점까지 시간을 체크하여 로그로 남긴 것을 볼 수 있는 모드입니다.
디버그 모드를 사용하는 방법은 간단합니다. flashVars 설정 시 debugMode=true 로 설정하여 주십시오.
Html 페이지에서 flashVars 를 정의 시 다음과 같이 debugMode=true 를 설정합니다.

```
//-----------------------------------------------------
// flashVars
//-----------------------------------------------------

var flashVars = "layoutURL="+layoutURL;

flashVars += "&dataURL="+dataURL;

// debug Mode 설정
flashVars += "&debugMode=true";
```

디버그 모드는 알메이트 차트 개발자를 위한 것입니다. 개발이 완료된 후에는 모두 제거하시길 권장합니다.

위처럼 debugMode=true 로 설정한 후 차트에서 마우스 오른쪽 버튼을 눌러 플래시 메뉴를 확인하십시오. Chart Log 라는 메뉴가 생성된 것을 볼 수 있습니다. 이를 클릭하면 차트 로그를 확인 할 수 있습니다.

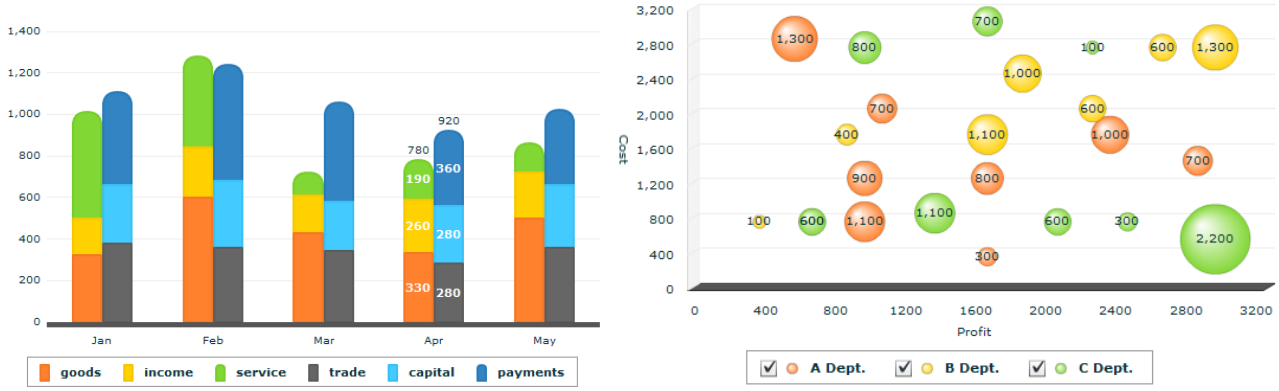


<그림 43 디버그 모드 설정 모습>

8. 고급 사용자를 위한 rMate Chart 레이아웃 설정.

8.1. 차트에 수치필드 표시하기

차트 데이터에 해당되는 수치를 아래 그림과 같이 차트에 표시할 수 있습니다.



이는 차트의 시리즈 속성으로 각각의 시리즈에 대한 속성은 labelPosition 입니다. 시리즈에 따라 labelPosition 의 유효값이 다릅니다. 아래 표는 labelPosition 의 유효값에 대한 설명입니다.

대표 시리즈 명	유효값	설 명
ColumnSeries(2D, 3D), BarSeries(2D, 3D), ImageSeries	inside, outside, both, none	수치필드를 안쪽에 표시할지 바깥쪽에 표시할지 또는 양쪽에 표시할지를 지정합니다.
Line2DSeries, Area2DSeries	up, down, both, none	수치필드를 데이터 포인트를 기준으로 위에 표시할지 아래에 표시할지 또는 양쪽에 표시할지를 지정합니다.
Bubble3DSeries	inside,none	버블 안쪽에 수치필드를 표시할지 여부를 나타냅니다.
PieSeries(2D,3D)	inside, outside, callout, insideWithCallout	파이 안쪽, 바깥쪽 또는 선을 긋고 바깥쪽에 표시할지를 지정합니다. insideWithCallout 을 설정한 경우 파이의 크기가 충분히 커 라벨이 표시될 공간이 있는 경우 inside 에 공간이 부족한 경우 callout 형태로 출력합니다.

<표 41 시리즈 별 labelPosition 유효 값 및 설명>

8.2. 차트 전체 배경에 컬러 지정하기.

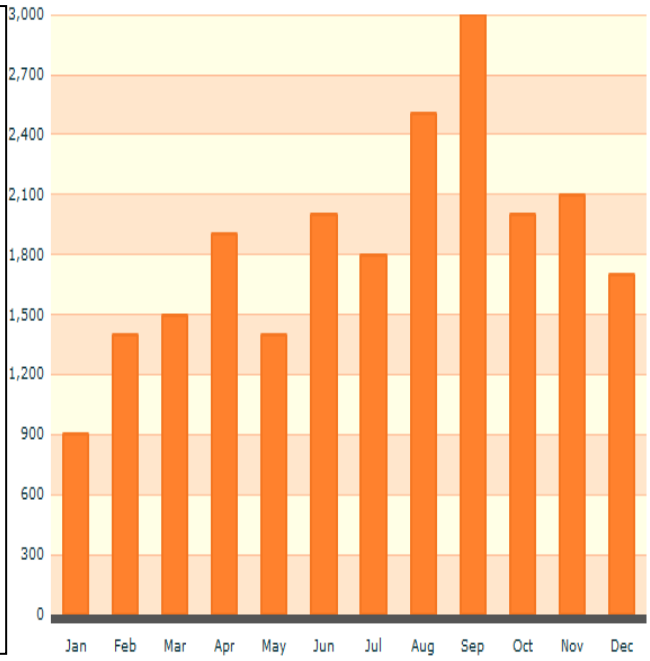
차트 전체에 컬러를 지정하는 방법은 SolidColor, LinearGradient, RadialGradient 3 가지가 있습니다. SolidColor 는 단색을 표현하고, LinearGradient 와 RadialGradient 는 두 가지 이상의 색을 이용하여 그라데이션 효과를 표현합니다. 두 그라데이션 표현의 차이는 선형과 원형이라 볼 수 있습니다.

다음 예제와 샘플을 참고하십시오.

8.2.1. 일반적인 컬러(단색) 지정하기.

```
<Column3DChart>

   <fill>
      <SolidColor color="0xFFFF00"
         alpha="0.5"/>
   </fill>

        ......

        ......
</Column3DChart>
```

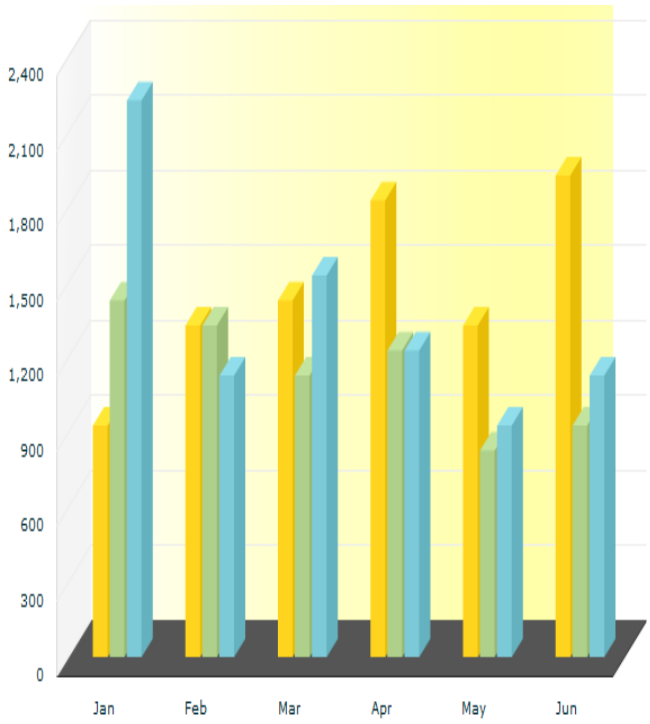
*칼럼 차트를 예로 든 경우 입니다.
반드시 Chart 의 자식으로 <fill>을 정의하십시오.



<예제 59 차트 전체 배경에 단색 컬러를 설정한 경우>

Linear 그라데이션 컬러 지정하기.

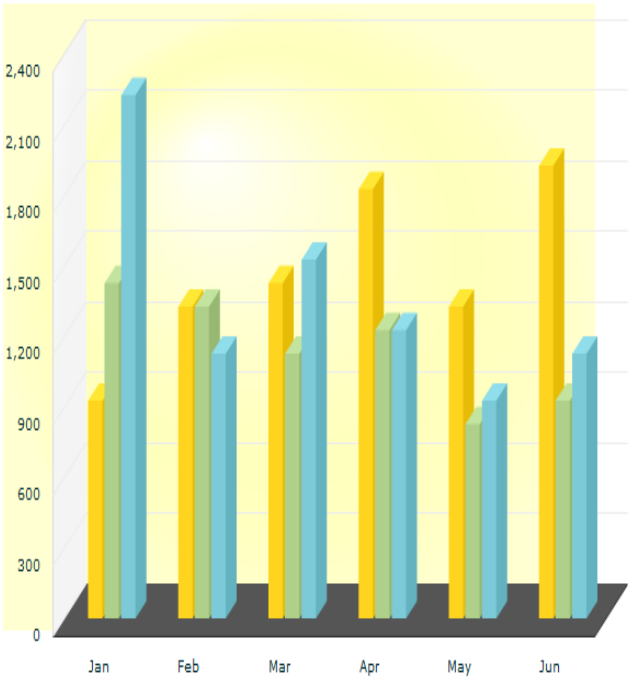
```
<Column3DChart>
     <fill>
        <LinearGradient angle="0">
           <entries>
              <GradientEntry
                   color="0xFFFFFF"
                   ratio="0.125"
                   alpha="1"/>
               <GradientEntry
                   color="0xFFFF99"
                   ratio="0.66"
                   alpha="0.65"/>
               <GradientEntry
                   color="0xFFFF00"
                   ratio="1"
                   alpha="0.3"/>
           </entries>
         </LinearGradient>
       </fill>

       ......
</Column3DChart>
```



<예제 60 차트 전체 배경에 Linear 그라데이션을 설정한 경우>

8.2.2. Radial 그라데이션 컬러 지정하기.

```
<Column3DChart>
       <fill>
          <RadialGradient
focalPointRatio="-0.65" angle="60">
           <entries>
              <GradientEntry
                   color="0xFFFFFF"
                   ratio="0"
                   alpha="1"/>
               <GradientEntry
                   color="0xFFFF99"
                   ratio="0.66"
                   alpha="0.65"/>
               <GradientEntry
                   color="0xFFFF66"
                   ratio="1"
                   alpha="0.3"/>
           </entries>
          </RadialGradient>
       </fill>
       ......
</Column3DChart>
```

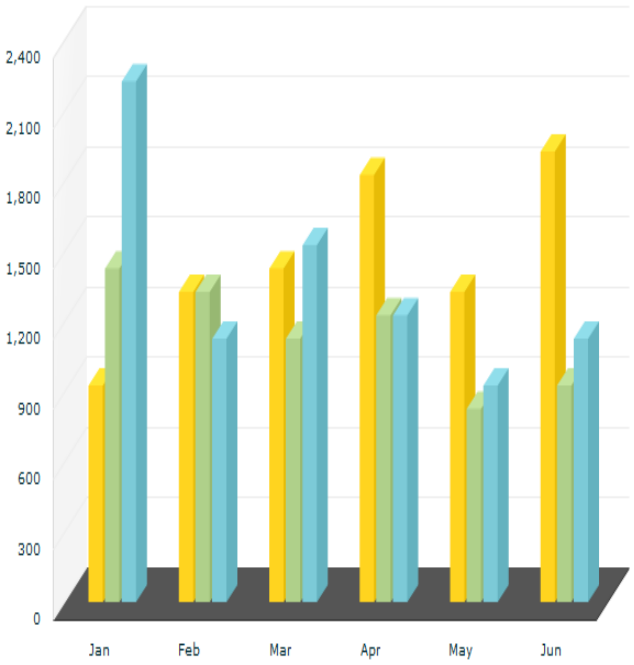


<예제 61 차트 전체 배경에 Radial 그라데이션을 설정한 경우>

8.3. 차트 시리즈 아이템 각각에 컬러 지정하기.

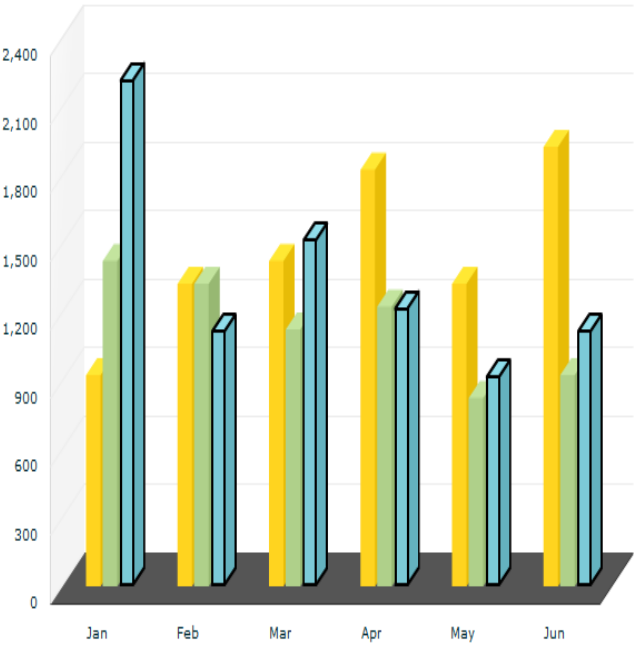
차트 시리즈 아이템의 컬러는 선 색깔과 채우기 색으로 나뉩니다.
아래 예제를 참고해 주십시오.

```
<series>
    <Column3DSeries yField="Profits">
      <fill>
         <SolidColor color="0xFF33CC"
         alpha="1"/>
      </fill>
    </Column3DSeries>
    <Column3DSeries yField="Costs">
      <fill>
          <SolidColor color="0xFFFF99"
          alpha="1"/>
      </fill>
    </Column3DSeries>
    <Column3DSeries yField="Overhead">
      <fill>
         <SolidColor color="0x33FFCC"
         alpha="1"/>
      </fill>
    </Column3DSeries>

</series>
```



<예제 62 각각의 시리즈에 채우기 색만 설정한 경우>

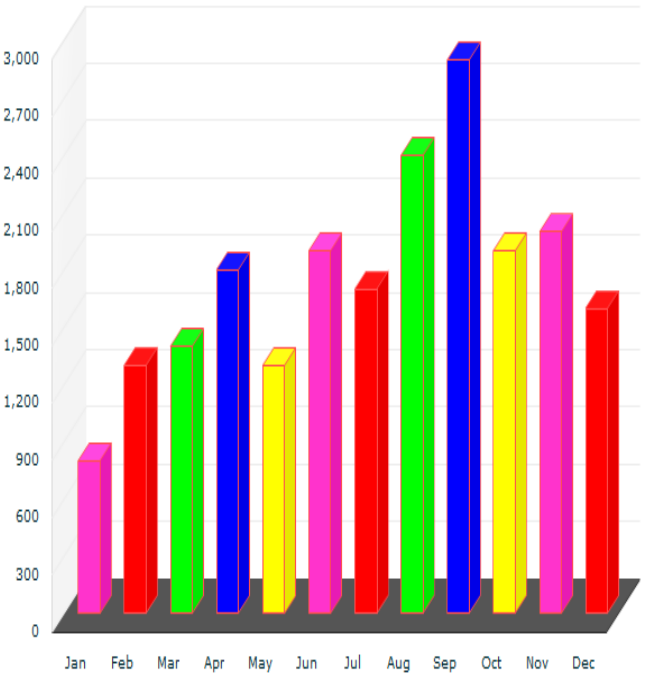
```
<series>
    <Column3DSeries yField="Profits">
      <fill>
          <SolidColor color="0xFF33CC"
          alpha="1"/>
      </fill>
    </Column3DSeries>

    ......

    <Column3DSeries yField="Overhead">
      <fill>
          <SolidColor color="0x33FFCC"
          alpha="1"/>
      </fill>
      <stroke>
          <Stroke weight="2"
          color="0x000000"/>
        </stroke>
    </Column3DSeries>

</series>
```



<예제 63 각각의 시리즈에 채우기 색을 설정하고, 마지막 시리즈에 선 색깔을 설정한 경우>

```
<series>
<Column3DSeries yField="Profits">
    <fills>
        <SolidColor color="0xFF33CC"/>
        <SolidColor color="0xFF0000"/>
        <SolidColor color="0x00FF00"/>
        <SolidColor color="0x0000FF"/>
        <SolidColor color="0xFFFF00"/>
    </fills>
</Column3DSeries>

<Column3DSeries
              yField ="Costs">
        ......
</series>
```

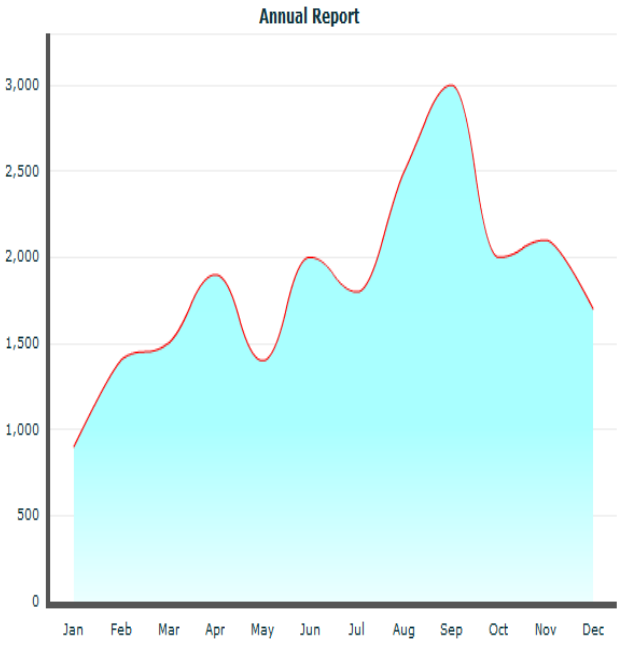
- 첫 번째 시리즈에 5개의 채우기 색을 정의하였습니다. 이는 5개의 색이 하나의 주기로 반복합니다. 만약 2개의 색이라면 2개의 색이 반복합니다.
- 2개 이상의 색에는 <fills>를 사용.



<예제 64 첫번째 시리즈에 fills 를 사용하여 5 개의 채우기 색을 설정한 경우>

```
<series>
   <Area2DSeries yField="Profits"
         form="curve"
         displayName="Profits">

      <areaFill>
            <SolidColor
 color="0x46c6c6"/>
    </areaFill>
    <areaStroke>
       <Stroke color="0xFF0000"
weight="3"/>
    </areaStroke>
   </Area2DSeries>
</series>
```

- 영역차트는 areaFill, areaStroke 를 사용한다는 점에 주의.



<예제 65 영역차트에 fill 과 stroke 를 한 경우>

```
<Line2DChart>
    <horizontalAxis>
        <CategoryAxis
categoryField="Month"/>
    </horizontalAxis>
    <series>
        <Line2DSeries yField="Profit"
form="curve" displayName="Profit">
            <lineStroke>
                <Stroke color="0xff5050"
weight="4"/>
// color는 선색깔을 나타냅니다.
// weight 은 선의 두께를 나타냅니다.
            </lineStroke>
        </Line2DSeries>
    </series>
</Line2DChart>
```

- 라인차트는 lineStroke 를 사용한다는 점에 주의.

<예제 66 라인차트의 선(stroke)를 변경한 경우>

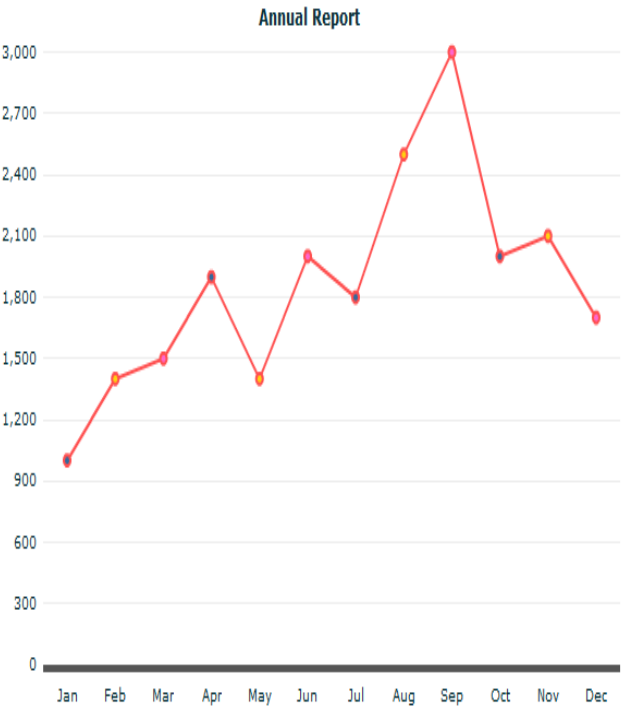
라인차트와 컴비네이션차트를 제외한 모든 차트에는 자체적으로 그림자 필터가 디폴트 값으로 존재합니다.

이 그림자를 다른 색 또는 그림자의 길이를 제어하고 싶을 때는 다음과 같이 정의 하십시오.

<Line2DChart> 속성으로 <seriesFilters>를 정의한 후 <DropShadowFilter>를 새로 만들어 설정합니다.

```
<Line2DChart>
    <horizontalAxis>
        <CategoryAxis
categoryField="Month"/>
    </horizontalAxis>
    <seriesFilters>
        <DropShadowFilter />
    </seriesFilters>
    <series>
        <Line2DSeries yField="Profit"
form="segment" displayName="Profit">
            <lineStroke>
                <Stroke color="0x339966"
weight="1"/>
// color는 선색깔을 나타냅니다.
// weight 은 선의 두께를 나타냅니다.
            </lineStroke>
        </Line2DSeries>
    </series>
</Line2DChart>
```

- 라인차트는 lineStroke 를 사용함에 주의



<예제 67 라인차트의 그림자를 해제한 예제>

DropShadowFilter 의 속성과 유효 값을 나타낸 표입니다.

속성 명	유효 값	설명
distance	Number(기본값:4.0)	그림자의 길이를 나타냅니다.
color	RGB컬러(기본값:0x000000)	그림자의 컬러를 나타냅니다.
alpha	0.0 ~ 1.0(기본값:1)	그림자의 투명도를 나타냅니다.
angle	Degree(0~360) (기본값:45)	그림자의 각도를 나타냅니다.

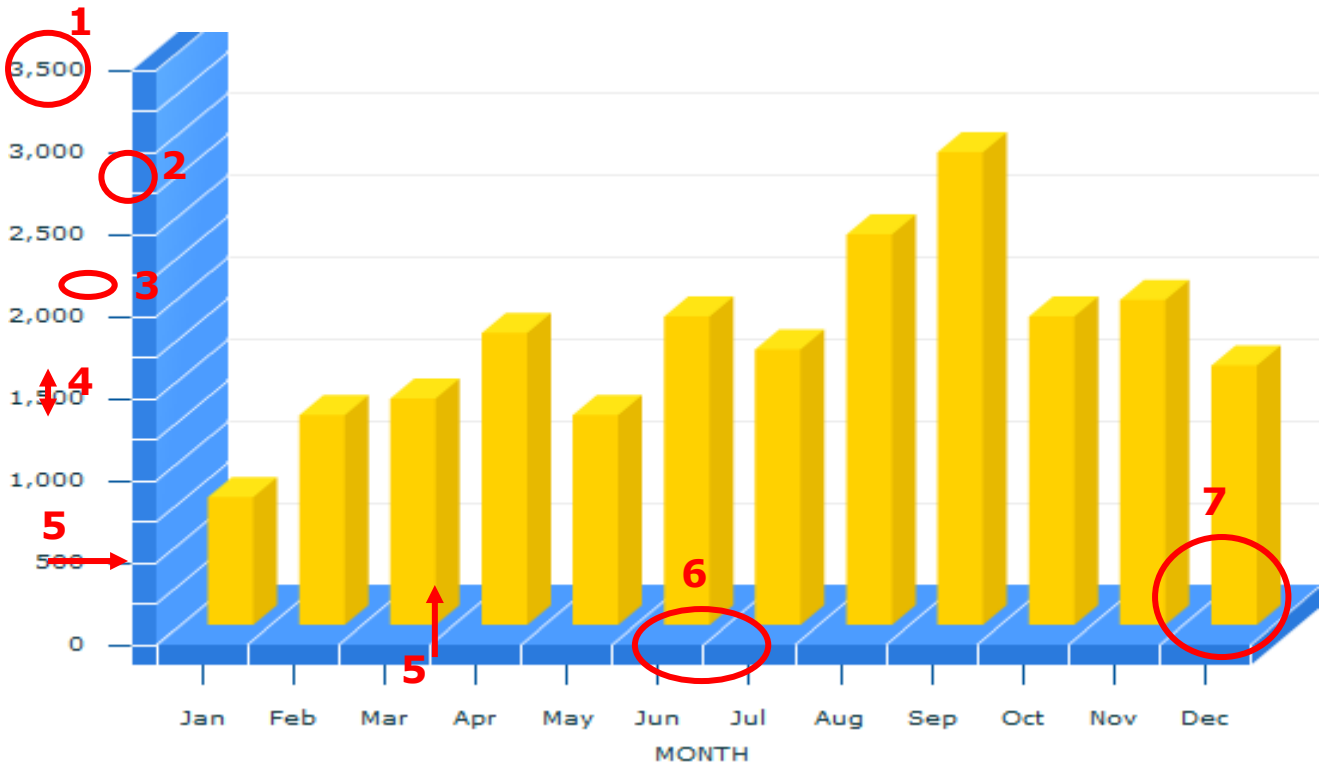
<표 42 DropShadowFilter 속성 명과 유효 값 설명>

- **차트에서 기본으로 설정된 그림자 필터를 제거하고자 할 때는 다음과 같이 설정 하십시오.**

```
<seriesFilters>
    <DropShadowFilter alpha='0' distance='0'/>
</seriesFilters>
```

8.4. 축(Axis)에 대한 스타일 적용하기.

8.4.1. 축 스타일 설명 및 예제.



<그림 44 축 꾸미기를 표현한 그림>

위는 수평축과 수직축을 사용자 정의한 화면입니다. 위에 표시된 번호의 설명은 아래와 같습니다.

1. 수직축의 Maximum 값을 나타냅니다. 이는 오직 LinearAxis 에서만 유효합니다.
2. minorTick 을 나타냅니다. minorTick 의 길이와 Stroke 스타일을 정의할 수 있습니다.
3. tick 을 나타냅니다. tick 길이와 tick 의 위치, Stroke 스타일을 정의할 수 있습니다.
4. 축에 표시된 숫자들의 간격을 의미합니다. 현재는 500 입니다. LinearAxis 에서만 유효합니다.
5. 수직축과 수평축의 전체적인 Stroke 스타일입니다. 현재 하늘색의 축을 정의하였습니다.
6. 축의 대표 제목을 붙일 수 있습니다.
7. tick 과 라벨과의 관계입니다. 라벨과 라벨 사이에 tick 존재하는 경우와 라벨 위에 tick 존재하는 경우로 나뉩니다. 현재는 라벨 위에 tick 이 존재하는 경우입니다.

위에서 지시하고 설명한 부분을 레이아웃에서 찾아보면 다음과 같습니다.

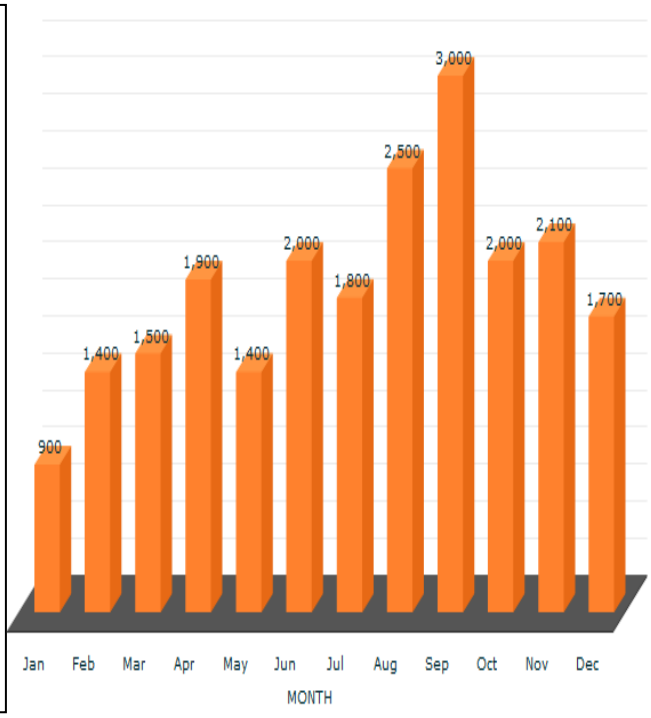
```
<Column3DChart>
      <horizontalAxis>                                          7
6           <CategoryAxis categoryField="Month" ticksBetweenLabels="false"
title="MONTH" displayName="Month"/>
      </horizontalAxis>
      <verticalAxis>                  4                         1
            <LinearAxis interval="500" baseAtZero="true" maximum="3500"/>
      </verticalAxis>
      <horizontalAxisRenderers> <!-- 수평축-->
            <Axis3DRenderer visible="true"
3                 tickLength="5"          2
                  minorTickLength="5"
                  tickPlacement="outside"
                  placement="bottom"  <!--수평축의 위치 나타냄 top 또는 bottom
                  canDropLabels="false"
                  showLabels="true"
                  labelAlign="center">
                                5
                  <axisStroke>
                        <Stroke weight="10" color="0x66CCFF" caps="none"/>
                  </axisStroke>
                  <tickStroke>
3                       <Stroke weight="1" color="0x000000" alpha="0.5" />
                  </tickStroke>
                  <minorTickStroke>  2
                        <Stroke weight="1" color="0x000000" caps="square"/>
                  </minorTickStroke>
            </Axis3DRenderer>
      </horizontalAxisRenderers>

        <verticalAxisRenderers> <!-수직축 꾸미기는 수평축과 같습니다. 수평축을 참고하여
                                    주십시오.-->
            <Axis3DRenderer visible="true"
                  tickLength="30"
                  minorTickLength="3"
                  tickPlacement="left"
                  placement="left"  <!-수직축의 위치, left 또는 right-->
                  canDropLabels="false"
                  showLabels="true"
                  labelAlign="center">
                  <axisStroke>
                        <Stroke weight="10" color="0x66CCFF" caps="none"/>
```

```
                    </axisStroke>
                    <tickStroke>
                          <Stroke weight="1" color="0x000000" />
                    </tickStroke>
                    <minorTickStroke>
                          <Stroke weight="1" color="0x000000" caps="square"/>
                    </minorTickStroke>
              </Axis3DRenderer>
        </verticalAxisRenderers>

        ......

</Column3DChart>
```

<예제 68 축 꾸미기 설정한 예제>

8.4.2. 축의 Visible 속성 사용하여 축의 유, 무 나타내기.

```
<Column3DChart>

        ......


    <horizontalAxisRenderers>
        <Axis3DRenderer
              placement="bottom"
              showLabels="true"
              labelAlign="center">
        </Axis3DRenderer>
    </horizontalAxisRenderers>

    <verticalAxisRenderers>
         <Axis3DRenderer
               visible="false">
         </Axis3DRenderer>
    </verticalAxisRenderers>

        ......

</Column3DChart>
```



<예제 69 수직축 보이지 않게 설정한 예제>

8.4.3. 축의 위치 바꾸기.

```
<rMateChart>
  <Column3DChart>

        ......

      <horizontalAxisRenderers>
         <Axis3DRenderer
              placement="top"
              showLabels="false">
         </Axis3DRenderer>
      </horizontalAxisRenderers>

      <verticalAxisRenderers>
         <Axis3DRenderer
              placement="right"
              showLabels="true"
              labelAlign="center">
         </Axis3DRenderer>
      </verticalAxisRenderers>

        ......
```

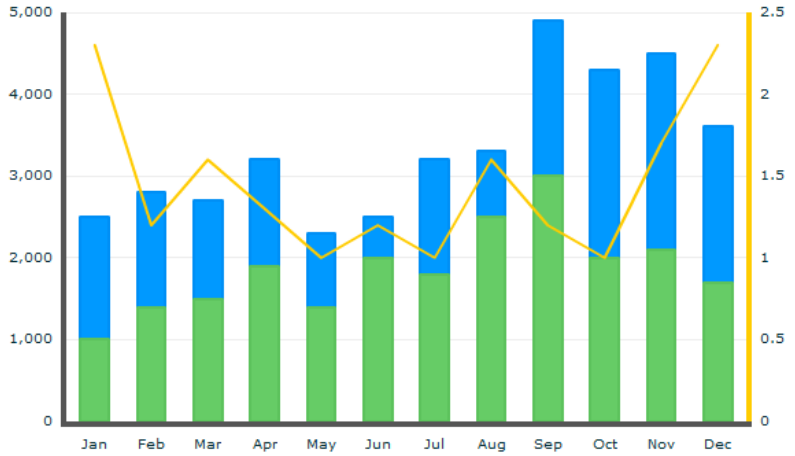
<예제 70 수평축을 위쪽으로, 수직축을 오른쪽으로...수직축 라벨을 감춘 예제>

8.4.4. 세로축 2 개(듀얼축)로 각각의 데이터 표현하기.

```
<rMateChart cornerRadius="12" borderStyle="solid">
    <Options>
        <Caption text="Annual Report" />
        <SubCaption text="2008" />
        <Legend />
    </Options>
    <Combination3DChart showDataTips="true">
        <horizontalAxis>
            <CategoryAxis categoryField="Month" />
        </horizontalAxis>
        <verticalAxis>   // 칼럼시리즈가 참고할 수직축을 정의합니다.
            <LinearAxis id="axis1"/>
        </verticalAxis>
        <series>
            <Column3DSeries yField="Profit" displayName="Profit">
                <showDataEffect>
                    <SeriesInterpolate />
```

```
                </showDataEffect>
                <fill>
                    <SolidColor color="0x66CC66" />
                </fill>
            </Column3DSeries>
            <Line2DSeries selectable="true" yField="Cost" displayName="Cost">
                <verticalAxis> // 라인시리즈가 참고할 수직축을 정의합니다.
                    <LinearAxis id="axis2"/>
                </verticalAxis>
                <showDataEffect>
                    <SeriesInterpolate />
                </showDataEffect>
                <filters>
                    <DropShadowFilter distance="3" color="0x666666" alpha=".8" />
                </filters>
                <lineStroke>
                    <Stroke color="0xFFCC00" weight="3" />
                </lineStroke>
            </Line2DSeries>
        </series>
        <verticalAxisRenderers> //차트에서 각각의 시리즈가 만든 축을 참조하도록 설정합니다.
            <Axis3DRenderer axis="{axis1}"/>
            <Axis3DRenderer axis="{axis2}"/>
        </verticalAxisRenderers>
    </Combination3DChart>
</rMateChart>
```

<예제 71 수직축 2 개 설정한 예>

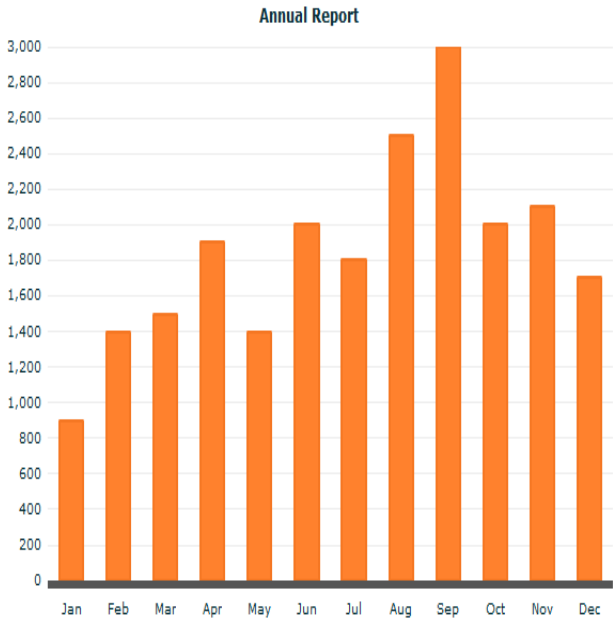


<그림 45 수직축 2 개 설정한 경우 결과>

8.4.5. 수직, 수평 축의 수치에 3 자리 단위로 컴마(,) 찍기

LinearAxis 로 정의된 수직 축 또는 수평 축은 출력 결과가 숫자입니다. 수의 단위가 천 단위를 넘었을 경우 컴마 구분자를 삽입하여 읽기 쉽게 표현할 수 있습니다.

```
<rMateChart cornerRadius="12">
  <Options>
     <Caption text="Anual Report"/>
  </Options>
  <NumberFormatter id="numfmt"
     useThousandsSeparator="true"/>
  <Column3DChart>
    <horizontalAxis>
       <CategoryAxis
          categoryField="Month"/>
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis
           formatter="{numfmt}"/>
    </verticalAxis>
    <series>
       <Column3DSeries yField="Profit"
          displayName="Profit"/>
    </series>
  </Column3DChart>
</rMateChart>
```



<예제 72 수직축 수치에 컴마 구분자 넣기>

속성 명	유효 값	설명
useThousandsSeparator	True | false (기본값:true)	true 의 경우, 수치는 3 자리수 마다 정해진 구분자로 구분됩니다.
useNegativeSign	true | false (기본값:true)	true 인 경우 음수 앞에 마이너스 부호(-)를 붙여 출력합니다. false 인 경우 음수를 괄호로 출력합니다.
precision	Number (기본값:-1)	출력 할 소수의 자릿수를 나타냅니다. precision를 -1 로 설정하면 precision를 무효로 할 수 있습니다. 값이 -1 의 경우, 자릿수를 변경하지 않습니다.
thousandsSeparatorTo	구분 문자 (기본값: 컴마(,))	출력 숫자의 자릿수 구분 기호로서 사용하는 캐릭터를 나타냅니다.
decimalSeparatorTo	구분 문자 (기본값:도트(.))	소수의 값을 출력할 경우에 사용하는 소수점의 단락 캐릭터를 나타냅니다.

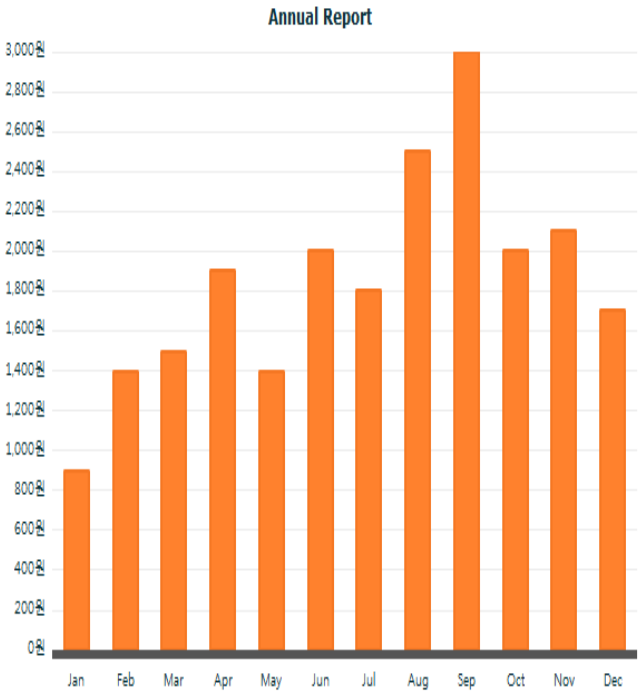
rounding	down", "nearest", "up", "none"(기본값:"none")	정해진 소수점 위치에서 올림, 반올림 등을 결정합니다.

<표 43 NumberFormatter 속성 설명표>

8.4.6. 수직, 수평 축에 통화단위 넣기

수직축 또는 수평축이 금액을 나타내는 경우 통화 단위를 넣을 필요가 있습니다. 아래 예제는 대한민국 통화단위인 "원"을 오른쪽에 삽입한 예제입니다

```
<rMateChart>
    <Options>
      <Caption text="Anual Report"/>
    </Options>
    <CurrencyFormatter id="fmt"
           currencySymbol="원"
           alignSymbol="right"/>
    <Column3DChart showDataTips="true"
              fontSize="11">
        <horizontalAxis>
            <CategoryAxis
          categoryField="Month"/>
        </horizontalAxis>
        <verticalAxis>
          <LinearAxis
formatter="{fmt}"/>
        </verticalAxis>
        <series>
            <Column3DSeries
              yField="Profit"/>
        </series>
        </Column3DChart>
</rMateChart>
```



<예제 73 수직 축에 통화단위 "원"을 삽입한 예제>

속성 명	유효 값	설명
currencySymbol	문자(기본값:$)	포맷의 대상이 되는 수치의 통화 기호로서 사용되는 캐릭터를 나타냅니다
alignSymbol	left | right (기본값:left)	통화 기호의 위치를, 포맷이 끝난 수치의 좌측 또는 우측으로 설정합니다.
useThousandsSeparator	True | false (기본값:true)	true 의 경우, 수치는 3 자리수 마다 정해진 구분자로 구분됩니다.

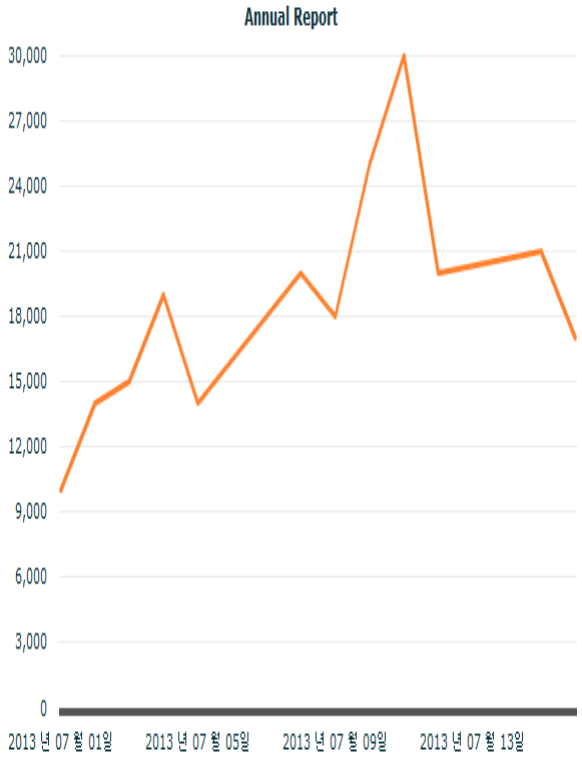
useNegativeSign	true | false (기본값:true)	true 인 경우 음수 앞에 마이너스 부호(-)를 붙여 출력합니다. false 인 경우 음수를 괄호로 출력합니다.
precision	Number (기본값:-1)	출력 할 소수의 자릿수를 나타냅니다. precision를 -1 로 설정하면 precision를 무효로 할 수 있습니다. 값이 -1 의 경우, 자릿수를 변경하지 않습니다.
thousandsSeparatorTo	구분 문자 (기본값: 컴마(,))	출력 숫자의 자릿수 구분 기호로서 사용하는 캐릭터를 나타냅니다.
decimalSeparatorTo	구분 문자 (기본값:도트(.))	소수의 값을 출력할 경우에 사용하는 소수점의 단락 캐릭터를 나타냅니다.

<표 44 CurrencyFormatter 속성 설명표>

8.4.7. 수직, 수평 축을 DateTimeAxis 정의한 경우 날짜 포맷 변환하기.

라인차트에서 수평축을 DateTimeAxis 로 정의한 경우 수평축은 날짜(시간)에 따라 데이터가 출력됩니다. 예를 들어 “days”에 따라 데이터가 표현될 때 수평축의 라벨은 “2009/07/10”, “2009/07/11” 과 같이 표현됩니다. 수평축에 표현된 날짜의 포맷을 “07/10/2009” 처럼 “월/일/년” 또는 “2009 년 7 월 10 일” 처럼 사용자 정의 형식으로 표현하고 할 때 DateFormatter 를 사용합니다.
아래 예제는 수평축의 라벨을 “2009 년 7 월 10 일” 로 표현한 것 입니다.

```
<rMateChart>
  <Options>
     <Caption text="Anual Report"/>
  </Options>
  <DateFormatter id="dateFmt"
  formatString="YYYY 년 MM 월 DD일" />
  <NumberFormatter id="curFmt" />
  <Line2DChart showDataTips="true"
  fontSize="11">
    <horizontalAxis>
       <DateTimeAxis dataUnits="days"
       labelUnits="days" interval="4"
       formatter="{dateFmt}"
       alignLabelsToUnits="false"
       displayLocalTime="true" />
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis formatter="{curFmt}"/>
    </verticalAxis>
    <series>
      <Line2DSeries xField="Date"
      yField="Profit"
      displayName="Profit"/>
    </series>
  </Line2DChart>
</rMateChart>
```



<예제 74 DateTime 축의 날짜 포맷 변환 예>

DateFormatter 를 정의한 후 임의의 id 를 부여합니다. 그리고 DateTimeAxis 의 formatter 속성에 DateFormatter 의 id 를 할당하십시오. **주의할 점은 id 는 일반 String 객체가 아닌 고유 객체를 나타내는 스트링이므로 중괄호로 묶습니다. (**<예제 14 ID 에 의한 속성 값 할당 법> 참고)

DateFormatter 의 속성인 formatString 표현하는 패턴을 정의하는 속성입니다.

예를 들어 "2009/12/01" 과 같이 표현하고자 할 때는 "YYYY/MM/DD"로 정의하십시오. 이에 대한 설명은 아래 표와 같습니다.

패턴 캐릭터	설 명	표현 예
Y	연도(year)입니다	YY = 05 YYYY = 2005 YYYYY = 02005
M	월(month)입니다.	M = 7 MM= 07 MMM=Jul MMMM= July
D	일(day)입니다.	D = 4

		DD = 04 DD = 10
H	시(Hour)입니다.	H = 1 HH = 01
N	분(Minute)입니다.	N = 1 NN = 01
S	초(Second)입니다.	SS = 30

<표 45 formatString 패턴 캐릭터 설명>

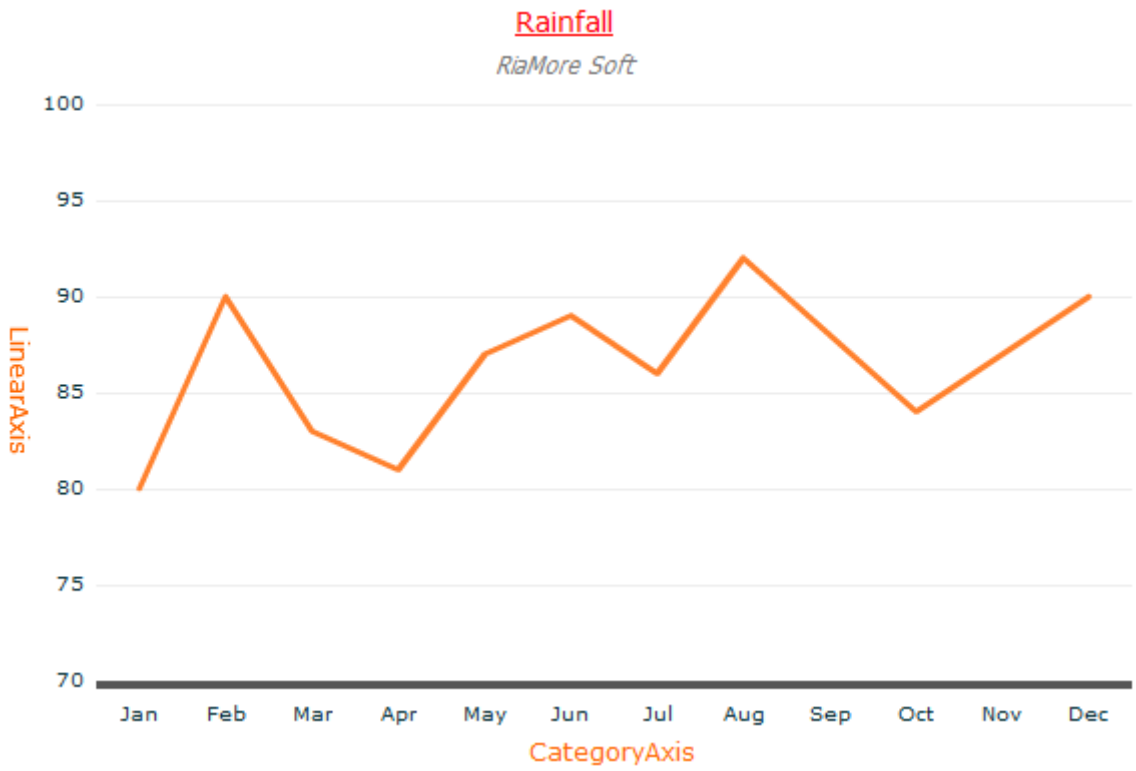
8.4.8. 축 제목(대표문자) 삽입하기.

수직 축 또는 수평 축이 현재 표현하는 데이터의 이름을 넣고자 할 때가 있습니다. 예를 들어 수직 축에 "Profit" 이라는 제목을 수평 축에 "Month" 라는 제목을 넣고자 할 때 예제는 아래와 같습니다.

```
<rMateChart>
        <Options>
                <Caption text="Anual Report"/>
        </Options>
        <Column3DChart showDataTips="true">
        <horizontalAxis>
           <CategoryAxis categoryField="Month" title="Month"/>//가로축의 제목은 Month
        </horizontalAxis>
        <verticalAxis>
            <LinearAxis title="Profit"/> //세로축의 제목은 Profit
        </verticalAxis>
        <horizontalAxisRenderers>              //가로축을 꾸미기 위해 정의하였습니다.
                        <Axis3DRenderer   axisTitleStyleName="title"/>
        </horizontalAxisRenderers>
        <verticalAxisRenderers>                //세로축을 꾸미기 위해 정의하였습니다.
                        <Axis3DRenderer   axisTitleStyleName="title"/>
        </verticalAxisRenderers>
        <series>
           <Column3DSeries yField="Profit"/>
        </series>
    </Column3DChart>
    <Style> //축의 제목(대표문자)를 꾸미기 위해 정의한 스타일 노드입니다.
        .title   //12포인트, 파랑의 볼드체로 축 제목을 꾸밉니다.
        {
                fontSize:12;
                color:0x0000FF;
```

```
                fontWeight:"bold";
        }
    </Style>
</rMateChart>
```

<예제 75 축에 제목(대표 문자)를 삽입한 예제>

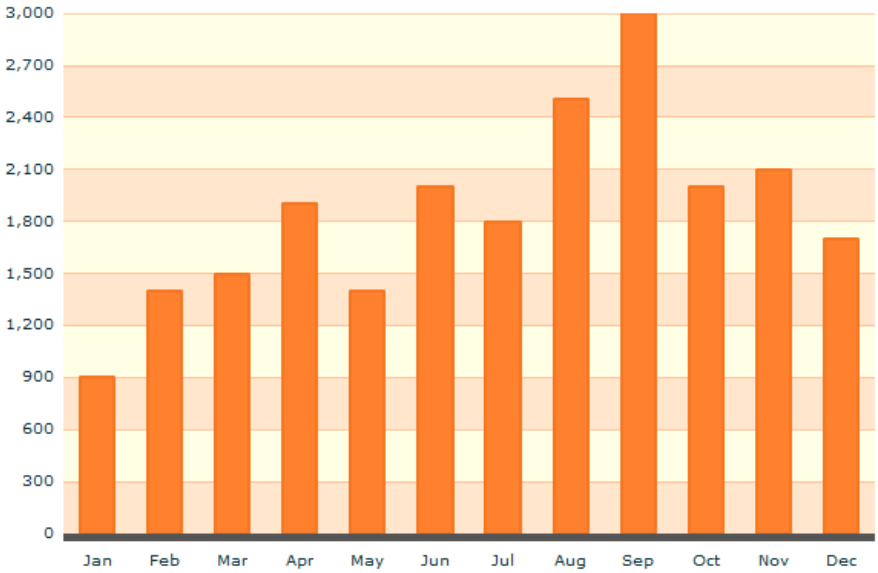


<그림 46 축에 제목(대표 문자)를 삽입한 결과화면>

8.5. 차트 축을 기준으로 안쪽의 배경 꾸미기.

8.5.1. 차트 안쪽 배경에 그리드 라인 삽입하기.

X 축과 Y 축을 갖는 모든 차트는 속성으로 backgroundElements 를 갖습니다. 이 backgroundElements 는 차트에서 축을 기준으로 안쪽에 해당되는 영역의 배경을 의미합니다. 이곳을 꾸미기 위해 backgroundElements 속성을 이용합니다.



<예제 76 차트의 축 안쪽 배경에 스타일 적용한 경우>

위 예제는 수평선만을 정의한 경우입니다. 이에 대한 옵션 설명은 다음과 같습니다.

- direction = "horizontal", "vertical", "both" : 수평선 정의, 수직선 정의, 수평,수직 모두 정의
- horizontalChangeCount = "1" : 1 줄 단위로 fill 과 alternateFill 색이 변합니다.
- horizontalStroke : fill 과 alternateFill 사이의 선을 정의합니다.
- horizontalFill : 최초 시작하는 채우기 색입니다.(예제의 흰색)
- horizontalAlternateFill : fill 의 상대적인 채우기 색입니다.(예제의 회색)

아래 레이아웃 예제에서 direction = "vertical" 로 변경하면 수직선이 나옵니다. 수직선 정의 부분은 수평선과 같습니다. 참고하십시오. (레이아웃 예제의 회색 강조 부분이 수직선을 의미합니다.

```
<rMateChart>
    <Column3DChart>
        <backgroundElements>
            <GridLines direction="horizontal" verticalChangeCount="1" horizontalChangeCount="1">
                <horizontalStroke>
                    <Stroke color="0x000000" alpha="1" weight="2"/>
                </horizontalStroke>
                <horizontalFill>
                    <SolidColor color="0xffffff" alpha="0.5"/>
                </horizontalFill>
                <horizontalAlternateFill>
                    <SolidColor color="0x000000" alpha="0.5"/>
                </horizontalAlternateFill>
                <verticalStroke>
                    <Stroke color="0x000000" alpha="1" weight="2"/>
                </verticalStroke>
                <verticalFill>
```

```
                        <SolidColor color="0x00ffff" alpha="0.5"/>
                    </verticalFill>
                    <verticalAlternateFill>
                        <SolidColor color="0x999999" alpha="0.5"/>
                    </verticalAlternateFill>
                </GridLines>
            </backgroundElements>
        </Column3DChart>
</rMateChart>
```

<예제 77 차트의 축 안쪽 배경에 스타일 적용한 예제>

8.5.2. 차트 안쪽 배경에 이미지 삽입하기.

차트 안쪽 배경에 외부 이미지를 삽입하기 위해서는 차트의 backgroundElements 속성 또는 annotationElements 속성을 이용합니다. backgroundElements 속성은 그래프가 뿌려진 레이어를 기준으로 뒷면을 의미하고 annotationElements 속성은 그래프가 뿌려진 레이어를 기준으로 앞면을 의미합니다. 다음은 backgroundElements 속성을 이용하여 이미지를 삽입하는 예제입니다.

```
<rMateChart backgroundColor="0xFFFFEE"   cornerRadius="12" borderStyle="solid">
        <Line2DChart showDataTips="true">
            <horizontalAxis>
                <CategoryAxis categoryField="Month"/>
            </horizontalAxis>
            <series>
                <Line2DSeries yField='Profit' displayName='Profit'
itemRenderer='mx.charts.renderers.CircleItemRenderer' radius='5'>
                    <showDataEffect>
                        <SeriesInterpolate />
                    </showDataEffect>
                    <lineStroke><Stroke color='0xFF3366' weight='2'/></lineStroke>
                    <stroke><Stroke color='0xFF3366' weight='2'/></stroke>
                </Line2DSeries>
            </series>
            <backgroundElements>
                <Image source="http://www.riamore.net/image/riamoreMain.jpg"
maintainAspectRatio="false" alpha="1"/>
            </backgroundElements>
        </Line2DChart>
</rMateChart>
```

<예제 78 차트 배경에 이미지 삽입 예제>

<그림 47 차트 배경에 이미지 삽입한 화면>

차트 앞면에 이미지를 삽입하는 방법은 <예제 78 차트 배경에 이미지 삽입 예제>에서 다음을 수정해 주십시오.

```
<annotationElements>
                        <Image source="http://www.riamore.net/image/riamoreMain.jpg"
maintainAspectRatio="false" alpha="0.25"/>
</ annotationElements>
```

<예제 79 차트 앞면에 이미지 삽입한 예제>

Image 의 투명도가 1 이라면 그래프가 전혀 보이지 않기 때문에 alpha 값을 적절히 조절해 주십시오.

다음은 Image 객체의 옵션 설명입니다.

속성명	설명
source	실제 이미지가 있는 URL 경로입니다.
maintainAspectRation	이미지 원본 비율을 유지할지 어떨지를 나타냅니다. 만약 이 속성이 true 라면 차트 고유 비율을 유지하려합니다. 하지만 가로이든 세로이든 차트에 맞춰진 모습을 나타냅니다. false인 경우 차트에 딱맞춰진 사이즈를 갖습니다.
alpha	이미지의 투명도를 나타냅니다.(유효값 : 0~1)

<표 46 Image 객체 속성 설명>

8.6. 차트 생성시 효과 적용하기.

차트 고유의 effect 에는 SeriesInterpolate, SeriesSlide, SeriesZoom 의 3 종류가 있습니다. rMate Chart 는 데이터가 화면에 표시될 때 effect 가 적용됩니다.

- SeriesInterpolate : 차트의 시리즈 계열에서 자신이 받은 수치를 표현하기 위해 초기값(0)에서 자신의 수치로 이동하는 모습을 나타내는 이펙트입니다.
- SeriesSlide : 차트의 시리즈가 오른쪽에서 왼쪽으로 미끄러지듯 나타나는 이펙트입니다.
- SeriesZoom : 차트의 시리즈가 점점 커지면서 나타나는 이펙트입니다.

속성 명	유효 값	설명	적용가능한 이펙트
duration	밀리 세컨드(기본값:500)	effect 전체를 완료하는데 필요한 시간을 밀리 세컨드 단위로 지정합니다.	공통
elementOffset	밀리 세컨드(기본값 : 20)	effect를 지연 시키는 시간을 밀리 세컨드 단위로 지정합니다.	
minimumElementDuration	밀리 세컨드(기본값 : 0)	각 엘리먼트의 최소 지속 시간을 설정합니다. 계열에 포함되는 어느 엘리먼트에 대해서도, 이 값(밀리 세컨드)보다 짧은 시간에 effect가 실행되지 않습니다.	
offset	밀리 세컨드(기본값 : 0)	effect 개시의 지연 시간을 밀리 세컨드 단위로 지정합니다	
perElementOffset	밀리 세컨드	각 엘리먼트의 개시 지연시간을 밀리 세컨드 단위로 지정합니다.	
direction	left | right | up | down (기본값 : left)	슬라이딩해 오는 방향을 설정합니다.	SeriesSlide
horizontalFocus	center | left | right (기본값 : center)	줌을 시작하는 수평의 초점을 정합니다.	SeriesZoom
verticalFocus	center | top | bottom (기본값 : center)	줌을 시작하는 수직의 초점을 정합니다.	
relativeTo	chart | series (기본값 : chart)	줌의 초점을 위한 경계를 정합니다.	

<표 47 차트 생성시 효과(Effect) 속성과 유효값 설명>

다음과 같이 효과의 설정이 가능합니다.

```
<rMateChart>
    <Column3DChart>
        <Column3DSeries yField="Profit">
            <showDataEffect>
                <SeriesSlide duration="1000"   //구체적으로 이펙트 설정한 경우..
                    direction="down"
                    minimumElementDuration="20"
                    perElementOffset="0"
                    elementOffset="30"/>
            </showDataEffect>
        </Column3DSeries>
        <Column3DSeries yField="Cost">
            <showDataEffect>
                < SeriesSlide />   // 기본값만 적용하여 이펙트 설정한 경우.
            </showDataEffect>
        </Column3DSeries>
        ……
        ……
</rMateChart>
```

<예제 80 SeriesSlide 로 이펙트 설정한 예제>

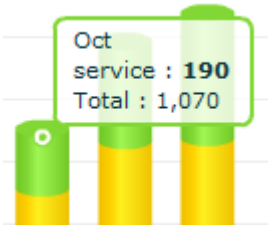
위에 작성한 예제의 이펙트는 다음과 같이 실행됩니다.
각 엘리먼트는 전의 엘리먼트의 30 밀리 세컨드 후에 개시해, 각각의 슬라이드를 완료하는데 20 밀리 세컨드가 걸립니다. effect 전체에서는, 데이터 계열이 모두 슬라이드할 때까지 1 초(1000 밀리 세컨드) 걸립니다. 슬라이딩해 오는 방향은 아래입니다.

8.7. 마우스 오버 시 데이터팁(툴팁) 나타내기.

데이터팁(툴팁)은 모든 차트(칼럼, 바, 영역, 라인 등등)에서 마우스 오버 시 나타나게 할 수 있습니다. 차트를 나타내는 노드의 속성 showDataTips 를 정의 한 후 속성값으로 true 설정을 합니다. 아래 예는 칼럼 3D 차트의 마우스 오버시 나타나는 데이터팁(툴팁)에 관한 예제입니다. showDataTips 속성은 모든 차트에 동일한 속성입니다

```
<rMateChart>
        <Options>
            <Caption text="Annual Report"/>
        </Options>
        <Column3DChart showDataTips="true">
            <horizontalAxis>
```

```
            <CategoryAxis categoryField="Month" title="Month"/>
        </horizontalAxis>
        <series>
            <Column3DSeries yField="Profit" displayName="Profit">
                <showDataEffect>
                    <SeriesInterpolate/>
                </showDataEffect>
            </Column3DSeries>

            ......
            ......
```



<마우스 오버시 모습>

```
</rMateChart>
```

<예제 81 데이터팁(툴팁) 예제>

showDataTips 를 true 로 설정하고 차트의 시리즈 속성 displayName 에 텍스트를 입력하면 차트에 마우스 오버 시 위와 같이 텍스트가 함께 출력됩니다. 예제는 Profit 이라는 텍스트를 입력하였기 때문에 Profit 이 출력되었습니다.

8.8. 칼럼 차트 스택형 데이터간 연결선 잇기.

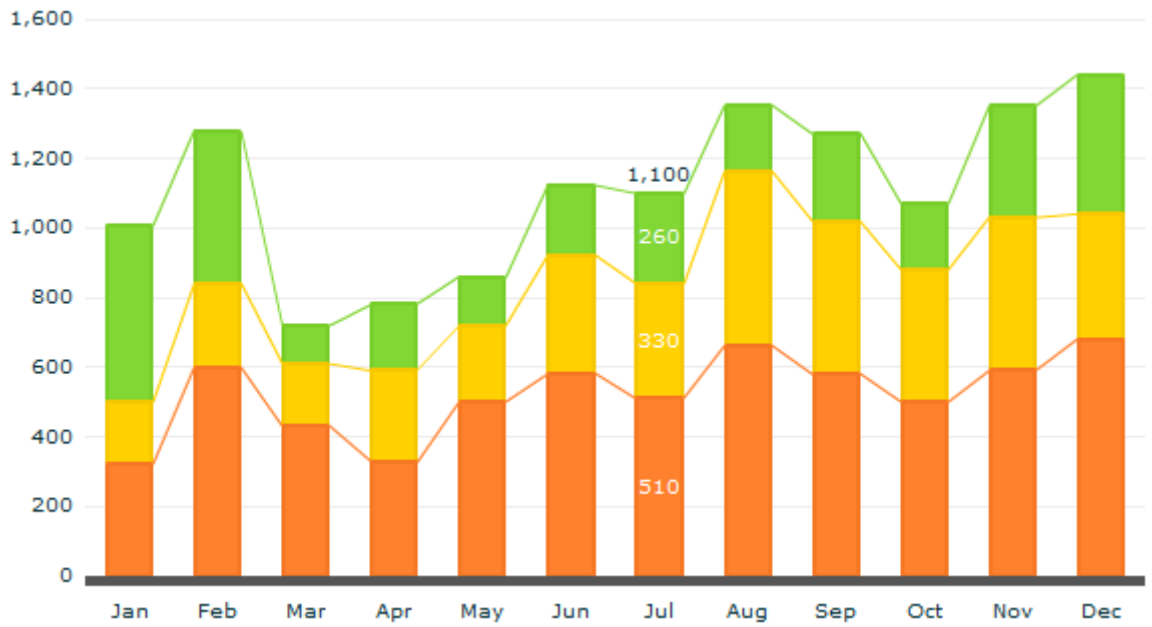
칼럼 2D, 3D 차트를 스택형으로 생성한 경우 각 데이터간 연결선을 이을 수 있습니다. 이는 Column2DSeries 또는 Column3DSeries 의 한 속성인 lineToEachItems 를 true 로 설정함으로써 가능합니다.

alwayShowLines : 데이터 0 에 대한 선을 그릴지 그리지 않을지 정의합니다.
True - (기본값) 데이터가 0 이더라도 선을 그립니다.
False – 데이터가 0 이라면 선을 그리지 않습니다.

```
<Column2DChart showDataTips="true" type="stacked"> //스택형으로 정의
        <horizontalAxis>
            <CategoryAxis categoryField="Month" />
        </horizontalAxis>
        <series>                                //연결선 잇기 true 설정
            <Column2DSeries yField="Profit" lineToEachItems="true" alwayShowLines="true">
                <linkLineStroke> //연결선의 스타일 정의
```

```
                        <Stroke weight="2" color="0x000000" caps="none" />
                    </linkLineStroke>
                    <showDataEffect>
                        <SeriesInterpolate />
                    </showDataEffect>
                </Column2DSeries>
                <Column2DSeries yField="Cost" lineToEachItems="true" alwayShowLines="true">
                    <linkLineStroke>
                        <Stroke weight="2" color="0x000000" caps="none" />
                    </linkLineStroke>
                    <showDataEffect>
                        <SeriesInterpolate />
                    </showDataEffect>
                </Column2DSeries>
                <Column2DSeries yField="Revenue" lineToEachItems="true" alwayShowLines="true">
                    <linkLineStroke>
                        <Stroke weight="2" color="0x000000" caps="none" />
                    </linkLineStroke>
                    <showDataEffect>
                        <SeriesInterpolate />
                    </showDataEffect>
                </Column2DSeries>
            </series>
</Column2DChart>
```

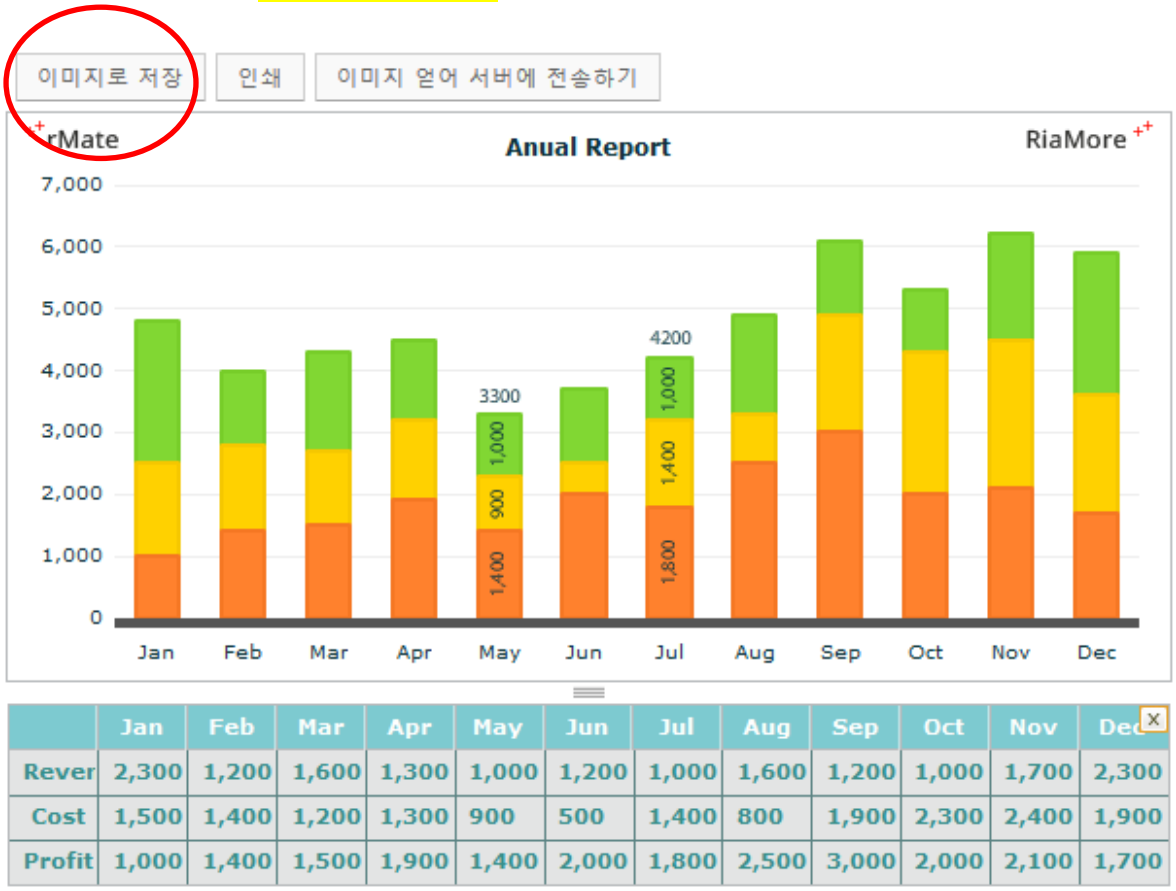
<예제 82 칼럼 2D 차트 스택형 데이터간 연결선 잇기 예제>



<그림 48 칼럼 2D 차트 스택형 연결선 이은 모습>

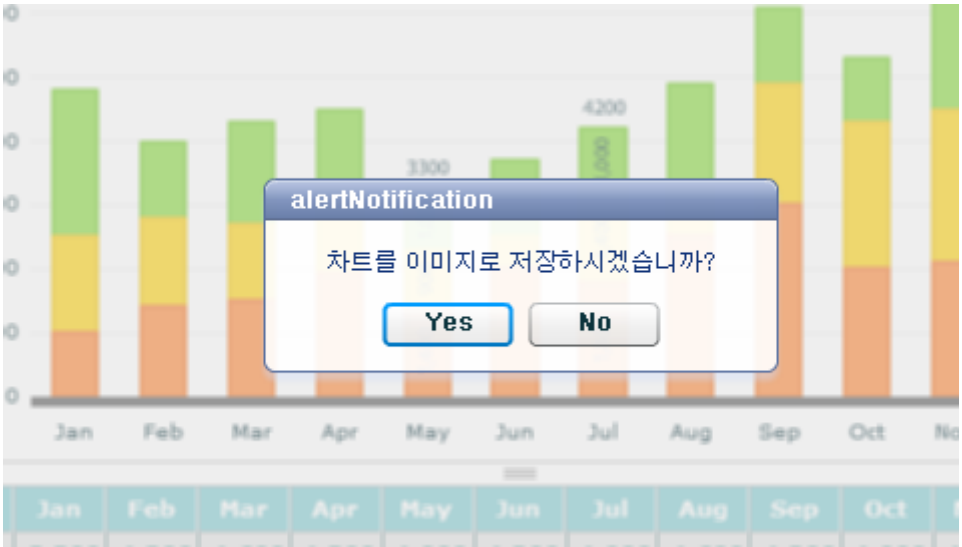
8.9. 차트 이미지 png 파일로 저장 설정하기.

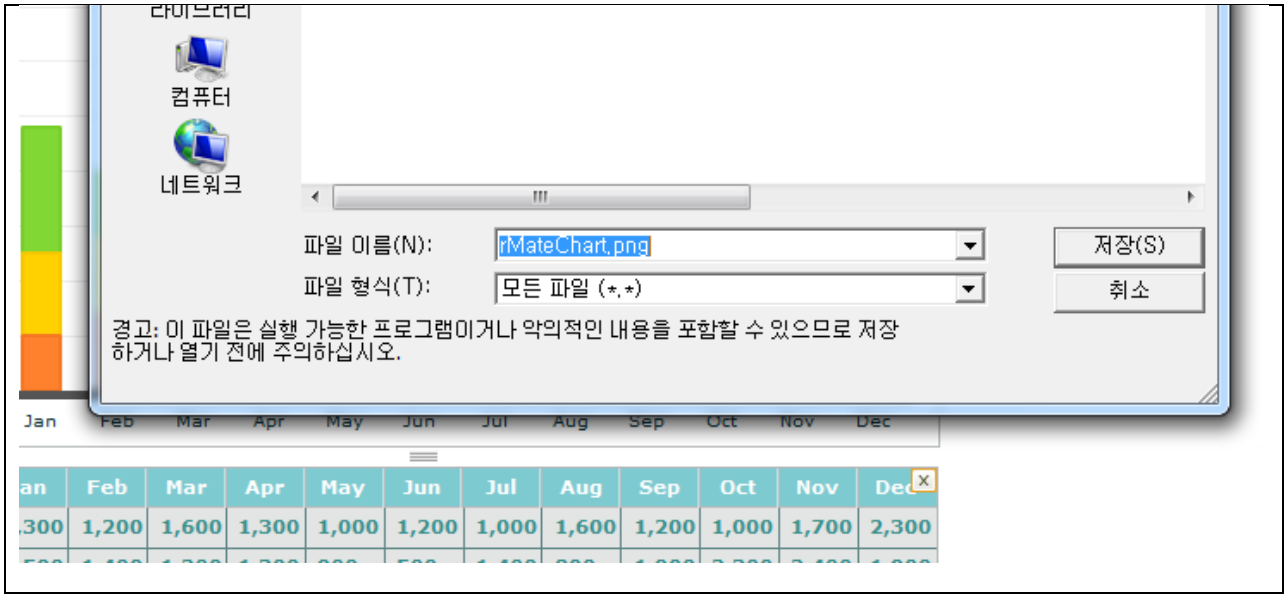
생성된 모든 차트에 대하여 png 이미지 파일로 로컬 PC 에 저장할 수 있습니다. **72dpi 로 이미지 저장을 할 수 있는 옵션이 기본값입니다.** 단, 클라이언트 PC 에 플래시 **플레이어 10 버전 이상이 요구**됩니다.
그래서 모든 차트에 대해 오른쪽 마우스 버튼을 눌렀을 때 "이미지로 저장" 메뉴를 확인 할 수 있습니다.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Rever	2,300	1,200	1,600	1,300	1,000	1,200	1,000	1,600	1,200	1,000	1,700	2,300
Cost	1,500	1,400	1,200	1,300	900	500	1,400	800	1,900	2,300	2,400	1,900
Profit	1,000	1,400	1,500	1,900	1,400	2,000	1,800	2,500	3,000	2,000	2,100	1,700

<그림 49 차트 이미지로 저장 메뉴 모습>





<그림 50 차트 이미지로 저장 실행 모습>

"차트 이미지로 저장"에 대한 설정은 flashVars 를 통해 이루어집니다. 이에 대한 설정 예제입니다.

```
// flashVars 정의..차트 레이아웃은 URL XML 형태로 읽고 데이터는 배열형태로 삽입.
var flashVars = "layoutURL="+layoutURL;
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;

flashVars += "&saveImage=true"; // 차트 이미지로 저장을 사용합니다.

flashVars += "&chartImageDpi=72"; // 이미지 dpi 는 72 로 설정합니다.
```

<예제 83 이미지로 저장 설정 예제>

8.10. 차트 아이템 클릭 시 불려지는 함수 설정하기.

차트의 아이템을 클릭 한 경우 해당 아이템의 정보를 스크립트 단으로 보낼 수 있습니다. 이는 자바스크립트의 함수를 설정하여 그 함수를 호출하는 방식으로 다양한 작업을 가능케 합니다.
먼저, 차트에게 호출할 함수 명을 지정해 줘야 합니다. 레이아웃 XML 에서 차트의 속성으로 itemClickJsFunction 을 설정하고, 자바스크립트 함수 명을 속성값으로 할당합니다. 칼럼 3D 차트인 경우 <Column3DChart/> 이며, 바차트인 경우 <Bar3DChart/>의 속성입니다.
파이차트를 예로 들면 아래와 같습니다. 이는 파이차트의 파이 한 조각(파이차트 아이템)을 클릭 한 경우 "chartClick" 이라는 함수를 호출하겠다는 이야기입니다. 이 "chartClick" 함수는 자바스크립트에서 반드시 정의되어야 합니다.

```
<rMateChart   cornerRadius="12" borderStyle="solid">
```

```
        <Options>
                <Caption text="Anual Report" />
        </Options>
        <Pie2DChart showDataTips="true" itemClickJsFunction="chartClick">
                <series>
                        <Pie2DSeries field="Profit" nameField="Month" displayName="Profit">
                                <showDataEffect>
                                        <SeriesInterpolate/>
                                </showDataEffect>
                        </Pie2DSeries>
                </series>
        </Pie2DChart>
</rMateChart>
```

<예제 84 차트 클릭 시 자바스크립트 함수 호출하기>

<예제 84 차트 클릭 시 자바스크립트 함수 호출하기> 과 같이 차트 레이아웃 XML 에서 아이템 클릭 시 호출 할 함수를 지정하였다면 이제 자바스크립트에서 이 함수명과 동일한 함수를 정의하여야 합니다. 예제는 아래와 같습니다.

```
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>


<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ---------------------- flashVars 설정 시작----------------------
var flashVars;
// 차트 레이아웃 URL 경로.
var layoutURL = encodeURIComponent("chartLayout.xml");
flashVars += "&layoutURL="+layoutURL;


// 데이타를 URL 경로를 통해 가져올 경우 설정하십시오.
// 배열형태로 데이터를 삽입할 경우 주석처리나 삭제하십시오.
// 배열형태와 같이 사용할 경우, 우선순위는 배열형태 데이터 삽입입니다.
var dataURL =encodeURIComponent("singleData.xml");
flashVars += "&dataURL="+dataURL;

// ---------------------- flashVars 설정 끝----------------------
```

```
// ------------------챠트에서 item클릭시 불려지는 함수 설정 ---------------

/*
// layout XML 에서 Chart 속성을 넣을 때 itemClickJsFunction을 주고,만든 javascript 함수명을 넣어줍니다
// 예) <Column3DChart showDataTips="true" itemClickJsFunction="chartClick">
//
// 파라메터 설명
// seriesId : layout XML에서 부여한 series의 id가 있을 경우, 해당 id를 보내줍니다.
// displayText : 화면상에 보여주는 dataTip(마우스 오버 시 보여주는 박스-tooltip)의 내용
// index : 클릭된 item(막대나 파이조각등)의 index 번호 - 맨 왼쪽 또는 맨 위 것이 0번입니다
// data : 해당 item의 값을 표현하기 위해 입력된 data(row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)
// values : 해당 item의 값 (배열로 전달되며, 챠트 종류에 따라 아래와 같이 들어옵니다.)
                Bar2D(3D)Series         0:x축값 1:y축값
                Column2D(3D)Series   0:x축값 1:y축값
                Area2DSeries             0:x축값 1:y축값
                Bubble3DSeries          0:x축값 1:y축값 2:radius값
                Line2DSeries             0:x축값 1:y축값
                HLOCSeries               0:x축값 1:open값 2:low값 3:high값 4:close값
                Pie2DSeries              0:값
*/
function chartClick(seriesId, displayText, index, data, values)
{
        alert("seriesId:"+seriesId+"₩ndisplayText:"+displayText+"₩nindex:"+index+"₩ndata:"+data+"₩nvalues:"+valu
es);
}

<!-- 사용자 정의 설정 끝  -->
</script>
</head>

<body onload="rMateChartInit()">
<table>
        <tr>
        <td>
                <script>
                <!--
                        rMateChartCreate("chart1","rMateChart",flashVars, 600, 400, "#FFFFFF");
                // -->
                </script>
        </td>
        </tr>
```

```
</table>
</body>
</html>
```

<예제 85 차트 클릭 시 호출하는 함수 자바스크립트에서 정의하기>

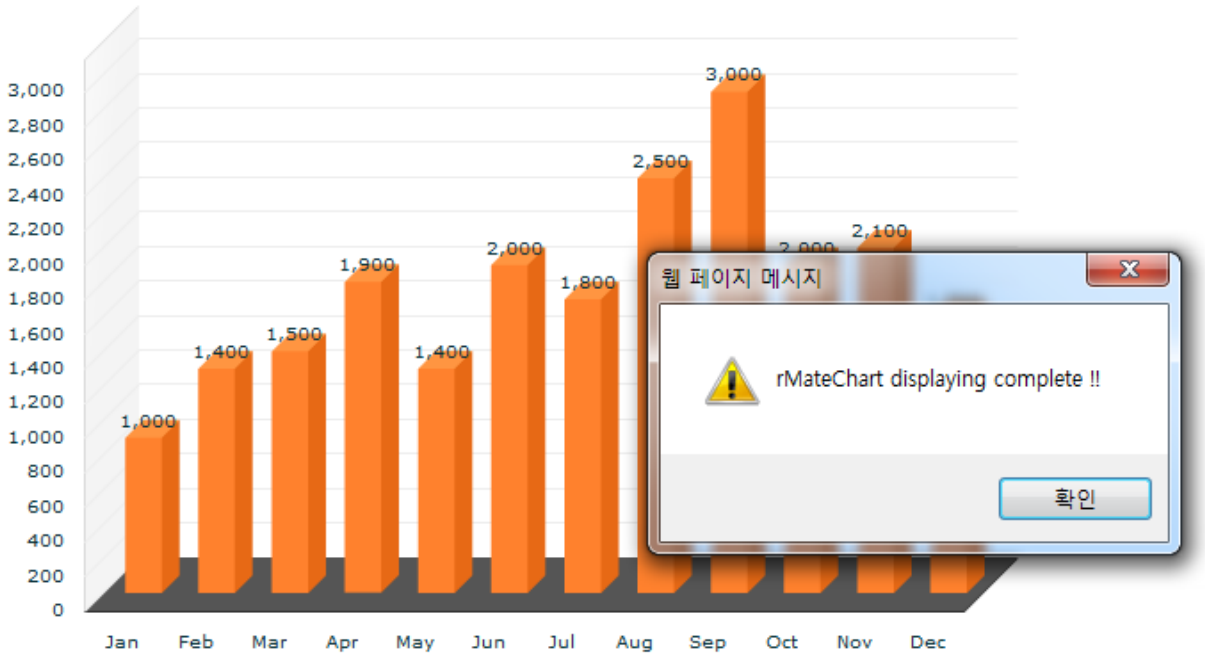
<예제 84 차트 클릭 시 자바스크립트 함수 호출하기> 를 보시면 자바스크립트에서 정의한 함수의 파라메터는 5 개가 있습니다.

파라메터 명	파라메터 설명
seriesId	레이아웃 XML에서 부여한 시리즈(Series)의 id가 있을 경우, 해당 id 입니다.
displayText	화면 상에 보여주는 데이터팁(툴팁, 마우스 오버시 나타나는 팁) 입니다.
index	클릭한 아이템(막대나 파이차트의 조각)의 인덱스 번호입니다. 맨 왼쪽, 맨 위쪽의 값이 0 번입니다.
data	해당 item의 값을 표현하기 위해 입력된 data 입니다. (row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)
values	해당 item의 값 입니다. (배열로 전달되며, 챠트 종류에 따라 아래와 같이 들어옵니다.)

Bar2DSeries(Bar3DSeries)	0 : x축 값, 1 : y축 값
Column2DSeries(Column3DSeries)	0 : x축 값, 1 : y축 값
Area2DSeries	0 : x축 값, 1 : y축 값
Bubble3DSeries	0 : x축 값, 1 : y축 값, 2: radius 값
Line2DSeries	0 : x축 값, 1 : y축 값
Pie2DSeries(Pie3DSeries)	0 : 값
HLOCSeries	0 : x축 값, 1 : open값 2: low값, 3 : high값, 4 : close값

<표 48 차트 클릭 시 호출하는 함수의 파라메터 설명>

지금까지 파이차트의 한 아이템을 클릭 한 경우 자바스크립트의 "chartClick" 함수를 호출하게 설정하였습니다. 이에 대한 실행화면으로 자바스크립트에서 알림창을 띄워 차트로부터 넘어온 파라메터를 보여주고 있습니다.



<그림 51 파이차트 클릭 시 함수 호출 실행화면>

8.11. 사용자 정의 함수 설정하기

사용자 정의 함수를 설정할 수 있는 경우는 총 3 가지가 존재합니다.

1. 데이터팁 사용자 정의
2. 축 라벨 사용자 정의(축에 직접정의, 축렌더러에 정의)
3. 칼럼, 바, 파이계열 차트의 수치필드 사용자 정의

자바스크립트 함수를 정의하고 레이아웃에 그 함수명을 지시하여 차트가 해당 함수를 호출하는 원리로 사용자 정의 함수가 적용됩니다.
축 라벨, 수치필드(데이터팁 제외)를 사용자 정의할 때 반드시 formatter 속성과 함께 사용하지 마십시오. formatter 는 기본적인 사용자 정의 함수입니다**. formatter 속성과 labelJsFunction 속성을 함께 정의한 경우 정상적인 차트를 생성할 수 없습니다.**

8.11.1. 데이터팁(툴팁) 함수 사용자 정의.

챠트에서 showDataTips="true" 설정 후 마우스 오버 시 보이는 데이터팁을 사용자 정의할 수 있습니다.
차트에서 마우스 오버 이벤트가 발생하면 레이아웃에서 미리 지정된 dataTipJsFunction 함수를 호출하여 그 함수가 리턴하는 값을 데이터팁으로 표시하게 됩니다.
레이아웃 XML 에서 dataTipJsFunction 속성을 추가하고 속성값으로 자바스크립트 함수명을 지정하십시오.
이 자바스크립트 함수는 차트 아이템에 마우스 오버 시 rMate 차트가 호출할 콜백함수입니다.

다음 예제는 칼럼 3D 차트의 데이터팁을 사용자 정의한 레이아웃 예제입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFEE"   cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Anual Report"/>
        </Options>
        <!-- dataTipFunc 는 자바스크립트 함수명 입니다. -->
        <Column3DChart showDataTips="true" dataTipJsFunction="dataTipFunc">
                <horizontalAxis>
                        <CategoryAxis categoryField="Month" displayName="날짜"/>
                </horizontalAxis>
                <verticalAxis>
                        <LinearAxis displayName="금액"/>
                </verticalAxis>
                <series>
                        <Column3DSeries id="series1" yField="Profit" displayName="Profit">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                        </Column3DSeries>
                        <Column3DSeries id="series2" yField="Cost" displayName="Cost">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                        </Column3DSeries>
                        <Column3DSeries id="series3" yField="Revenue" displayName="Revenue">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                        </Column3DSeries>
                </series>
        </Column3DChart>
</rMateChart>
```

<예제 86 데이터팁 사용자 정의 레이아웃 예제>

레이아웃에서 자바스크립트 콜백함수를 등록한 후 rMate 차트를 임베딩하는 파일, 예를 들면 html, jsp, php 파일에서 다음과 같이 콜백함수를 실제로 정의하십시오.

```
/*
// ------------------------ 데이터팁 사용자 정의 함수 --------------------------
// 챠트에서 showDataTips="true" 설정 후 마우스 오버 시 보이는 데이터팁 정의
// layout XML 에서 Chart 속성을 넣을때 dataTipJsFunction를 주고, 만든 javascript 함수명을 넣어줍니다
```

```
// 예) <Column3DChart showDataTips="true" dataTipJsFunction를="dataTipFunc">
//
// 파라메터 설명
// seriesId : layout XML에서 부여한 series의 id가 있을 경우, 해당 id를 보내줍니다.
// seriesName : 시리즈의 displayName 으로 정의한 시리즈 이름을 보내줍니다.
// xName : X 축에 displayName 을 정의하였다면 X축의 displayName을 보내줍니다.
// yName : Y 축에 displayName 을 정의하였다면 Y축의 displayName을 보내줍니다.
// data : 해당 item의 값을 표현하기 위해 입력된 data(row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)
// values : 해당 item의 값 (배열로 전달되며, 챠트 종류에 따라 아래와 같이 들어옵니다.)
                Bar2D(3D)Series        0:x축값 1:y축값
                Column2D(3D)Series   0:x축값 1:y축값
                Area2DSeries       0:x축값 1:y축값
                Bubble3DSeries   0:x축값 1:y축값 2:radius값
                Line2DSeries       0:x축값 1:y축값
                HLOCSeries       0:x축값 1:open값 2:low값 3:high값 4:close값
                Pie2D(3D)Series        0:값 1:퍼센트값 2:nameFiled명
*/
function dataTipFunc(seriesId, seriesName, index, xName, yName, data, values)
{
        return "<font Size='11' color='#CC3300'>데이터팁 사용자 정의</font>₩nseriesId:"+seriesId
                        +"<br><font size='11' color='#0000FF'>현재 데이터 : </font>"
                        +"<b>"+seriesName+"</b>"+"₩nitemIndex:"+index
                        +"<br><font size='11' color='#0000FF'>xDisplayName : </font>"+"<b><font size='11'>"+xName+"</font></b>"
                        +"<br><font size='11' color='#0000FF'>yDisplayName : </font>"+"<b><font size='11'>"+yName+"</font></b>"
                        +"₩ndata:"+data+"₩nvalues:"+values;
}
```

<예제 87 자바스크립트 데이터팁 콜백함수 예제>

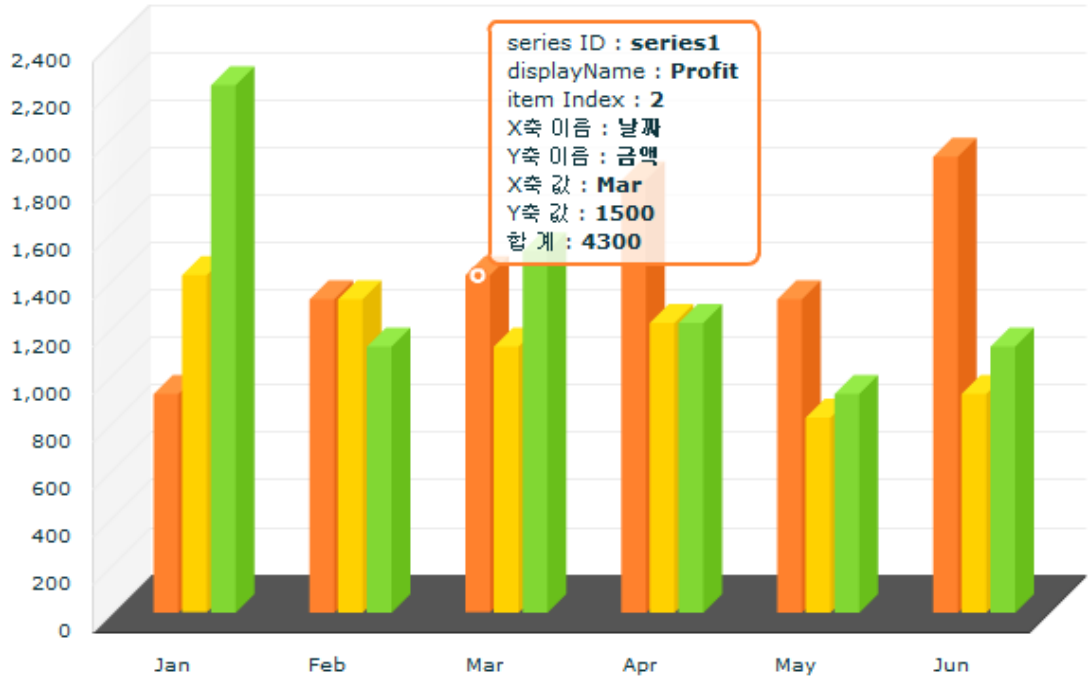
- **데이터팁 사용자 정의는
과 같은 html 태그 사용이 유효합니다.**

다음은 자바스크립트 콜백함수를 정의할 때 함께 정의되어야 할 함수 파라메터에 대한 설명입니다.

파라메터 명	파라메터 설명
seriesId	레이아웃 XML에서 부여한 시리즈(Series)의 id가 있을 경우, 해당 id 입니다.
seriesName	화면 상에 보여주는 데이터팁(툴팁, 마우스 오버시 나타나는 팁) 입니다.
index	클릭한 아이템(막대나 파이차트의 조각)의 인덱스 번호입니다. 맨 왼쪽, 맨 위쪽의 값이 0 번입니다.
xName	X축에 displayName을 정의하였을 경우 나타나는 X축 이름입니다.

<table>
<tr><td>yName</td><td>Y축에 displayName을 정의하였을 경우 나타나는 Y축 이름입니다.</td></tr>
<tr><td>data</td><td>해당 item의 값을 표현하기 위해 입력된 data 입니다. (row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)</td></tr>
<tr><td>values</td><td>해당 item의 값 입니다. (배열로 전달되며, 챠트 종류에 따라 아래와 같이 들어옵니다.)
<table>
<tr><td>Bar2DSeries(Bar3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr>
<tr><td>Column2DSeries(Column3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr>
<tr><td>Area2DSeries</td><td>0 : x축 값, 1 : y축 값</td></tr>
<tr><td>Bubble3DSeries</td><td>0 : x축 값, 1 : y축 값, 2: radius 값</td></tr>
<tr><td>Line2DSeries</td><td>0 : x축 값, 1 : y축 값</td></tr>
<tr><td>Pie2DSeries(Pie3DSeries)</td><td>0 : 값 1:퍼센트 값 2:nameField</td></tr>
<tr><td>HLOCSeries</td><td>0 : x축 값, 1 : open값
2: low값, 3 : high값, 4 : close값</td></tr>
</table></td></tr>
</table>

<표 49 데이터팁 사용자 정의 콜백함수 파라메터 설명>



<그림 52 데이터팁 사용자 정의 실행모습>

8.11.2. 축 라벨 사용자 정의

X, Y 축이 존재하는 데카르트좌표계열 차트의 축에 표시되는 라벨값 사용자 정의는 X 축, Y 축에 해당되는 해당 축 노드에 labelJsFunction 속성을 정의하고 자바스크립트 콜백 함수명을 지정합니다.
축 라벨 사용자 정의는 축에 직접 하는 방법과 축렌더러에 정의하는 방법이 있습니다. 적용방법은 두가지가 같습니다. 축렌더러에 정의한 콜백함수가 늦게 호출됩니다.

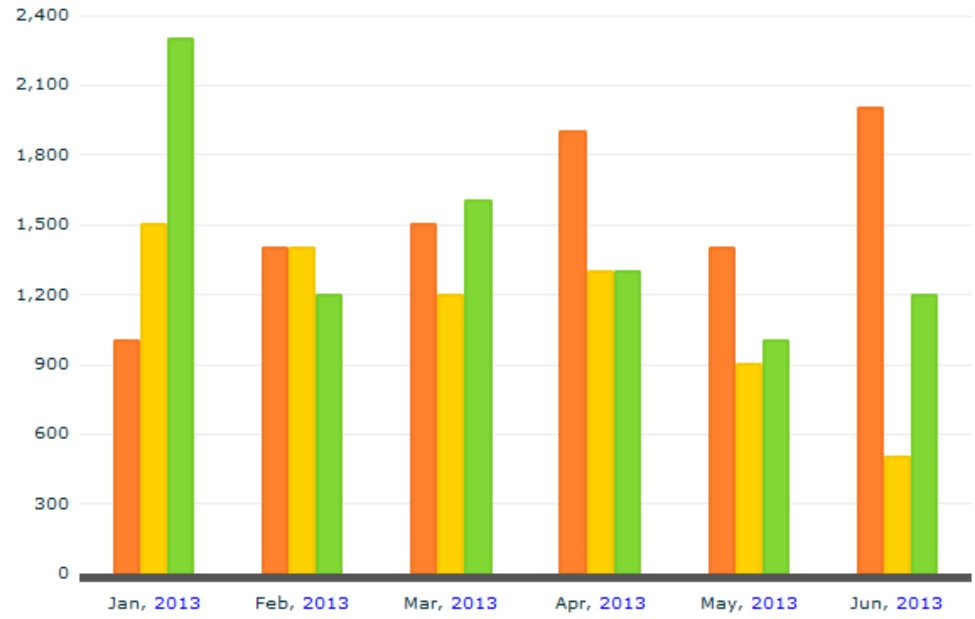
다음은 X 축 라벨을 사용자 정의하기 위해 설정한 레이아웃 예제입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFEE"   cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Anual Report"/>
        </Options>
        <Column3DChart showDataTips="true">
                <horizontalAxis>
                        <!-- axisLabelFunc 는 자바스크립트 함수명 입니다. -->
                        <CategoryAxis categoryField="Month" displayName="날짜"
labelJsFunction="axisLabelFunc"/>
                </horizontalAxis>
                <verticalAxis>
                        <LinearAxis displayName="금액"/>
                </verticalAxis>
                ...
                ...
                ...
        </Column3DChart>
</rMateChart>
```

<예제 88 축 라벨 사용자 정의 레이아웃 예제>

```
/*
// ------------------------- X축 라벨 사용자 정의 함수 -----------------------------
// X, Y 축이 존재하는 데카르트 좌표계열 챠트에서 축 라벨을 사용자 정의 합니다.
// layout XML 에서 축 렌더러 속성을 넣을때 labelJsFunction 주고, 만든 javascript 함수명을 넣어줍니다
// 예)<horizontalAxis>
//                      <CategoryAxis id="hAxis" categoryField="Month" labelJsFunction="axisLabelFunc" />
//              </horizontalAxis>
//
// 파라메터 설명
// id : 축 (or 렌더러 )의 id // 버전 4.5부터 생긴 파라메터 입니다. 이전 버전을 사용하시던 분은 변경하여 주십시오.
// value : 현재 아이템에 맞는 축의 라벨 값
*/
function axisLabelFunc(id,value)
{
        return   "id : "+ id + " : "+ value + ", <font size='11' color='#0000FF'>2009</font>";
}
```

<예제 89 축 라벨 자바스크립트 콜백함수 정의 예제>



<예제 90 축 라벨 사용자 정의 실행화면>

데이터팁 사용자 정의와 마찬가지로 **축라벨 사용자 정의에서 html 태그를 함께 사용할 수 있습니다**.

8.11.3. 수치필드 사용자 정의

수치필드를 표시할 수 있는 칼럼, 바, 파이차트 계열에서 수치필드를 사용자 정의하고자 하는 경우 해당 Series 노드에 labelJsFunction 속성을 추가한 후 자바스크립트 함수명을 속성값으로 지시하십시오.
다음은 칼럼 2D 차트에 수치필드를 사용자 정의한 레이아웃 예제입니다.

```
<series>
        <!-- seriesLabelFunc 는 자바스크립트 함수명 입니다. -->
        <Column2DSeries yField="Profit" displayName="Profit" labelPosition="outside" styleName="seriesStyle"
outsideLabelJsFunction="seriesLabelFunc">
                <showDataEffect>
                        <SeriesInterpolate/>
                </showDataEffect>
        </Column2DSeries>
</series>
```

<예제 91 수치필드 사용자 정의 레이아웃 예제>

```
/*
// ------------------------ 수치 필드 사용자 정의 함수 ----------------------------// 칼럼, 바 챠트에서 labelPosition
속성을 설정한 경우 수치 필드를 사용자 정의하는 함수입니다.
// layout XML 에서 Series 속성을 넣을때 labelJsFunction 주고, 만든 javascript 함수명을 넣어줍니다
//
// 예) <Column2DSeries yField="Profit" labelPosition="outside" outsideLabelJsFunction="seriesLabelFunc">
```

```
//
// 파라메터 설명
// seriesId : 해당 시리즈의 id
// index : 해당 아이템의 인덱스.
// data : 해당 item의 값을 표현하기 위해 입력된 data(row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)
// values : 해당 item의 값 (배열로 전달되며, 챠트 종류에 따라 아래와 같이 들어옵니다.)
                Bar2D(3D)Series       0:x축값 1:y축값
                Column2D(3D)Series   0:x축값 1:y축값
                Pie2D(3D)Series       0:값 1:퍼센트값
                Bubble3DSeries       0:x축값 1:y축값 2:z축값
//
// 참고 : 수치필드 사용자 정의 시엔 <br>태그와 같은 html형식의 코딩 삽입이 불가능합니다.
*/
function seriesLabelFunc(seriesId, index, data, values)
{
        return "$"+values[1];
}
```

<예제 92 수치필드 자바스크립트 콜백함수 예제>

시리즈 명	수치 사용자 정의 함수 명
Column2DSeries Column3DSeries, Bar2DSeries, Bar3DSeries	insideLabelJsFunction, outsideLabelJsFunction
Line2DSeries, Area2DSeries	upLabelJsFunction, downLabelJsFunction
Bubble3DSeries	insideLabelJsFunction
Pie2DSeries, Pie3DSeries,	labelJsFunction

<표 50 시리즈 별 labelJsFunction 명>

위 labelJsFunction 은 labelPosition 의 영향을 받습니다.

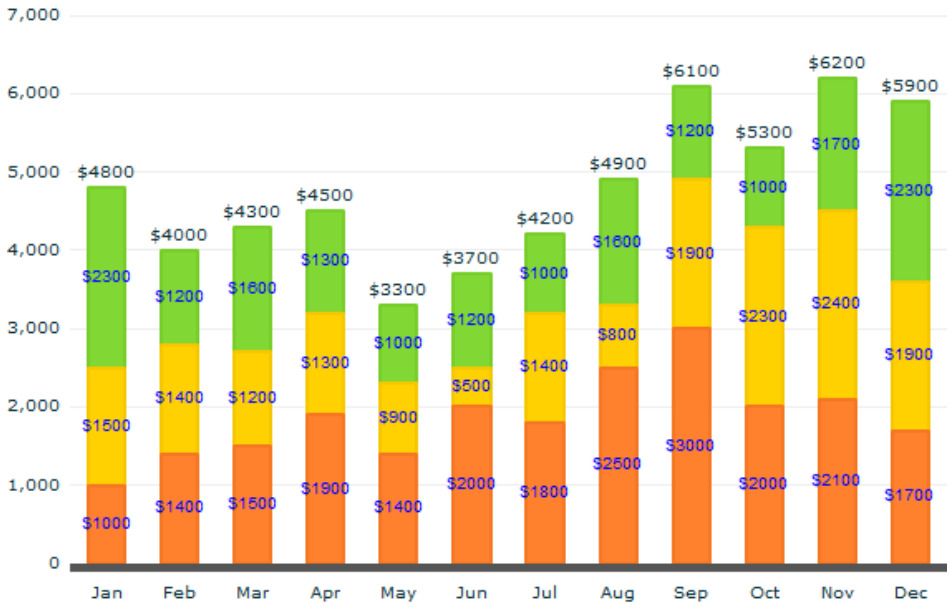
- **수치필드 사용자 정의는
과 같은 html 태그가 적용되지 않습니다.**

다음은 자바스크립트 콜백함수를 정의할 때 함께 정의되어야 할 함수 파라메터에 대한 설명입니다.

파라메터 명	파라메터 설명
seriesId	해당 시리즈의 id 입니다.
index	아이템(막대나 파이차트의 조각)의 인덱스 번호입니다. 맨 왼쪽, 맨 위쪽의 값이 0 번입니다.

<table>
<tr><td>data</td><td>해당 item의 값을 표현하기 위해 입력된 data 입니다. (row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)</td></tr>
<tr><td>values</td><td>해당 item의 값 입니다. (배열로 전달되며, 챠트 종류에 따라 아래와 같이 들어옵니다.)
<table>
<tr><td>BarSeries(Bar3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr>
<tr><td>ColumnSeries(Column3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr>
<tr><td>Pie2DSeries(Pie3DSeries)</td><td>0 : 값 1:퍼센트 값</td></tr>
</table></td></tr>
</table>

<표 51 수치필드 콜백함수 파라메터 설명>



<그림 53 수치필드 사용자 정의 화면>

8.11.4. 채우기 색 사용자 정의

차트의 채우기 색(fill) 을 어떤 조건에 따라 또는 기호에 맞게 사용자 정의하고 하는 경우 해당 Series 노드에 fillJsFunction 속성을 추가한 후 자바스크립트 함수명을 속성값으로 지시하십시오.
다음은 칼럼 3D 차트의 채우기색을 사용자 정의한 레이아웃 예제입니다.

```
<Column3DChart showDataTips="true" gutterLeft="0" showLabelVertically="true">
        <series>
            <Column3DSeries yField="Profit"
            fillJsFunction="fillJsFunc"  styleName="seriesStyle">
                <showDataEffect>
                    <SeriesSlide direction="up" duration="1000"/>
                </showDataEffect>
            </Column3DSeries>
        </series>
```

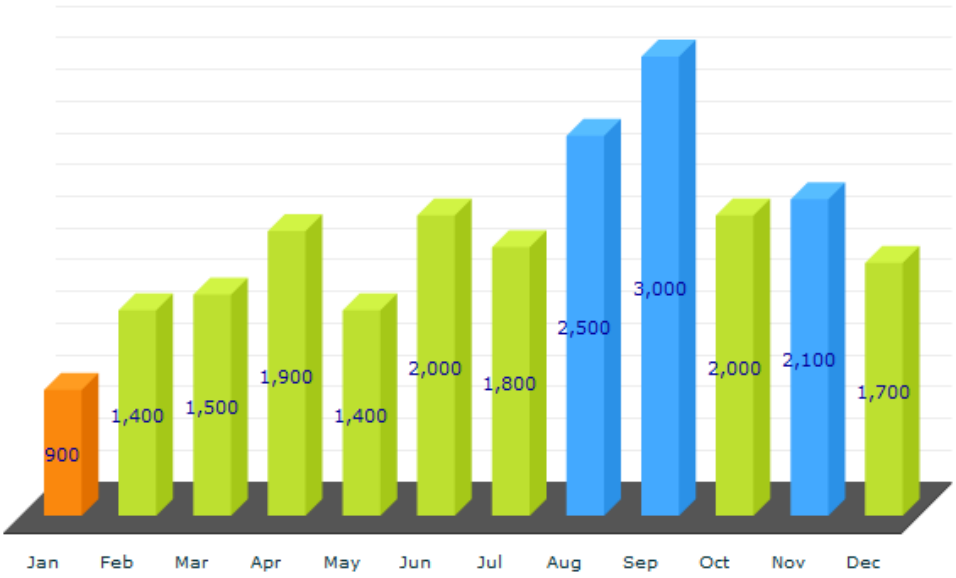
```
...
...
...
</Column3DChart>
```

<예제 93 채우기 색 사용자 정의 레이아웃>

다음은 html, jsp, php, asp 등의 스크립트 파일에서 정의한 자바스크립트 예제입니다.

```
/*
//-------------------- 채우기 색 사용자 정의 함수 ---------------------------------
//
//차트의 채우기 색을 특정 조건에 따라 지정하는 사용자 정의 함수입니다.
//layout XML 에서 Series 속성을 넣을 때 fillJsFunction 을 주고, 만든 javascript 함수명을 넣어줍니다.
//
//예) <Pie2DSeries field="Profit" fillJsFunction="fillJsFunc">
//
//파라메터 설명
//seriesId : 해당 시리즈의 id
//index : 해당 아이템의 인덱스.
//data : 해당 item의 값을 표현하기 위해 입력된 data(row값에 해당 - data로 입력된 종류에 따라 XML의 내용 또는 배열이 됩니다)
//values : 해당 item의 값 (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)
            Bar2D(3D)Series       0:x축값 1:y축값
            Column2D(3D)Series   0:x축값 1:y축값
            Area2DSeries      0:x축값 1:y축값
            Bubble3DSeries   0:x축값 1:y축값 2:radius값
            Line2DSeries      0:x축값 1:y축값
            HLOCSeries      0:x축값 1:open값 2:low값 3:high값 4:close값
            Pie2D(3D)Series       0:값 1:퍼센트값 2:nameField명

                              **
                              * From-To Chart 에서 minField 를 지정했다면 values 의 마지막 인덱스 값에 minField 값이 들어옵니다.
*/
function fillJsFunc(seriesId, index, data, values)
{
          if(values[1] > 2000)
                    return "0xFF3366";
          else if(values[1] > 1000)
                    return "0xFFFF33";
          else
                    return "0xFF9999";
}
```

<예제 94 채우기색 자바스크립트 콜백함수 예제>



<그림 54 채우기색 사용자 정의 화면>

8.12. 상하한선 긋기.

상하한선은 X 축, Y 축이 존재하는 카테시안 계열 차트(예, 칼럼차트 바차트 플롯차트 등등)만 해당되는 속성입니다.
상하한선을 긋는 속성은 Axis2DRenderer(3D 차트인 경우 Axis3DRenderer) 의 속성입니다.
다음은 상하한선과 관련된 속성 및 유효값 설명입니다.

속성명	유효값	설명
targetValue	실제 데이터값의 범위 Number	데이터가 매핑된 축을 바탕으로 해당 value에 선을 그립니다.
targetValueName	String	속성 targetValue 의 라벨 텍스트 기본값을 변경할 수 있는 스타일입니다. 예를 들어 타겟 라인을 1000 에 그렸다면 해당 vlaue 는 1000으로 표시됩니다. 1000을 High Value 로 표현하고자 할 경우 이 스타일을 이용하십시오.
targetSecondValue	실제 데이터값의 범위 Number	데이터가 매핑된 축을 바탕으로 해당 value에 선을 그립니다. 이 속성은 선을 그릴 데이터 2번째 값을 나타냅니다.
targetSecondValueName	String	속성 targetSecondValue 의 라벨 텍스트 기본값을 변경할 수 있는 스타일입니다. 예를

		들어 target second 라인을 1000 에 그렸다면 해당 vlaue 는 500으로 표시됩니다. 500을 Low Value 로 표현하고자 할 경우 이 스타일을 이용하십시오.
targetValueStyleName	Target Value 라벨 스타일 이름	targetValue 라벨의 스타일 이름입니다.
targetSecondValueStyleName	Target Second value 라벨 스타일 이름	targetSecondValue 라벨의 스타일 이름입니다.
targetLineStroke	Stroke 객체	차트 배경에 targetValue 를 참고하여 그리는 라인의 스타일을 정의합니다.
targetSecondLineStroke	Stroke 객체	차트 배경에 targetSecondValue 를 참고하여 그리는 라인의 스타일을 정의합니다.
fillTargetArea	false|true(기본값:false)	targetValue 와 targetSecondValue 를 정의한 경우 안쪽 채우기를 할지를 나타냅니다.
targetAreaFill	SolidColor 객체	targetValue 와 targetSecondValue 를 정의한 경우 안쪽 채우기 색을 정의합니다.

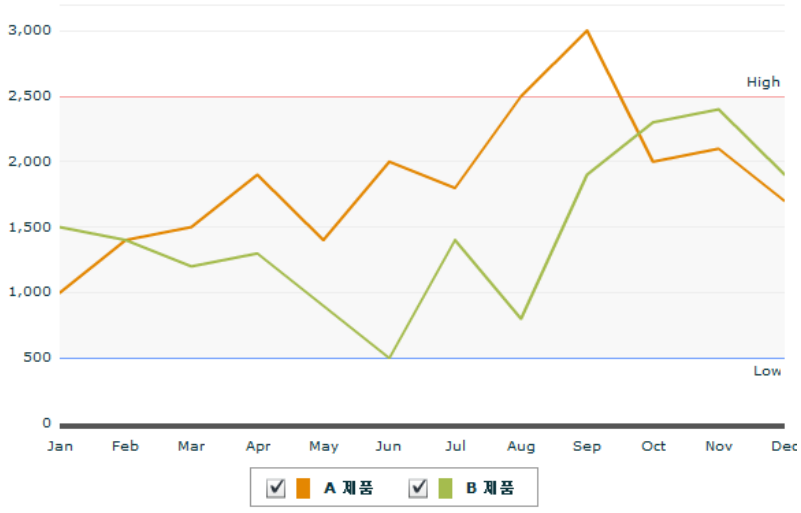
<표 52 상하한선 긋기 속성 설명 표>

예로 라인 차트에 상하한선을 그어보도록 하겠습니다.
먼저 라인 차트의 레이아웃을 정의하면 다음과 같습니다.

```
<rMateChart backgroundColor="0xFFFFEE"  cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Anual Report"/>
                <Legend useVisibleCheck="true" fontSize='12'/>
        </Options>
        <NumberFormatter id="fmt"/>
        <Line2DChart id="chart" showDataTips="true">
                <horizontalAxis>
                        <CategoryAxis id="hAxis" categoryField="Month" padding="0"/>
                </horizontalAxis>
                <verticalAxis>
                        <LinearAxis id="vAxis1" formatter="{fmt}" minimum="0" maximum="3200"
interval="500"/>
                </verticalAxis>
                  <series>
                    ...
                    ...
                    시리즈 정의 부분 생략 ...
                    ...
                    ...
```

```
                </series>

                <verticalAxisRenderers>   // Y축에 상하한선을 긋도록 설정한 부분.
                        // 첫번째 라인(상한선)은 2500 에 긋고 표기는 High 로 표현합니다.
                        // 두번째 라인(하한선)은 500 에 긋고 표기는 Low 로 표현합니다.
                        // 이 때 두 선으로 그려진 영역은 채우기색 적용(fillTargetArea)합니다.
                    <Axis2DRenderer axis="{vAxis1}" placement="left" targetValue="2500"
targetSecondValue="500" targetValueName="High" targetSecondValueName="Low" fillTargetArea="true">
                            <targetLineStroke> // 첫번째 라인 정의
                                <Stroke weight="1" color="0xFF0000"/>
                            </targetLineStroke>
                            <targetSecondLineStroke> // 두번째 라인 정의
                                <Stroke weight="1" color="0x0000FF"/>
                            </targetSecondLineStroke>
                            <targetAreaFill> // 안쪽 채우기 색 정의
                                <SolidColor color="0x00FFFF" alpha="0.1"/>
                            </targetAreaFill>
                    </Axis2DRenderer>
                </verticalAxisRenderers>
        </Line2DChart>
</rMateChart>
```

<예제 95 상하한선 긋기 레이아웃 예제>

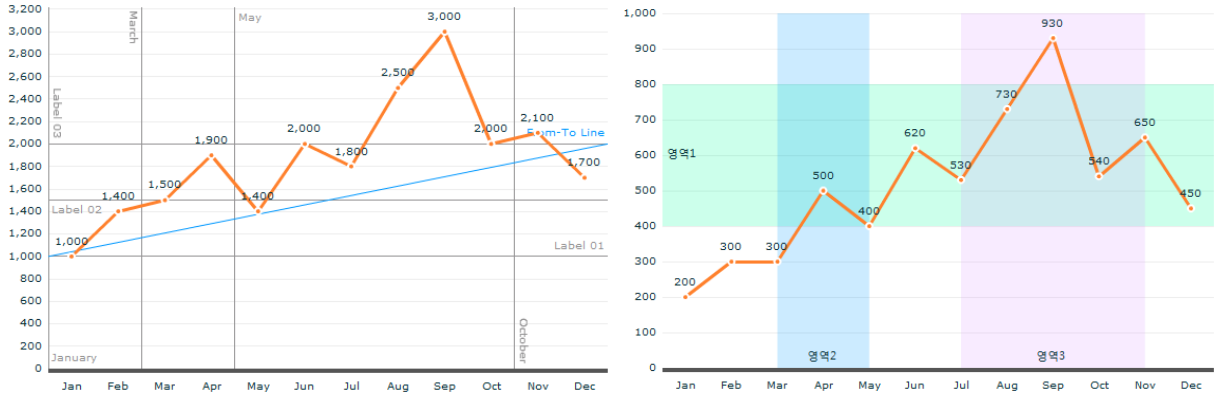


<그림 55 상하한선 긋기 실행화면>

8.13. 차트 안 범위 지정 및 다수의 선 긋기

상하한선은 min/max 를 잡아 영역을 칠하고 선을 긋는 반면 차트 안 범위 지정 및 선 긋기는 범위지정 및 선을 긋는데 제한이 없습니다.

예로 아래 그럼처럼 다수의 선을 긋고 영역 표시를 할 수 있습니다.



<그림 56 차트 안 범위 지정 및 다수의 선 긋기>

위를 표현하기 위한 노드는 다음과 같습니다.

차트의 backgroundElements 또는 annotationElements 속성	선이나 범위를 차트 아래에 표현하고자 하는 경우는 backgroundElements, 차트 위에 나태내고자 하는 경우 annotationElements 속성을 사용합니다.
AxisMarker	AxisLine 과 AxisRange 를 표현할 수 있는 대표자입니다.
AxisLine	라인을 표현할 수 있습니다.
AxisRange	영역을 표현할 수 있습니다.

실제로 차트에 영역과 라인을 표현하는 레이아웃 샘플입니다.

```
<rMateChart backgroundColor="0xFFFFFF">
        <Stroke id="stroke1" color="0xFF0000" weight="1"/>
        <Line2DChart showDataTips="true">


                ......
                  // 차트의 배경에 표시하고 있음.
        <backgroundElements>
                <GridLines/> // 선과 영역포함하여 차트 그리드도 출력하기 위함
                <AxisMarker> // 선과 영역을 포현할 수 있는 대표자 선언


                <lines> // 라인을 긋고자함
                // AxisLine 을 3개 정의하여 3개의 라인을 그음.
```

```
                    <AxisLine value="1000" label="Label 01" stroke="{stroke1}" labelUpDown="up"
color="0xFF0000"/>

                    <AxisLine value="1500" label="Label 02" stroke="{stroke1}" labelAlign="left"
labelUpDown="down" color="0xFF0000"/>

                    <AxisLine value="2000" label="Label 03" stroke="{stroke1}" labelUpDown="up" labelAlign="left"
labelRotation="90" color="0xFF0000"/>
                    </lines>


                    <ranges> // 영역을 표현하고자함.
                    //AxisRange 2개 정의하여 2개의 범위 지정
                    <AxisRange startValue="1000" endValue="2000" label="영역1" fontSize="11"
                    labelHorizontalAlign="left" color="0xFF00FF">
                              <fill>
                                        <SolidColor color="0x00FF99" alpha="0.2"/>
                              </fill>
                    </AxisRange>
                    <AxisRange startValue="Mar" endValue="May" label="영역2" fontSize="11"
labelVerticalAlign="bottom" color="0x0066FF" horizontal="false">
                              <fill>
                                        <SolidColor color="0x0099FF" alpha="0.2"/>
                              </fill>
                    </AxisRange>
                    </ranges>

                    AxisMarker>
                    </backgroundElements>
          </Line2DChart>
</rMateChart>
```

다음은 해당 노드들의 속성 및 유효값 설명입니다.

속성명	유효값	설명
lines	AxisLine 노드	차트에 표현할 라인들을 정의합니다.
ranges	AxisRange 노드	차트에 표현할 영역들을 정의합니다.

AxisRange 와 AxisLine 은 기본적으로 라벨을 출력시킵니다. 따라서 Caption 의 속성을 모두 포함하고 있습니다. 아래 표는 Caption 의 속성 부분은 제외합니다. 이에 대한 속성 설명은 "5.4.1 차트 제목(Caption), 부제목(SubCaption) 넣기." 를 참고하여 주십시오.

AxisLine -

속성명	유효값	설명
labelAlign	center,left,right (기본값:right)	라인에 표시할 라벨의 수평정렬. 속성 horizontal이 false일 경우 left는 top, right는 bottom의 기능을 합니다
labelUpDown	up,down (기본값:up)	라벨을 라인의 위에 표시할지 아래에 표시할지 여부. 속성 horizontal이 false일 경우 up은 우측, down은 좌측에 위치 시킵니다.
labelRotation	0~360(기본값:0)	표시할 라벨의 회전. Embeded폰트를 사용할 경우에는 모든 각도 가능하나 시스템 폰트를 사용할 경우에는 0만 가능합니다.
linePosition	center, left, right	CategoryAxis의 경우 라인을 표시할 위치. CategoryAxis에서는 한개의 값에 대한 선을 그을 수 있는 위치가 여러개가 가능하여 선택할 수 있도록 합니다.
stroke	Stroke 객체	라인의 스타일을 설정합니다.
startValue	Number,String	표시 하는 선이 사선일 경우 라인이 시작되는 위치 값. value값이 설정되면 이 값은 무시됩니다.
endValue	Number,String	표시하려는 선이 사선일 경우 라인이 끝나는 위치 값. value 값이 설정되면 이 값은 무시됩니다.
Value	Number,String	표시하려는 선이 사선이 아닌경우 라인이 표시될 값. 이 값이 설정되면 startValue, endValue값은 무시됩니다.
label	String	선에 표시할 텍스트를 정의하십시오.

<표 53 AxisLine 속성 및 유효값 설명>

AxisRange -

속성 명	유효 값	설 명
labelHorizontalAlign	String(center,left,right)	범위에 표시할 라벨의 수평정렬입니다.
labelVerticalAlign	String(top,middle,bottom)	범위에 표시할 라벨의 수직정렬입니다.
labelRotation	Number	표시할 라벨의 회전 각도입니다. Embeded폰트를 사용할 경우에는 모든 각도 가능하나 시스템 폰트를 사용할 경우에는 0, 90만 가능합

		니다.
fill	Uint	범위안을 채울 색상을 지정합니다.
startValue	Number, String	표시하려는 범위의 위치 시작 값.
endValue	Number, String	표시하려는 범위의 위치 종료 값.
horizontal	Boolean(true, false)	수평범위 여부. 수평범위(true)이면 세로축을 기준으로 범위를 그리게 되며 false이면 수직범위를 그리게 됩니다. true이면 수직좌표의 값이 들어가야 하며 false이면 수평좌표의 값이 들어가야 합니다.
label	String	범위 안에 넣을 텍스트를 정의하십시오.

<표 54 AxisRange 속성 및 유효값 설명>

8.14. 내장폰트 사용하기.

rMate 차트는 기본적으로 Adobe Style Explorer 에서 제공하는 Myriad Web Pro 폰트를 내장하고 있습니다.
내장 폰트를 사용하기 위해서는 fontFamily 스타일을 Myriad 로 정의하십시오.
모든 문자를 내장하지 않고 다음과 같은 문자만을 제공합니다.

제공되는 문자	유니코드
[0..9]	U+0030-U+0039
:	U+003A
!, ”, #, $, %, ’, (,), *, +,, ,- ,. ,/	U+0021-U+002F

<표 55 내장폰트 제공되는 글자 리스트>

만약 Myriad 폰트를 fontFamily 로 정의한 후 위 리스트에 표시되지 않은 문자를 출력하였다면 그 글자는 정상적으로 출력되지 않습니다. 즉, 영문자나 한글과 같은 문자는 표시되지 않습니다.
내장폰트를 사용하면 글자를 회전하여 출력하거나, 깔끔한 문자처리가 가능합니다.

다음은 내장폰트를 사용하여 축 라벨을 회전한 예제입니다.

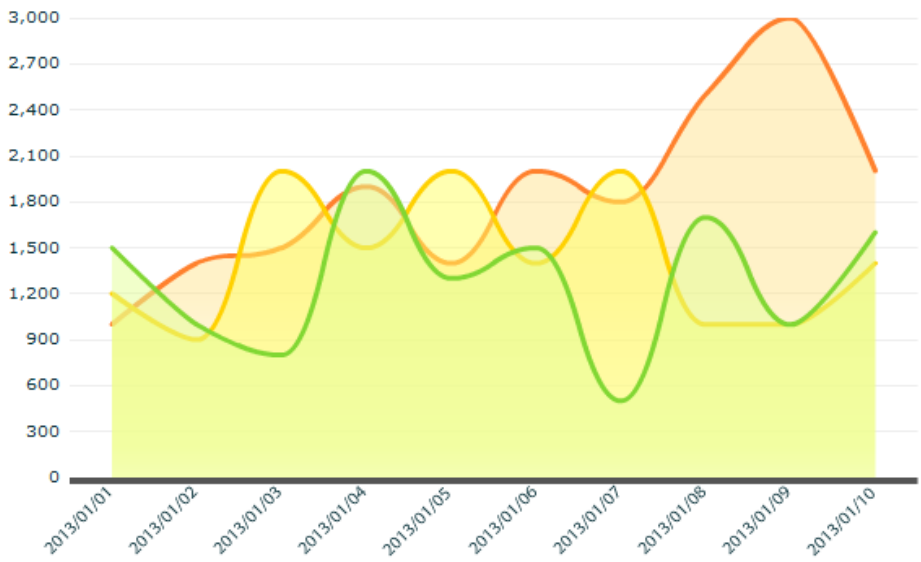
```
<?xml version="1.0" encoding="utf-8"?>
```

```
<rMateChart backgroundColor="0xFFFFEE"  cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="X축 라벨 회전하여 출력하기"/>
                <SubCaption text="45도 각도 회전하여 출력한 모습입니다." textAlign="right" fontSize="11"/>
        </Options>
        <Area2DChart showDataTips="true">
                <horizontalAxis>
                        <CategoryAxis id="hAxis" categoryField="Date" />
                </horizontalAxis>
                <series>
                        <Area2DSeries yField="Data1" form="curve" displayName="Profit">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                                <areaStroke>
                                        <Stroke weight="1" color="0xE48701"/>
                                </areaStroke>
                                <areaFill>
                                        <SolidColor color="0xE48701" alpha="0.25"/>
                                </areaFill>
                        </Area2DSeries>
                        <Area2DSeries yField="Data2" form="curve" displayName="Cost">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                                <areaStroke>
                                        <Stroke weight="1" color="0xA5BC4E"/>
                                </areaStroke>
                                <areaFill>
                                        <SolidColor color="0xA5BC4E" alpha="0.25"/>
                                </areaFill>
                        </Area2DSeries>
                        <Area2DSeries yField="Data3" form="curve" displayName="Revenue">
                                <showDataEffect>
                                        <SeriesInterpolate />
                                </showDataEffect>
                                <areaStroke>
                                        <Stroke weight="1" color="0x1B95D9"/>
                                </areaStroke>
                                <areaFill>
                                        <SolidColor color="0x1B95D9" alpha="0.25"/>
                                </areaFill>
                        </Area2DSeries>
                </series>
```

```
                <horizontalAxisRenderers>
                        <Axis2DRenderer axis="{hAxis}" styleName="xAxisStyle"/>
                </horizontalAxisRenderers>
        </Area2DChart>
        <Style>
        .xAxisStyle {
                fontFamily:Myriad;
                fontSize:11;
                labelRotation:45;
        }
        </Style>
</rMateChart>
```

<예제 96 내장폰트 사용한 레이아웃 예제>

위 레이아웃을 출력한 화면은 다음과 같습니다.



<그림 57 내장폰트 사용한 차트 출력모습>

8.15. 사용자가 만든 외부폰트 사용하기

기본 내장폰트는 Myriad 입니다. 이 폰트를 사용해야만 라벨을 회전하거나 깔끔한 처리를 할 수 있습니다. 그러나 사용자가 알메이트용 폰트를 제작하여 차트에 삽입한다면 사용자가 지정한 폰트를 회전시킬 수 있습니다.(즉, Myriad 와 같은 효과를 출력할 수 있음)

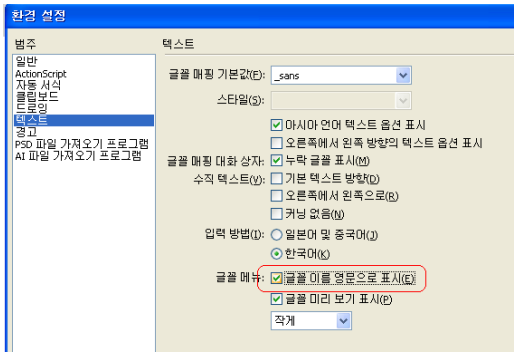
다음은 알메이트용 차트를 제작하는 방법입니다.

1. 플래시를 실행시킵니다.(본 문서 설명은 CS5 기준입니다.)

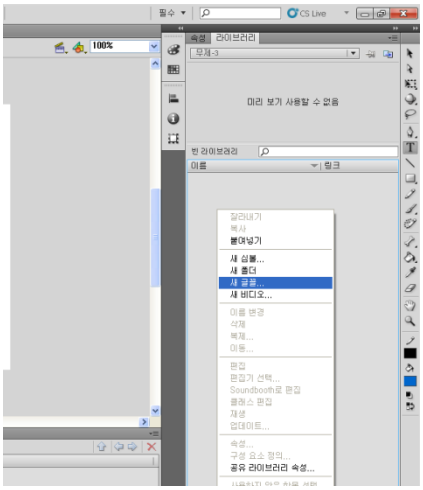
2. 새로 만들기에서 ActionScript 3.0 문서를 선택합니다.

3. ActionScript 3.0 새 문서 만든 후 가장 먼저 상단 탭 메뉴의 편집 -> 환경설정을 클릭합니다.

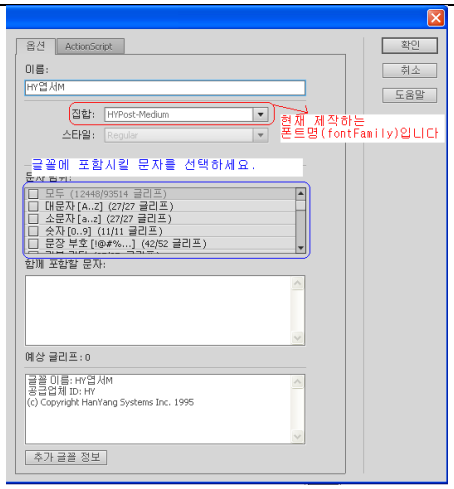
4. 아래 스샷과 같이 텍스트 선택 후 “글꼴 이름 영문으로 표시” 선택 후 확인 누릅니다.



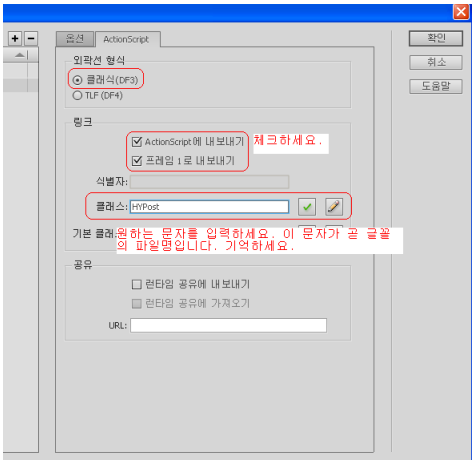
5. 메인화면의 오른쪽 상단에서 라이브러리 선택 후 아래 스샷과 같이 해당 영역에서 오른쪽 버튼을 눌러 “새 글꼴”을 선택합니다.



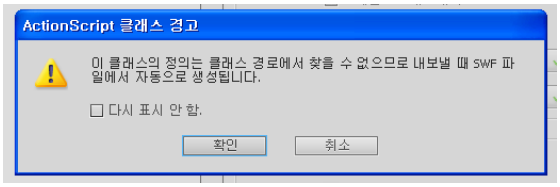
6. 글꼴 포함 창에서 원하는 글꼴과 포함시킬 문자를 선택하세요.(스샷 참고)
   - **집합에 나타난 문자가 곧 폰트명이 됩니다. 알메이트 제품군의 레이아웃에서 여기서 나타난 문자를 fontFamily 속성값으로 할당하십시오.**



7. 옵션 옆의 ActionScript 탭을 선택하여 이동하세요.
   스샷처럼 체크, 설정하십시오.
   클래스는 원하는 문자를 입력하되 후에 내보내기(Export) 할 때 여기서 적은 문자와 같은 파일명으로 내보내기 해야 합니다.



8. 확인을 눌르십시오.
   아래 스샷과 같이 나오면 확인을 누르고 빠져나가세요.



9. 모든 준비가 끝났습니다.

10. 상단의 탭 메뉴에서 “파일” -> “내보내기” -> “동영상 내보내기” 선택하세요.

11. 파일명은 글꼴창에서 입력한 클래스명과 같이 넣어주세요. 이 문서에서는 HYPost 라고 적었기 때문에 내보내기 파일명 역시 HYPost 입니다.

12. 저장 후 HYPost.swf 가 생성된 것을 알 수 있습니다.

- 이 문서에서는 'HY엽서M' 이라는 폰트를 알메이트에서 사용할 수 있게 하나 만들었습니다.
  알메이트에서 이 폰트에 접근할 수 있는 폰트명(fontFamily)은 집합명이였던 HYPost-Medium 입니다.

<예제 97 알메이트용 폰트 만들기>

위처럼 HYPost.swf 폰트 파일을 flashVars 변수에 다음과 같이 등록합니다.

```
// 해당 폰트는 차트 컴포넌트 swf 가 있는 같은 도메인에 위치해야 합니다.
// 이런 이유로 절대 경로는 지원하지 않으며 상대 경로만 지원합니다.
var fontURL =encodeURIComponent("./Fonts/ HYPost.swf");

// 해당 폰트가 있는 곳의 URL 입니다.
flashVars += "&fontURL=" + fontURL;
```

<예제 98 flashVars 폰트 등록 예제>

위 처럼 등록하면 레이아웃에서 fontFamily 속성에 HYPost-Medium 를 설정하여 HY 엽서-M 폰트를 이용하여 회전출력을 할 수 있습니다.

8.16. 확대/축소와 마우스 이동에 따른 십자가 표시하기

확대/축소 및 십자가 표시 기능은 아래 리스트를 제외한 모든 차트에서 사용 가능합니다.

- 레이더 차트
- 파이, 도넛 차트
- 히스토리, 스크롤 차트
- 게이지 차트

확대/축소와 십자가 표시는 CrossRangeZoomer 노드의 정의로 사용이 가능합니다.

다음과 같이 레이아웃을 작성하고 차트의 annotationElements 속성에 CrossRangeZoomer 를 설정하십시오.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart>
<Combination2DChart showDataTips="true">
  <series>
    <Line2DSeries yField="Profit" displayName="Profit"/>
  </series>
  <annotationElements>
    <CrossRangeZoomer zoomType="horizontal" fontSize="11" color="0x00000"
      verticalLabelPlacement="bottom" enableZooming="true" enableCrossHair="true">
      <zoomRangeStroke>
        <Stroke weight="1" color="0xFF0000" alpha="0.3"/>
      </zoomRangeStroke>
      <zoomRangeFill>
        <SolidColor color="0x0000FF" alpha="0.2"/>
      </zoomRangeFill>
    </CrossRangeZoomer>
  </annotationElements>
</Combination2DChart>
</rMateChart>
```

<예제 99 확대/축소 및 십자가 표시 예제>

속 성	유효값	설 명
horizontalStroke	Stroke 객체	십자가의 가로선을 지정합니다.
verticalStroke	Stroke 객체	십자가의 세로선을 지정합니다.
zoomRangeStroke	Stroke 객체	줌 영역의 선 스타일을 지정합니다.
zoomRangeFill	SolidColor 객체	줌 영역의 채우기 색을 지정합니다.
horizontalLabelFormatter	포매터 id	십자가의 가로선에 나타나는 라벨에 적용할 포맷터.
horizontalLabelOppFormatter	포매터 id	십자가의 가로선, 오른쪽에 나타나는 라벨에 적용할 포맷터(useDualCrossXLabel="true" 인 경우 유효)
horizontalLabelPlacement	left| right(기본값:left)	십자가의 가로선 라벨의 위치를 나타냅니다. left일 경우 Chart 왼쪽에 right일 경우 오른쪽에 좌표값이 표시됩니다.
showValueLabels	true| false (기본값:true)	십자가의 라벨에 표시되는 좌표 값 출력 여부입니다. (현재 마우스가 위치한 값 표시여부.)
verticalLabelFormatter	포매터 id	십자가의 세로선에 나타나는 라벨에 적용할 포맷터.

verticalLabelOppFormatter	포매터 id	십자가의 세로선, 윗쪽에 나타나는 라벨에 적용할 포맷터(useDualCrossYLabel="true" 인 경우 유효)
verticalLabelPlacement	top|bottom (기본값:bottom)	십자가의 세로선 라벨의 위치를 나타냅니다. top일 경우 Chart 위에 bottom일 경우 아래에 좌표값이 표시됩니다.
useDualCrossXLabel	false|true (기본값:false)	수직축을 2개 이상 설정한 경우 2개의 축에 대하여 라벨을 표시할지 여부를 나타냅니다. 즉, useDualCrossXLabel 를 true 로 설정한 경우, 수직축에 해당되는 라벨이 축 위치에 맞게 표시됩니다. (horozontalLabelPlacement 속성은 무시됨) 예를 들어 수직축을 3개 설정하였다면 앞에서 2개의 축에 대한 라벨만 표시됩니다.
useDualCrossYLabel	false|true (기본값:false)	수평축을 2개 이상 설정한 경우 2개의 축에 대하여 라벨을 표시할지 여부를 나타냅니다. 즉, useDualCrossYLabel 를 true 로 설정한 경우, 수평축에 해당되는 라벨이 축 위치에 맞게표시됩니다. (verticalLabelPlacement 속성은 무시됨) 예를 들어 수평축을 3개 설정하였다면 앞에서 2개의 축에 대한 라벨만 표시됩니다.
enableCrossHair		마우스 포인터를 기준으로 십자가 사용 여부를 나타냅니다.
enableZooming		Chart의 Zoom 기능 사용 여부를 나타냅니다.
zoomType	horizontal, vertical, both (기본값:both)	Chart의 Zoom 기능 사용 시 zoom 기능을 수직축, 수평축, 모두에 적용할지를 지정합니다. 예를 들어 zoomType 을 "horizontal" 로 지정하고 zoom in 을 수행했다면 수직축에 대해서는 zoom in 을 적용하지 않고 수평축에 대해서만 기능을 수행합니다.
resetMode	initial, auto(기본값: auto)	줌 기능에서 Restore 실행 방식을 결정합니다. initial 인 경우 사용자에 의해 결정된 최초의 minimum, maximum 값을 유지하고, auto 인 경우 차트에 의해 minimum, maximum 이 정해집니다. 개발자에 의해 축의 minimum, maximum 설정 값이 유지되도록 하기 위해서는 initial 값을 할당하세요.

<표 56 CrossRangeZoomer 속성 및 유효값 설명표>

8.17. 차트 안에 메모 표시하기.

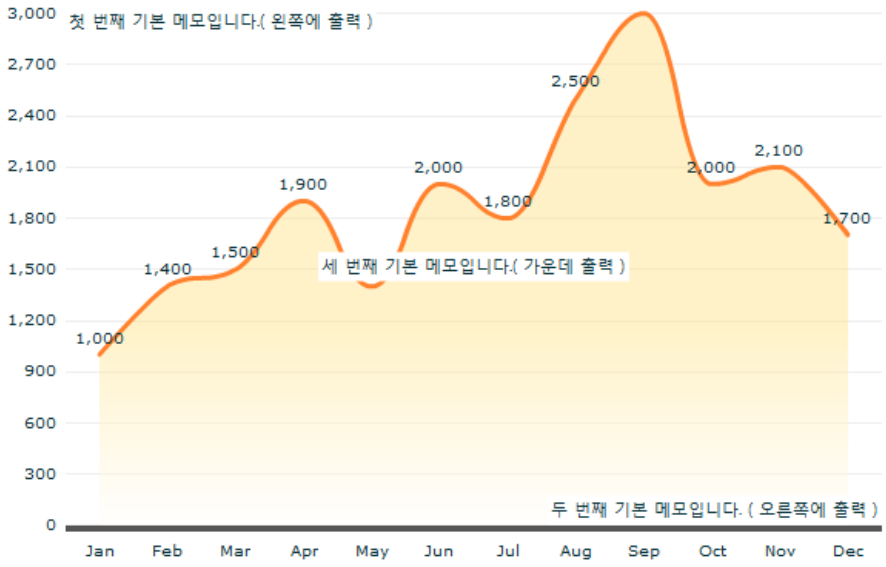
차트 안에 메모표시는 특정 데이터 기준이 아닌 차트 안쪽의 x,y 좌표계를 이용하여 메모를 표시하게 됩니다. 차트 위에 메모를 출력하려면 차트의 annotationElements 속성을, 차트 밑에 메모를 출력하려면 backgroundElements 속성을 사용하십시오.
본문의 예제는 차트 위에 메모를 표시해 보도록 하겠습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart>
  <Line2DChart showDataTips="true">
    <series>

    ..... 라인 시리즈 설정 부분 중략

    </series>
    <annotationElements>

      <CanvasElement> <!—그룹으로 설정할 라벨들을 묶습니다. →

        <!—라벨 1→
        <Label left="0" top="20" width="200" height="20" color="0x0000FF" fontSize="15" textAlign="center"
            text="텍스트 쓰기 1(왼쪽)" backgroundAlpha="0" borderStyle="solid" borderColor="0xFF0000">
        </Label>
        <!—라벨 2→
        <Label width="200" height="40" color="0x0000FF" fontSize="15" textAlign="center" verticalCenter="0"
            horizontalCenter="0" text="텍스트 쓰기 2&#xa;(차트 중앙 표시)" backgroundColor="0xFFFF00"
            backgroundAlpha="0.2" borderStyle="solid" borderColor="0xFF0000">
          <filters>
              <DropShadowFilter/>
          </filters>
        </Label>

        <!—라벨 3→
        <Label right="0" bottom="0" height="40" color="0xFF0000" fontSize="15" textAlign="center"
              text="텍스트 쓰기 3(오른쪽)" backgroundAlpha="0" borderStyle="none">
          <filters>
              <DropShadowFilter/>
          </filters>
        </Label>

        <Label left="50" top="100" height="40" color="0x009966" fontSize="12" textAlign="center" text="임의의
        X,Y 좌표(50,100)에 표시" backgroundAlpha="0" borderStyle="none">
```

```
            </Label>
        </CanvasElement>
    </annotationElements>
</Line2DChart>
</rMateChart>
```

<예제 100 차트 안에 메모 표시 예제>

위처럼 정의한 메모들이 출력된 모습은 아래와 같습니다.



<그림 58 차트 안에 메모 표시한 모습>

- 라벨의 속성 및 유효값에 대한 설명은 “5.4.1 차트 제목(Caption), 부제목(SubCaption) 넣기.” 를 참고하여 주십시오.

다음은 라벨의 고유 속성(즉, 제목, 부제목 속성이 아닌 라벨 속성)을 나타낸 표입니다.

속 성	유효값	설 명
backgroundColor	RGB 컬러	바탕의 색을 지정합니다.
backgroundAlpha	0~1 실수	바탕색 투명도를 지정합니다.
borderColor	RGB 컬러	테두리 선 색깔을 지정합니다.
borderAlpha	0~1 실수	테두리 선 투명도를 지정합니다.
borderThickness	Number(픽셀 단위)	테두리 두께를 지정합니다.
cornerRadius	Number	테두리 코너를 둥그렇게 합니다.
borderStyle	"solid", "none"	테두리 선을 넣을지 결정합니다.

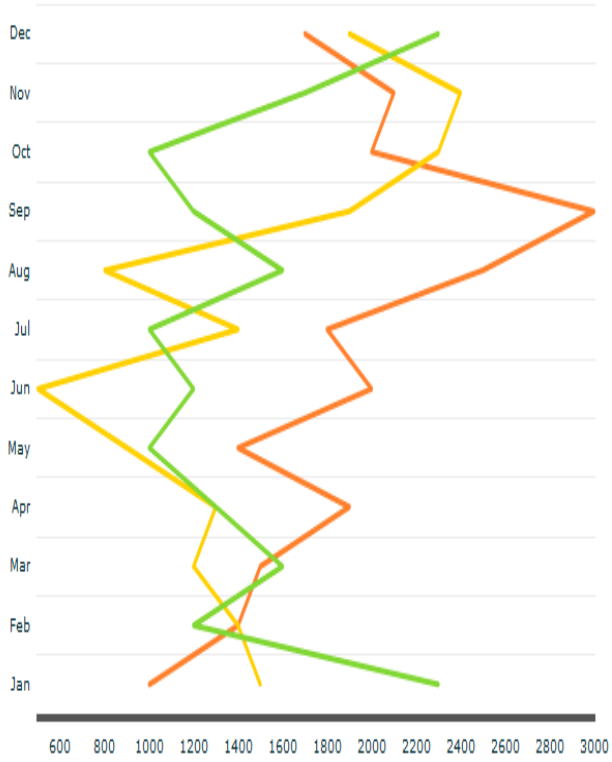
< 57 차트에 라벨 표시 속성 유효값 및 설명>

8.18. 라인차트에서 수직선으로 데이터 출력하기.

라인차트는 기본적으로 수평축을 수치데이터로 삼고 수평방향으로 선 그래프가 진행되는 방식입니다. 라인차트의 선이 수직방향으로 수치 데이터를 표현하고자 할 때는 다음과 같이 하십시오.

- Line2DChart 의 horizontalAxis 의 카테고리축을 verticalAxis 로 변경하십시오.
- Line2DSeries 의 xField 와 yField 의 값을 반대로 하십시오.
- Line2DSeries 의 sortOnXField 를 false 로 설정합니다.

```
<rMateChart>
  <Line2DChart>
    <verticalAxis>
        <CategoryAxis categoryField="Month"/>
    </verticalAxis>
    <series>
        <Line2DSeries xField="Profits" yField="Month"
sortOnXField="false" displayName="Profits">
        </Line2DSeries>
        <Line2DSeries xField="Costs" yField="Month"
sortOnXField="false" displayName="Costs">
        </Line2DSeries>
         <Line2DSeries xField="Revenue"
yField="Month" sortOnXField="false"
displayName="Revenue">
         </Line2DSeries>
        </series>
  </Line2DChart>
</rMateChart>
```

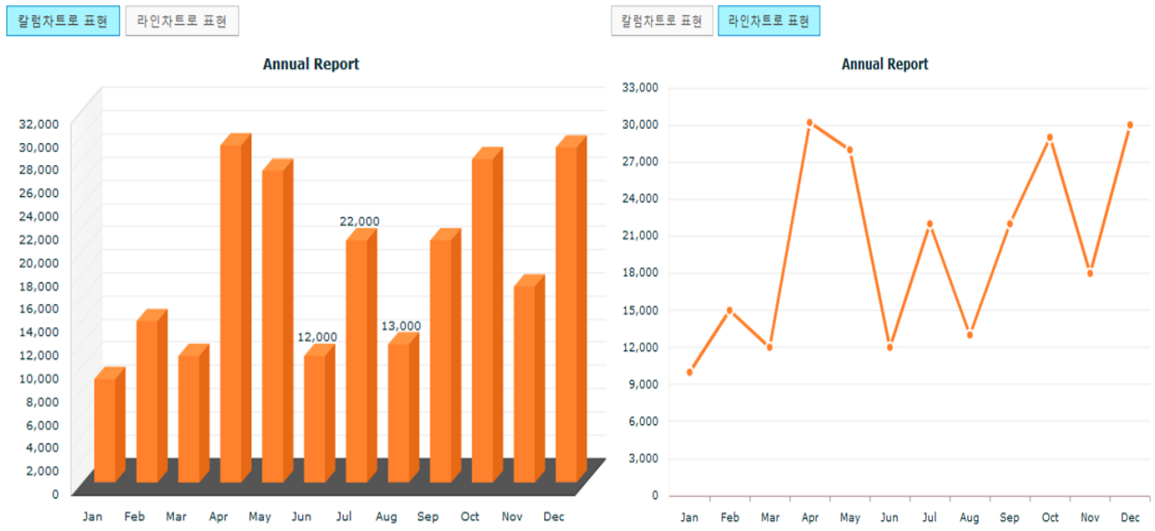


<예제 101 라인차트에 데이터 수직으로 출력 예제>

8.19. 동적으로 차트 레이아웃 또는 데이터 변경하기.

때때로 이미 출력된 차트의 데이터를 동적으로 변경시키거나 칼럼차트로 표현된 데이터를 라인차트나 바 차트로 표현해야 할 경우가 있습니다. 이런 경우 rMate 차트는 웹 페이지를 갱신하지 않고 동적으로 차트에 데이터나 레이아웃을 변경시킬 수 있습니다. 아래 예제는 기본적으로 데이터 1 을 라인차트로 출력합니다. Html 문서 안의 스크립트에서 만든 버튼으로 칼럼차트로 데이터 1 을 표현하거나 데이터 2 를 표

현하는 방법을 나타낸 예제입니다. rMate 차트 swf 파일은 생성하고 하는 차트에 맞는 swf 파일을 선택해야 합니다. 그러나 이와 같이 동적으로 레이아웃을 변경하였을 때 그에 맞는 차트를 생성해야 하기 때문에 rMateIntegration.swf 를 활용합니다. 통합 차트는 모든 차트를 생성할 수 있습니다.



<그림 59 동적으로 레이아웃을 바꿔 출력한 결과>

동적으로 바뀌는 차트를 작성하는 방법을 제시한 html 문서는 아래와 같습니다.

```
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />

<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">


// ----------------------차트  flashVars 설정 시작----------------------
// rMate 차트와 스크립트 간의 동기화가 완료된 후 호출되는 함수를 지시하여
// 이 함수를 통하여 레이아웃과 데이터를 차트에 입력합니다.
// 따라서 레이아웃과 데이터의 XML 경로를 입력하지 않습니다.
var flashVars;

//동기화가 완료 되었을 때 호출 할 함수를 정의 하십시오.
var rMateOnLoadCallFunction = "rMateChartOnLoad";
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;
// ----------------------차트  flashVars 설정 끝----------------------
```

```
// rMate 차트와  스크립트 간의 동기화가 완료되면 호출하는 함수입니다.
// 사용자는 이 함수를 flashVars에 반드시 등록해야 합니다.
// 여기서 작업을 하십시오.
// 차트 콜백함수 4개 존재합니다.
// 1. setLayout - 스트링으로 작성된 레이아웃 XML을 삽입합니다.
// 2. setData - 배열로 작성된 데이터를 삽입합니다.
// 3. setLayoutURL - 레이아웃 XML 경로를 지시합니다.
// 4. setDataURL - 데이터 XML 경로를 지시합니다.
// 아래 예제는 가능한 방법을 제시한 것 입니다.
// 현제 2번 차트는 레이아웃을 스트링 형태로 받아들이고 데이터는 배열 형태로 받아들입니다.
function rMateChartOnLoad()
{
	//예제1. 레이아웃 스트링 형태,데이터 배열 형태
	chart.setLayout(layoutStr);
	chart.setData(chartData);

	//예제2. 레이아웃 XML URL경로, 데이터 배열 형태
	//chart.setLayoutURL(layoutURL);
	//chart.setData(chartData);

	//예제3. 레이아웃 스트링 형태, 데이터 XML URL경로
	//chart.setLayout(layoutStr);
	//chart.setDataURL(dataURL);

	//예제4. 레이아웃 URL, 데이터 URL
	//chart.setLayoutURL(layoutURL);
	//chart.setDataURL(dataURL);
}

// 동적으로 할당할 레이아웃 정의.
var layoutURL = encodeURIComponent("chartLayout.xml");
var layoutStr = "<rMateChart cornerRadius='12' borderStyle='solid'>"
	+"<Options><Caption text='Annual Report'/></Options>"
	+"<Line2DChart showDataTips='true'>"
	+"<horizontalAxis><CategoryAxis categoryField='Month'/></horizontalAxis>"
	+"<series><Line2DSeries yField='Profit' displayName='Profit'>"
	+"<showDataEffect><SeriesInterpolate/></showDataEffect>"
	+"<lineStroke><Stroke color='0xFF0000'weight='4'/>"
	+"</lineStroke></Line2DSeries>"
	+"</series></Line2DChart></rMateChart>";


var chartData = [{"Month":"Jan", "Revenue":10000, "Cost":5000, "Profit":5000},
```

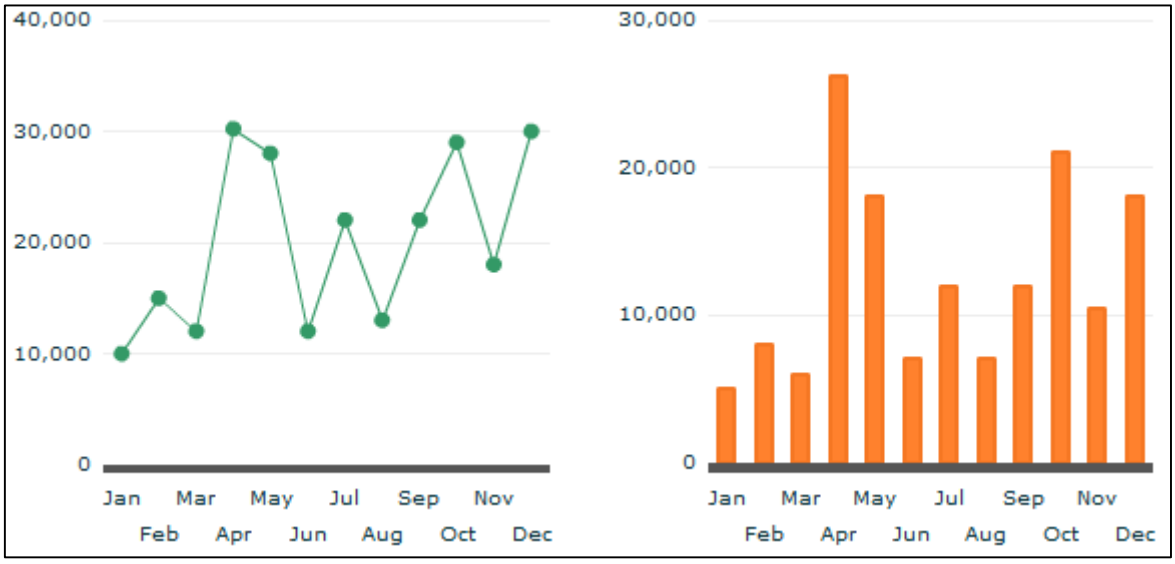
```
            {"Month":"Feb", "Revenue":15000, "Cost":7000, "Profit":8000},
            {"Month":"Mar", "Revenue":12000 , "Cost":6000, "Profit":6000},
            {"Month":"Apr", "Revenue":30200, "Cost":4000, "Profit":26200},
            {"Month":"May", "Revenue":28000, "Cost":10000, "Profit":18000},
            {"Month":"Jun", "Revenue":12000, "Cost":5000, "Profit":7000},
            {"Month":"Jul", "Revenue":22000, "Cost":10000, "Profit":12000},
            {"Month":"Aug", "Revenue":13000, "Cost":6000, "Profit":7000},
            {"Month":"Sep", "Revenue":22000, "Cost":10000, "Profit":12000},
            {"Month":"Oct", "Revenue":29000, "Cost":8000, "Profit":21000},
            {"Month":"Nov", "Revenue":18000, "Cost":7500, "Profit":10500},
            {"Month":"Dec", "Revenue":30000, "Cost":12000, "Profit":18000} ];

var chartData2 = [{"Month":"Jan", "Revenue":1000, "Cost":500, "Profit":500},
            {"Month":"Feb", "Revenue":1500, "Cost":700, "Profit":800},
            {"Month":"Mar", "Revenue":1200 , "Cost":600, "Profit":600},
            {"Month":"Apr", "Revenue":3020, "Cost":400, "Profit":2620},
            {"Month":"May", "Revenue":2800, "Cost":1000, "Profit":1800},
            {"Month":"Jun", "Revenue":1200, "Cost":500, "Profit":700},
            {"Month":"Jul", "Revenue":2200, "Cost":1000, "Profit":1200},
            {"Month":"Aug", "Revenue":1300, "Cost":600, "Profit":700},
            {"Month":"Sep", "Revenue":2200, "Cost":1000, "Profit":1200},
            {"Month":"Oct", "Revenue":2900, "Cost":800, "Profit":2100},
            {"Month":"Nov", "Revenue":1800, "Cost":750, "Profit":1050},
            {"Month":"Dec", "Revenue":3000, "Cost":1200, "Profit":1800} ];

// 레이아웃 변경 함수
function changeLayout()
{
    chart.setLayoutURL(layoutURL);
}

// 데이터 변경  함수
function changeData()
{
    chart.setData(chartData2);
}

// 레이아웃, 데이터 모두 변경  함수.
function changeBoth()
{
    chart.setLayout(layoutStr);
    chart.setData(chartData);
}
```

```
</script>
<!-- 사용자 정의 설정 끝  -->
</head>

<body onload="rMateChartInit()">
<table>
	<tr>
		<td>
	<input type="button" value="칼럼차트로 표현" onclick="changeLayout()"/>
	<input type="button" value="데이터 변경" onclick="changeData()"/>
	<input type="button" value="Change Both" onclick="changeBoth()"/>
		</td>
	</tr>
	<tr>
	<td>
		<script>
		<!--  //여러 차트를 동적으로 표현하기 위해서 콤비네이션 차트 swf 파일 활용
	rMateChartCreate("chart","rMateIntegration",flashVars, 400, 400, "#FFFFFF");
		// -->
		</script>
	</td>
</tr>
</table>

</body>
</html>
```

<예제 102 동적으로 차트 레이아웃과 데이터를 변경시키는 예제>

8.20. 하나의 html 문서 안에 여러 개의 차트를 생성하기

Html 문서 한 페이지에 여러 개의 차트를 생성하고자 할 때는 아래 예제를 참고하세요. 아래 예제는 한 페이지에 2 개의 차트를 생성시키는 방법을 제시합니다.



<그림 60 하나의 html 문서에서 2 개의 차트 생성 결과>

이 예제는 라인차트와 칼럼차트를 생성 시키는 예제입니다. 모든 차트를 생성시킬 수 있는 rMateIntegration.swf 를 활용하십시오. 라인과 칼럼 차트는 기본적인 차트이므로 rMateChart.swf 를 사용하셔도 무관합니다.

```
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />

<script src="AC_OETags.js" language="javascript"></script>
<script src="rMateChart.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ----------------------첫번째 차트  flashVars 설정 시작----------------------
// 1번차트는 rMate 차트와 스크립트 간의 동기화가 완료된 후 호출되는 함수를 지시하여
// 이 함수를 통하여 레이아웃과 데이터를 차트에 입력합니다.
```

```
var flashVars;

var rMateOnLoadCallFunction = "rMateChartOnLoad";
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;
// ----------------------첫번째 차트  flashVars 설정 끝----------------------


// ----------------------두번째 차트  flashVars 설정 시작----------------------
// 2번차트는 rMate 차트와 스크립트 간의 동기화가 완료된 후 호출되는 함수를 지시하여
// 이 함수를 통하여 레이아웃과 데이터를 차트에 입력합니다.
var flashVars2;

var rMateOnLoadCallFunction2 = "rMateChartOnLoad2";
flashVars2 += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction2;
// ----------------------두번째 차트  flashVars 설정 끝----------------------


// rMate 차트와  스크립트 간의 동기화가 완료되면 호출하는 함수입니다.
// 사용자는 이 함수를 flashVars에 반드시 등록해야 합니다.
// 여기서 작업을 하십시오.
// 차트 콜백함수 4개 존재합니다.
// 1. setLayout - 스트링으로 작성된 레이아웃 XML을 삽입합니다.
// 2. setData - 배열로 작성된 데이터를 삽입합니다.
// 3. setLayoutURL - 레이아웃 XML 경로를 지시합니다.
// 4. setDataURL - 데이터 XML 경로를 지시합니다.
// 아래 예제는 가능한 방법을 제시한 것 입니다.
// 현재 1번 차트는 레이아웃을 스트링 형태로 받아들이고 데이터는 배열 형태로 받아들입니다.
function rMateChartOnLoad()
{
        //예제1. 레이아웃 스트링 형태,데이터 배열 형태
        chart1.setLayout(layoutStr);
        chart1.setData(chartData);

        //예제2. 레이아웃 XML URL경로, 데이터 배열 형태
        //chart1.setLayoutURL(layoutURL2);
        //chart1.setData(chartData2);

        //예제3. 레이아웃 스트링 형태, 데이터 XML URL경로
        //chart1.setLayout(layoutStr2);
        //chart1.setDataURL(dataURL2);

        //예제4. 레이아웃 URL, 데이터 URL
        //chart1.setLayoutURL(layoutURL);
        //chart1.setDataURL(dataURL);
```

```
}

// 2번 차트에서 동기화가 완료되면 호출하는 함수(사용자에 의해 정의됨)
function rMateChartOnLoad2()
{

        //예제2. 레이아웃 XML URL경로, 데이터 배열 형태
        chart2.setLayoutURL(layoutURL2);
        chart2.setData(chartData2);

}
// 동적으로 할당할 레이아웃 정의.
var layoutURL2 = encodeURIComponent("chartLayout.xml");
var layoutStr = "<rMateChart cornerRadius='12' borderStyle='solid'>"
        +"<Options><Caption text='Annual Report'/></Options>"
        +"<Line2DChart showDataTips='true'>"
        +"<horizontalAxis><CategoryAxis categoryField='Month'/></horizontalAxis>"
        +"<series><Line2DSeries yField='Profit' displayName='Profit'>"
        +"<showDataEffect><SeriesInterpolate/></showDataEffect>"
        +"<lineStroke><Stroke color='0xFF0000'weight='4'/>"
        +"</lineStroke></Line2DSeries>"
        +"</series></Line2DChart></rMateChart>";


// 동적으로 할당할 데이터 정의.
var chartData = [{"Month":"Jan", "Revenue":10000, "Cost":5000, "Profit":5000},
        {"Month":"Feb", "Revenue":15000, "Cost":7000, "Profit":8000},
        {"Month":"Mar", "Revenue":12000 , "Cost":6000, "Profit":6000},
        {"Month":"Apr", "Revenue":30200, "Cost":4000, "Profit":26200},
        {"Month":"May", "Revenue":28000, "Cost":10000, "Profit":18000},
        {"Month":"Jun", "Revenue":12000, "Cost":5000, "Profit":7000},
        {"Month":"Jul", "Revenue":22000, "Cost":10000, "Profit":12000},
        {"Month":"Aug", "Revenue":13000, "Cost":6000, "Profit":7000},
        {"Month":"Sep", "Revenue":22000, "Cost":10000, "Profit":12000},
        {"Month":"Oct", "Revenue":29000, "Cost":8000, "Profit":21000},
        {"Month":"Nov", "Revenue":18000, "Cost":7500, "Profit":10500},
        {"Month":"Dec", "Revenue":30000, "Cost":12000, "Profit":18000} ];

var chartData2 = [{"Month":"Jan", "Revenue":1000, "Cost":500, "Profit":500},
        {"Month":"Feb", "Revenue":1500, "Cost":700, "Profit":800},
        {"Month":"Mar", "Revenue":1200 , "Cost":600, "Profit":600},
        {"Month":"Apr", "Revenue":3020, "Cost":400, "Profit":2620},
        {"Month":"May", "Revenue":2800, "Cost":1000, "Profit":1800},
        {"Month":"Jun", "Revenue":1200, "Cost":500, "Profit":700},
```

```
            {"Month":"Jul", "Revenue":2200, "Cost":1000, "Profit":1200},
            {"Month":"Aug", "Revenue":1300, "Cost":600, "Profit":700},
            {"Month":"Sep", "Revenue":2200, "Cost":1000, "Profit":1200},
            {"Month":"Oct", "Revenue":2900, "Cost":800, "Profit":2100},
            {"Month":"Nov", "Revenue":1800, "Cost":750, "Profit":1050},
            {"Month":"Dec", "Revenue":3000, "Cost":1200, "Profit":1800} ];



</script>
<!-- 사용자 정의 설정 끝  -->
</head>

<body onload="rMateChartInit()">
<table align="center" border="0" cellpadding="0" cellspacing="0">
            <tr>
            <td align="center">
                        <script>
                        <!--
                                    rMateChartCreate("chart1","rMateIntegration",flashVars, 400, 400, "#FFFFFF");
                        // -->
                        </script>
            </td>
            <td align="center">
                        <script>
                        <!--
                                    rMateChartCreate("chart2","rMateIntegration",flashVars2, 400, 400, "#FFFFFF");
                        // -->
                        </script>
            </td>

</tr>
</table>
</body>
</html>
```

<예제 103 하나의 html 문서에 2 개의 차트 생성하는 예제>

8.21. 실시간 차트 예제 – 주식 모니터링 차트

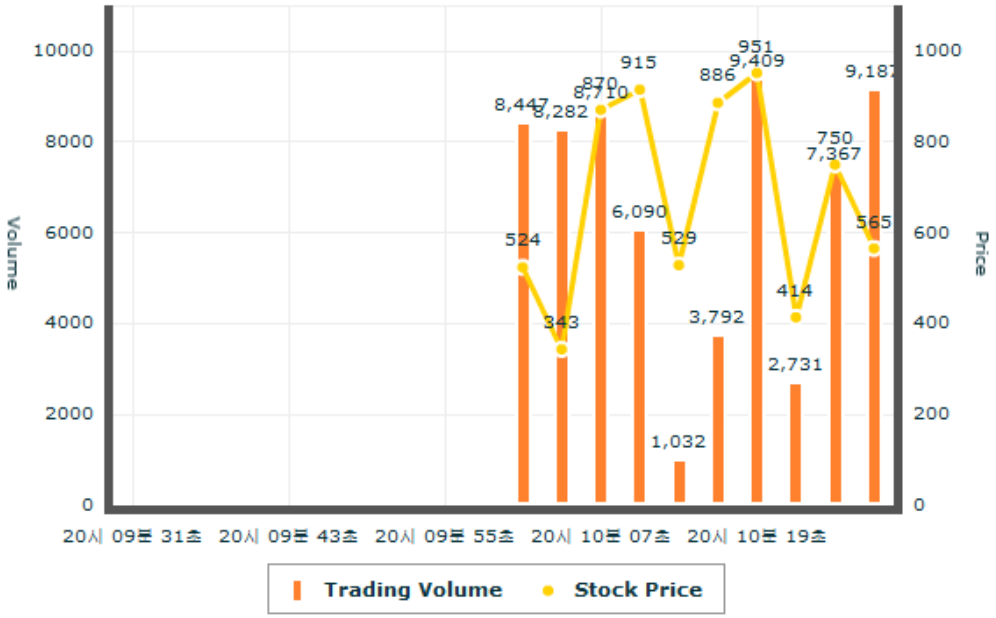
실시간 차트의 가장 일반적인 예제인 주식 모니터링 예제입니다. 이 레이아웃은 기본으로 제공되는 레이

아웃입니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="Stock Monitoring" />
                <SubCaption text="매 3초 단위로 업데이트합니다.(데이터는 랜덤)" fontSize="11"
textAlign="right"/>
                <Legend/>
        </Options>
        <DateFormatter id="dateFmt" formatString="HH시 NN분 SS초" />//포매터 정의

       //실시간 차트는 type은 time, 60초동안 3초단위의 데이터 표현
        <RealTimeChart id="chart" dataDisplayType="time"   timePeriod="60" interval="3" showDataTips="true">

              //뒷 배경에 격자 모양을 그림.
                <backgroundElements>
                        <GridLines direction="both"/>
                </backgroundElements>

            //수평축으로 DateTime축 사용, data와 축라벨은 모두 초(second)단위이고
            //데이터는 3초단위로 들어오며, 축라벨 시간 간격은 9초. GMT시간이 아닌 로컬시간표시
                <horizontalAxis>
                        <DateTimeAxis id="hAxis" dataUnits="seconds" labelUnits="seconds" dataInterval="3"
interval="9" formatter="{dateFmt}" displayLocalTime="true"/>
                </horizontalAxis>
                <series>
                        <Column2DSeries yField="Volume" xField="Time"
                                displayName="Trading Volume"
                        itemRenderer="net.riamore.rmate.charts.renderers.GradientColumnItemRenderer">
                                <filters> //칼럼 시리즈에 그림자 필터 적용
                                        <DropShadowFilter distance="3" color="0x666666" alpha=".8"/>
                                </filters>
                                <fill> //칼럼 시리즈의 채우기 색 지정
                                        <SolidColor color="0xB0C759"/>
                                </fill>
                                <verticalAxis> 칼럼 시리즈가 참고하는 축 정의
                                   <LinearAxis id="vAxis1"   title="Volume" minimum="0"
maximum="10000"/>
                                </verticalAxis>
                        </Column2DSeries>

                        <Line2DSeries xField="Time" yField="Price" displayName="Stock Price"
                        itemRenderer="mx.charts.renderers.CircleItemRenderer"   radius="5" fill="0xFFFF00">
                                <filters>
```

```
                                        <DropShadowFilter distance="3" color="0x666666" alpha="0.8"/>
                                    </filters>
                                    <stroke> //라인시리즈에서 데이터가 있는 곳의 표시를 나타내는 도형의 선 모양 정의
                                        <Stroke color="0xE48701" weight="2"/>
                                    </stroke>
                                    <lineStroke> //라인시리즈의 선 모양 정의
                                        <Stroke color="0xE48701" weight="4"/>
                                    </lineStroke>
                                    <verticalAxis> //라인시리즈가 참고하는 축 정의
                                        <LinearAxis id="vAxis2" title="Price" minimum="0" maximum="1000"/>
                                    </verticalAxis>
                                </Line2DSeries>
                            </series>
                            <verticalAxisRenderers> //두개의 축의 위치와 참고 축 정의.
                                <Axis2DRenderer placement="left" axis="{vAxis1}">
                                </Axis2DRenderer>
                                <Axis2DRenderer placement="right" axis="{vAxis2}">
                                </Axis2DRenderer>
                            </verticalAxisRenderers>
                        </RealTimeChart>
                        // 실제적으로 데이터를 풀링하는 RPC 정의.
                        <HttpServiceRepeater url="http://demo.riamore.net/chartTest/data4.jsp" target="{chart}" interval="3" method="get"/>
                    </rMateChart>
```

<예제 104 실시간 차트 주식 모니터링 레이아웃>

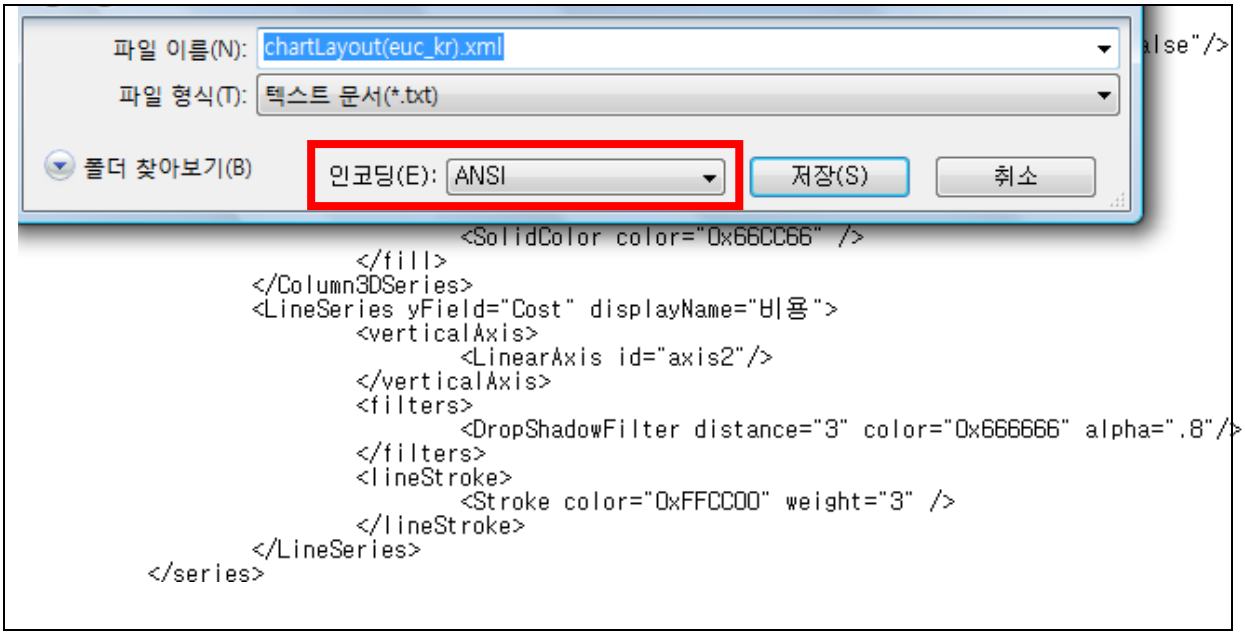


<그림 61 실시간차트 - 주식 모니터링>

8.22. 차트 레이아웃 및 데이터 euc-kr 포맷 형식의 한글 지원

기본적으로 한글이 포함된 레이아웃이나 데이터 XML 은 utf-8 포맷형식일 때 정상적인 한글표현이 가능합니다. 그러나 특수한 상황에서 utf-8 포맷형식을 사용하지 못할 경우 euc-kr 형식으로 rMate 차트는 한글을 지원합니다.

1. 레이아웃이나 데이터 XML 파일을 저장 할 때 인코딩 방식을 euc-kr(ANSI) 형태로 저장하십시오.



<예제 105 메모장의 다른 이름으로 저장하기>

2. flashVars 설정 부분을 다음과 같이 설정하십시오.

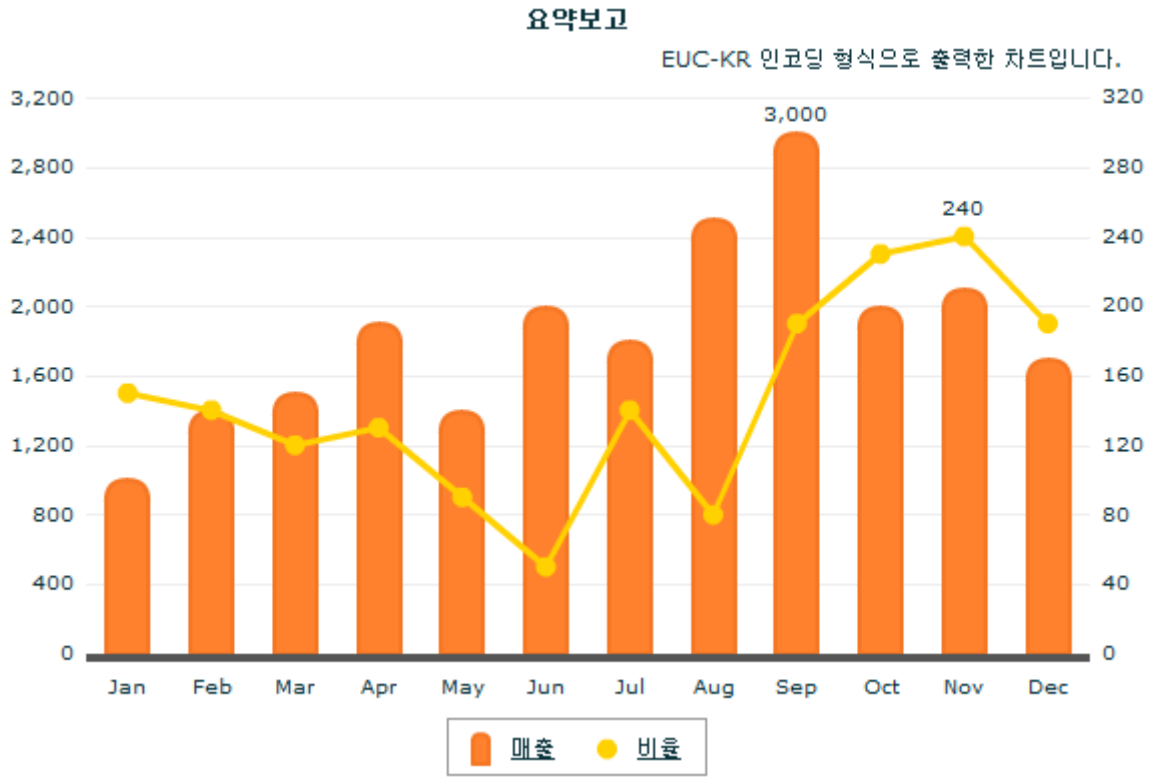
```
var flashVars = "layoutURL="+layoutURL;
flashVars += "&rMateOnLoadCallFunction="+rMateOnLoadCallFunction;

// 시스템 코드 사용..
flashVars += "&flash.system.System.useCodePage=true";
```

3. xml 헤더 부분의 encoding 을 utf-8 에서 euc-kr 로 변경 하십시오

```
<?xml version="1.0" encoding="euc-kr"?>
<rMateChart backgroundColor="0xFFFFEE"  cornerRadius="12" borderStyle="solid">
        <Options>
                <Caption text="요약보고"/>
                <SubCaption text="EUC-KR 인코딩 형식으로 출력한 차트입니다." fontSize="11"
```

```
textAlign="right" paddingRight="10"/>
                    <Legend fontSize="11" fontWeight="normal" textDecoration="underline"/>
            </Options>
                .
                .
                .
</rMateChart>
```

위 3 단계를 거친 후 실행을 하면 euc-kr 포맷 형식의 한글을 볼 수 있습니다.



<그림 62 euc-kr 포맷 형식의 한글 출력 모습>

8.23. 장애인을 위한 기능

8.23.1. 시각 장애인을 위한 대체 텍스트

시각장애인으로, 차트를 보지 못하는 분을 위하여 대체 텍스트를 제공합니다.
차트에 데이터를 삽입하였을 경우 차트 내에서 삽입된 데이터를 알아내어 대체 텍스트를 저장합니다.
대체 텍스트를 읽어 줄 수 있는 프로그램이나 기기들을 사용하여 대체 텍스트를 읽어주게 됩니다.
프로그램이나 기기와 관련하여서는 ㈜리아모어소프트에서 취급하거나 제공하는 부분은 없습니다.
이 기능은 사용자가 특별히 조작하거나 수정해야할 부분은 없습니다.

rMateChart 에서는 Flash 에서 지원하는 Accessibility 를 이용하여 대체 텍스트를 제공하고 있습니다.

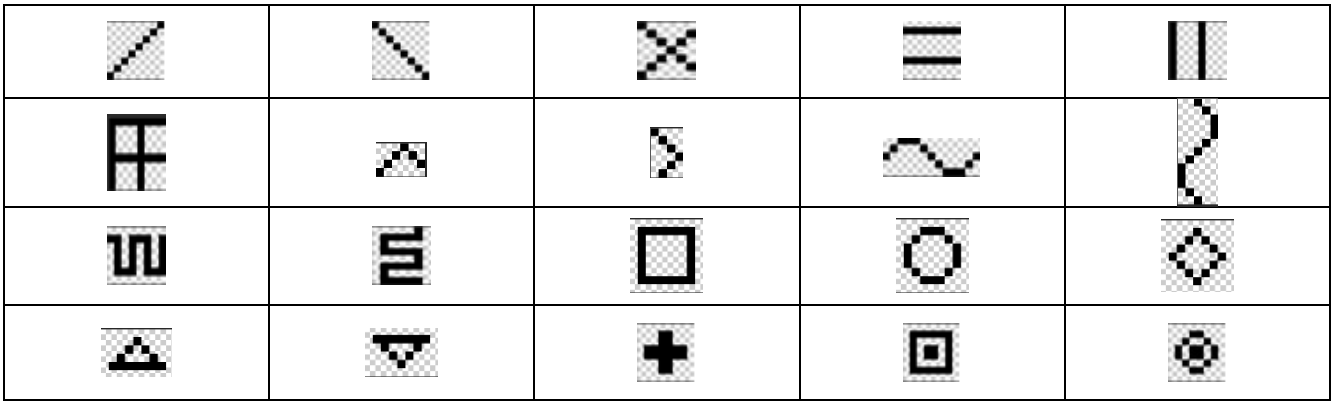
```
updateProperties.name = "텍스트"; // name수정
updateProperties.description = "텍스트2"; // description수정
this.accessibilityProperties = updateProperties; // 수정한 updateProperties적용
Accessibility.update.updateProperties(); // 수정한 대체 텍스트로 업데이트
```

Accessibility 의 name 과 description 프로퍼티를 이용하여 대체텍스트를 제공하여 차트가 로딩되었을 때 자동으로 대체 텍스트를 읽어주게 됩니다.
웹 접근성 연구소에서도 Accessibility 에 관해 설명하고 있습니다.
http://www.wah.or.kr/RIA/Flex/fx_guide01.asp

8.23.2. 색맹이나 색약을 위한 패턴지원

색맹이나 색약인 분은 차트의 색상을 구분을 하기가 어렵습니다.
이 때문에 차트 출력시 색상이 아닌 특정 모양을 출력하여 차트를 구분할 수 있게 도와주는 기능입니다.
지원하고 있는 패턴 모양은 20 가지 입니다.

아래 그림이 rMateChart 에서 지원하는 패턴이며 그 순서 입니다.



<그림 61 패턴의 모양과 순서>

아래와 같이 flashVars 에 패턴을 사용하도록 설정합니다.

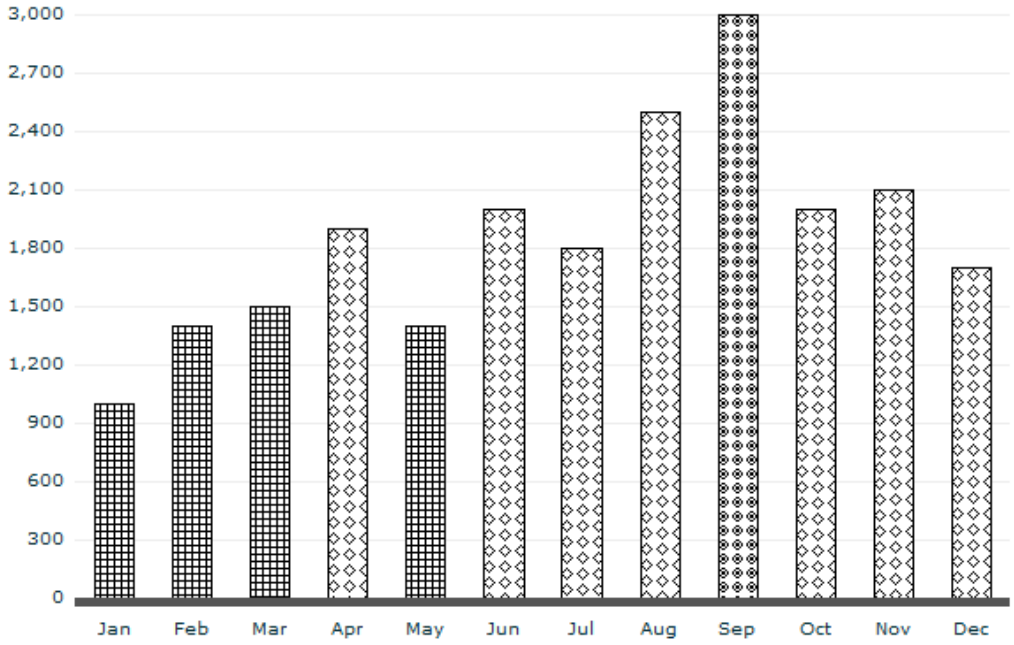
```
// 웹 접근성 패턴을 사용하시려면 usePattern을 true로 설정하여야 합니다.
flashVars += "&usePattern=true";
```

위 사항을 적용하시면 rMateChart 는 usePattern 을 확인하고 pattern 을 가지고 있는 pattern.swf 를 불러 rMateChart 에 저장 해놓습니다. Pattern.swf 는 rMateChart 컴포넌트가 불려지는 경로와 동일해야 합니다.

이후에 사용자 조작에의해 패턴을 출력하시려면 아래와 같이 적용하십시오.

```
document.getElementById("chart1").accessibilityPattern(true);
```

차트 객체에 접근하여 accessibilityPattern 이라는 함수를 실행합니다.
해당 함수로 적용되는 것은 Boolean 값이며 true 는 패턴을 적용, false 는 해제 입니다



<그림 62 패턴을 적용한 Column2DChart>

9. 5.0 업그레이드

9.1. 차트의 기본디자인 개편

차트내의 색상은 거의 대부분 이용자가 지정하여 사용하실 수 있습니다.
그러나, 대부분의 고객의 경우 차트에서 제공되는 기본색상을 그대로 사용하고 계십니다.
또한 축의 형태,폰트 등도 기본을 그대로 사용하십니다.
이번 5.0 버전에서는 차트의 색상 및 축, 랜더러 등의 기본 구성요소들의 디자인을 최근의 세태에 맞는 트랜디한 스타일로 새롭게 개편하였습니다.
저희차트의 기본을 그대로 사용하시는 고객에게는 좀 더 멋진 차트로 보여질 것입니다.

9.2. 기타 추가된 기능

가. 레이더 차트에서 범례와 연동된 수치표시 기능의 추가 :
레이더 차트에서는 차트의 내부에 데이터 값을 표시하지 않습니다.
이는 내부공간이 협소하여 읽기의 어려움때문입니다.
이를 보완하기 위하여 범례와 연동하여, 해당 포커스가 간 범례에 한하여, 데이터의 값을 표시할 수 있도록 하는 기능이 추가 되었습니다.

9.3. 업그레이드 주의사항

이번 5.0 업그레이드에서는 디렉토리의 구조가 변경되었습니다.

하단의 내용를 참고하여 주시기 바랍니다.

- 차트 swf 파일

  Component/rMateChart.swf -> rMateChart /Component/rMateChart.swf

- 자바스크립트 파일

  Samples/JS/AC_OETags.js -> rMateChart /JS/AC_OETags.js

  Samples/JS/rMateChart.js -> rMateChart /JS/rMateChart.js

- Samples/ JS/ rMateChartLicense.js -> LicenseKey / rMateChartLicense.js

<감사합니다.>