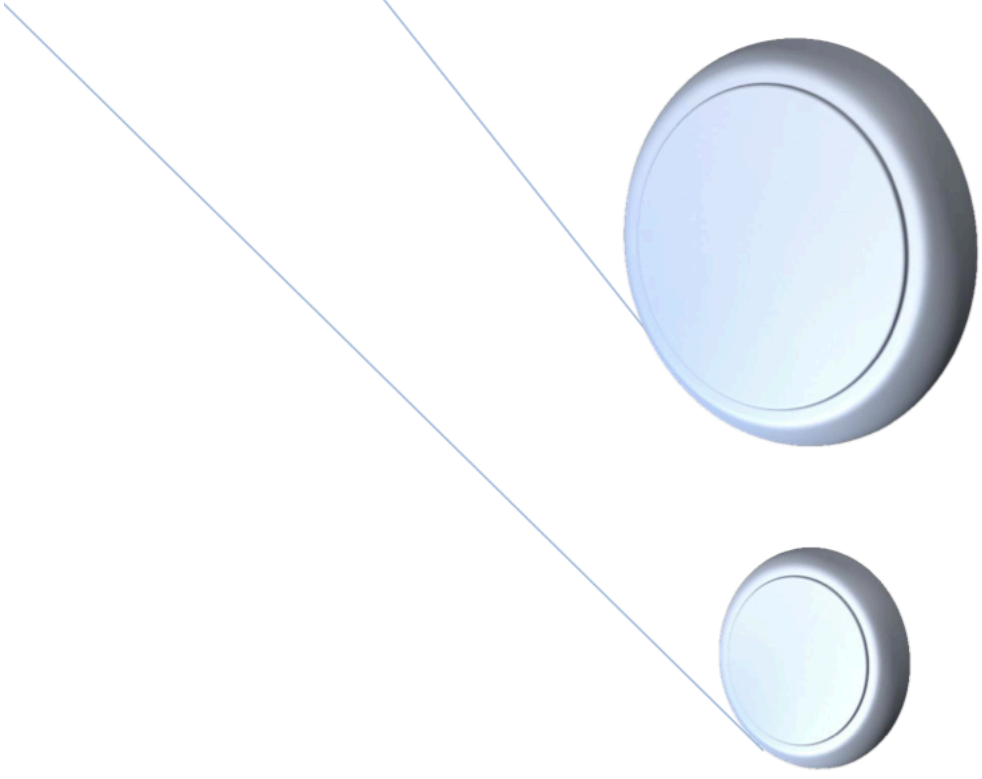


[오프라인강좌]

ExtJS를 활용한 Ria App개발하기

강사 : 곽옥석
benneykwag@gmail.com

목차

1. 선행기술의 이해

2. ExtJS OverView

2.1. Ext JS란?
2.2. 주요 내장 콤포넌트
2.3. JSON을 통한 개발방식
2.4. Event 중심의 개발방식
2.5. MVC의 구현

3. ExtJS 시작하기

3.1. 준비물
3.2. 디렉토리 구조(MVC)
3.3. ext.js와 ext-all.js의 차이점
3.4. 디버그 모드 사용하기
3.5. Hello World

4. ExtJS Class System

4.1. 클래스의 선언과 생성
4.2. Data Package OverView
4.3. Ext.data.Model
4.4. Ext.data.Proxy
4.5. Ext.data.Store

1. 선행기술의 이해

자바스크립트란?

ExtJs는 수많은 컴포넌트를 내장하고 있으며 어떠한 ActiveX도 설치 되지 않으면서 일관되고 재활용가능한 개발이 가능합니다.

그중에서도 그리드 기능은 가장 강력하여 기업용 Application을 개발 구축하는데 장점으로..
ExtJS는 오픈소스이며 누구나 자유롭게 다운로드하고 개발에 적용할 수 있습니다.
또한 이 프레임웍을 익힐 경우 Sencha-Touch라는 모바일 프레임웍의 이해도 금방할 수 있다…

Ajax

Ajax(Asynchronous JavaScript + Xml)의 약자로 자바스크립트를 통해 비동기 통신을 서버와 클라이언트간에 제공한다. Ajax는 신기술이 아니지만 이를 포함하여 ExtJS와 같은 신 기술들이 쏟앚고,...

Ajax는 기존의 리로딩 방식의 UI개발에서 페이지를 이동하지 않고 변경되는 데이터만 전환할 수 있다.
여기서 동기와 비동기 방식을 이해할 필요가 있다.

동기방식 : 이는 여러개의 스레드가 순차적으로 실행되고 순차적으로 종료되는 것을 말한다.
모든 것이 순서에 이해 진행된다.

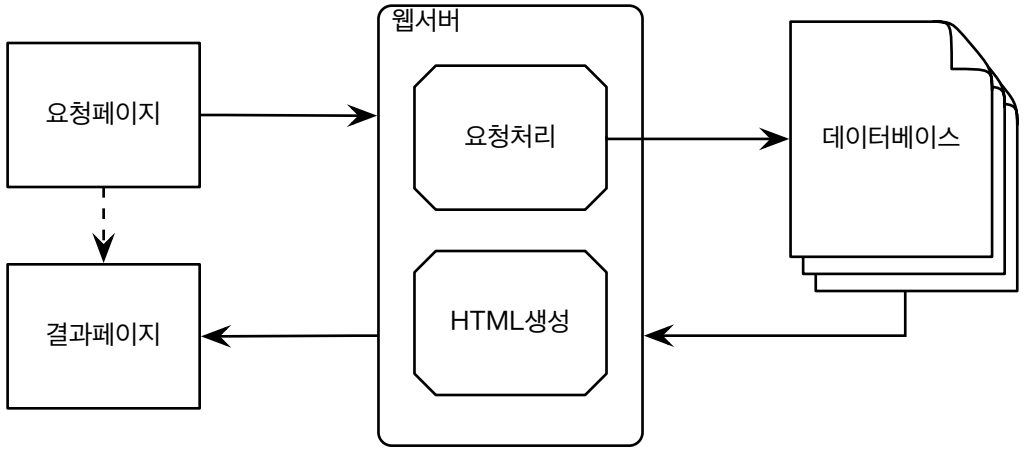
비동기방식 : 이 방식은 모든 스레드가 독립적으로 실행되고 종료되고 각각의 스레드는 연관성이 없어
웹페이지에 경우처럼 복잡하고 다양한 컨텐츠를 표현시 사용자에게 지루함을 주지 않고 ..

Ajax의 장단점

장점	단점
페이지 리로드없이 화면전환	크로스브라우저에 대응 필요
먼저 끝날수 있는 요청은 기다리지 않고 처리	요청을 남발하면 서버의 부하를
서버측 처리를 각 PC에 분산 가능	HTTP클라이언트의 기능이 하넞ㅇ
수신 데이터랑이 현저히 줄어듬.	-
단일, 복수 파일의 Lock 처리	-

기존 개발 방법

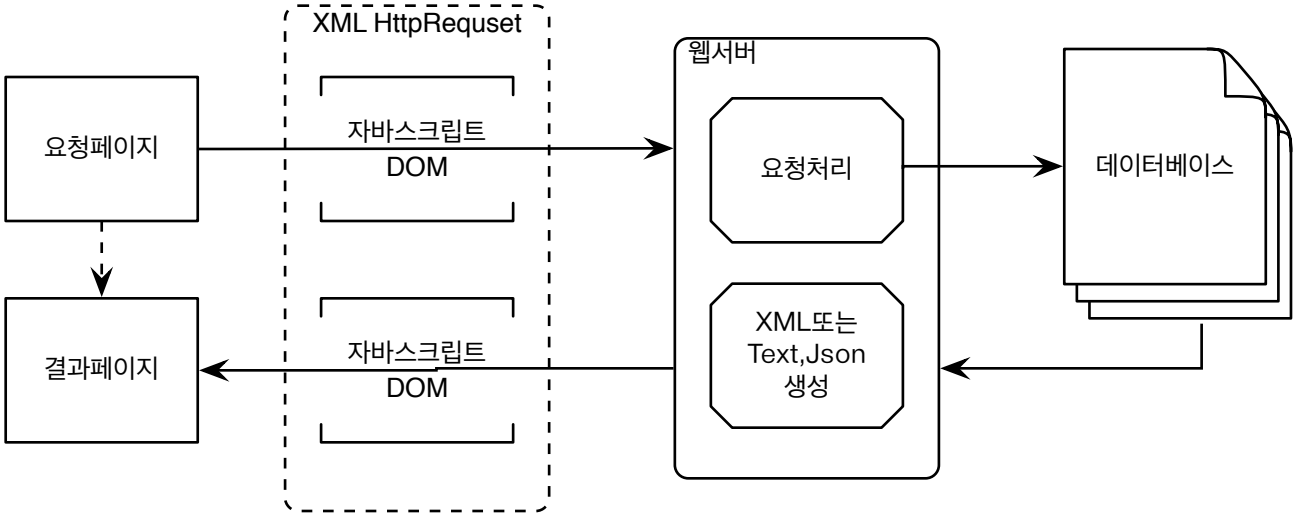
기존 방식은 웹브라우저가 웹서버에 요청을 전송하고 웹서버는 관련정보를 분석하여 결과를 리턴 브라우저에 화면을 그려주고 이때 사용되는 것이 HTML이다. 결과적으로 서버는 요청을 받아 또 다른 페이지로 이동하도록 하고잇다…..



Ajax 개발 방법

사용자가 이벤트를 발생하고 자바스크립트는 DOM을 사용하여 필요한 정보를 구한 뒤 XMLHttpRequest객체를 통해서 웹서버에 요청을 전달
웹서버는 요청에 대해 알맞게 처리하고 결과를 xml이나 json형태로 결과를 제공하고 자바스크립트는 응답데이트를 DOM을 이용 사용자 화면을 생성하거나 변경된 데이터만 수정한다.

이때 사용자는 페이지의 이동 즉 리로드가 발생하지 않고 변경되어야 하는 부분만 교체할 수 있게 된다.



Ajax의 주요구성 요소

이름	설명
XMLHttprequest	사용자 요청을 웹서버에 전달 또는 결과를 브라우저에 전달
DOM	문서의 구조를 나타냄, 폼 등의 정보나 화면 구성을 조작할 때 사용 함.
CSS	글자색, 배경색, 위치, 투명도 등 UI요소 담당
JavaScript	모든 DOM객체의 탐색 및 제어, 이벤트발생, 요청,응답처리

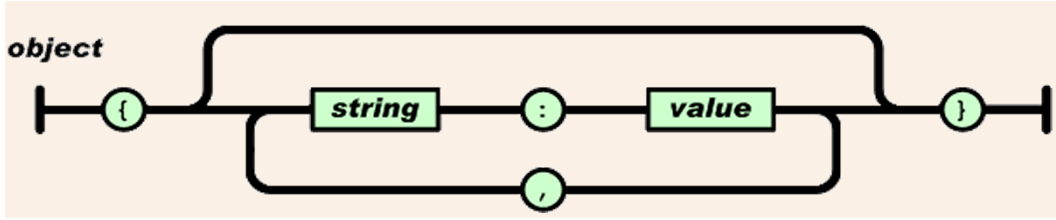
제공처 : http://blog.naver.com/jhoh86?Redirect=Log&logNo=10133727118

JSON(JavaScript Object Notation)

우리가 학습할 UI프레임웍인 ExtJs는 JSON(JavaScript Object Notation)을 사용하여 Data를 교환합니다.
JSON은 경량의 DATA교환을 위한 형식으로 불리우고. 이 형식은 사람이 읽고 쓰기에 용이하며, 기계가 분석하고 생성하기에도 용이합니다.
JSON은 완벽하게 언어로 부터 독립적이며 모든 언어에서 활용기 가능한 이상적인 DATA교환 방식입니다.

Object

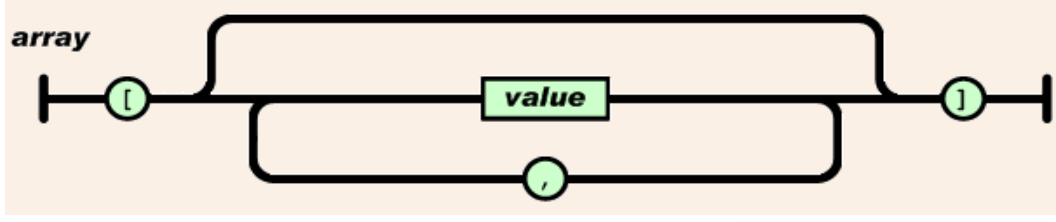
object는 name/value쌍들의 비순서화된 SET이다, object는 "{" 중괄호로 시작하고 "}"로 끝내어 표현한다.각 name뒤에는 ":" colon을 붙이고 "," comma로 name/value 쌍들 간을 구분한다.




```
{
	"title":"MS Office 2007처음 사용하는.",
	"input_user_name":"홍길동",
	"input_date":"2012.01.29",
	"dept_name":"생산2팀"
}
```

Array

array는 값들의 순서화된 collection이다. array는 "[" left bracket으로 시작해서 "]" right bracket으로 끝내어 표현된다. "," comma로 array의 값들을 구분한다.

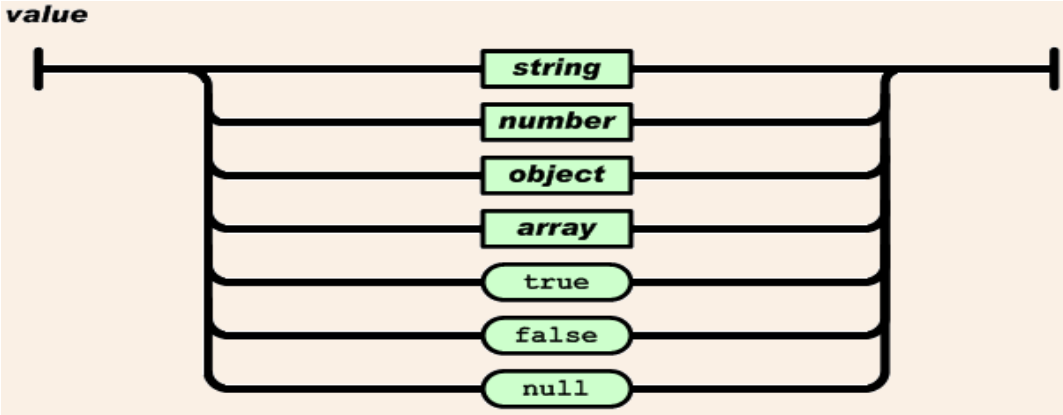



```
[
	{
		"title":"MS Office 2007처음 사용하는.",
		"input_user_name":"홍길동",
		"input_date":"2012.01.29",
		"dept_name":"생산2팀"
	},
	{
		"title":"생산공정을 단순화한 과제수행",
		"input_user_name":"이순",
		"input_date":"2012.02.29",
		"dept_name":"생산4팀"
	}
]
```

JSON(JavaScript Object Notation)

Value

value는 큰따옴표안에 string, number, true, false, null, Object, array가 올수 있다.




```
{
	"title":"String 타입",
	"count": 100,
	"use_yn": true,
	"dept_name":null
}
```

2. ExtJS Overview

2.1 ExtJS란?

ExtJs는 수많은 컴포넌트를 내장하고 있으며 어떠한 ActiveX도 설치 되지 않으면서 일관되고 재활용가능한 오픈소스 UI프레임웍이다.
무료이며 웹상에서 CS UI를 지원하며 그리드를 중심으로 수많은 콤포넌트를 내장하고 있다. 우리는 이제 디자이너에 의존했던 UI구성의 단점을 개발자가 직접 만들어내고 고객과 함께 Proto Type을 구성하므로 해서 납기단축과 디자인 공수를 최소화 할 수 있다.
ExtJS를 통해 구현할 수 있는 예시는 "http://docs.sencha.com/ext-js/4-1/#!/example"에서 확인 할 수 있다.

2.2. 주요 내장 콤포넌트

내장 클래스명	설명
Ext.grid.Panel	필터링, 소팅, 컬럼숨김 기능 제공
Ext.container.Container	대표적으로 콤포넌트를 배치 할 수 있는 클래스
Ext.layout.Layout	콤포넌트의 배치형태를 구현
Ext.Ajax	ExtJS의 통신을 담당
Ext.window	레이어팝업 지원(html, 콤포넌트배치지원)
Ext.data.Model	Store와 쌍으로 데이터형태 표현
Ext.data.Store	Proxy, Model과 함께 데이터를 포함 함.

3.3. JSON을 통한 개발방식

앞장에서 설명한 JSON방식은 ExtJS의 모든 것을 표현한다. 즉 모든 코드는 JSON방식으로 코딩되며 서버에서 전달받은 데이터 또한 JSON방식으로 해석하고 표현한다.

```
// app/view/Viewport.js
Ext.define('ria.view.Viewport', {
    extend: 'Ext.container.Viewport',
    layout : 'border',
    items: [
    {
        region: 'center',
        // 기존 panel에서 여러개의 패널을
        // 붙여야 하므로 tabpanel로 변경합니다.
        xtype: 'tabpanel'
    },{
        region: 'west',
        xtype : 'WestTabPanel',
        width: 200
    }]
});
```

2.4. Event 중심의 개발방식

이벤트(event)란 어떤 동작이 발생했을 때를 말한다. 이벤트는 클릭과 같은 사용자 행위가 될수 있고 또는 Ajax호출에 따른 응답이 될 수도 있다. 사용자가 버튼을 누르면 하나 이상의 이벤트가 발생한다.
커서가 버튼위에 있거나, 클릭하거나, 버튼의 범위를 벗어날 때도 역시 이벤트가 발생한다. 이러한 이벤트들이 발생할 때 이들을 처리하는 함수인 이벤트 처리함수(event listener)를 붙일 수 있다.

```
{
        xtype: 'button',
        text: '검색',
        action : 'common.search',
        // listener는 버튼을 감시한다.
        listeners : {
                // 버튼을 감시 중 클릭 이벤트가 일어날 경우
                click : function(a){
                        // console에 '안녕하세요'를 표현한다.
                        console.log('안녕하세요');
                }
        }
}
```

2.5. MVC의 구현

ExtJS는 Model, View, Controll 즉 MVC모델을 구현할 수 있다. 이는 각기 쓰임이 다른 레이어를 구분하고 역할을 나누므로서 로직과 뷰등이 혼재되어 스파케티화 되는 것을 막을 수 있다.
또 이는 유지보수 입장에서도 큰 잇점으로 작용하며 전체적인 가독성을 높이는 역할을 한다.

MVC	설명
Ext.data.Model	모든 객체의 Config(특성정보)를 담고 있다.
Ext.view.View	모든 보이는 형태를 표현할 최상위 클래스
Ext.app.Controller	모델로 객체의 형태를 담고 뷰로 표현하면 제어는 Controller가 한다.

```
Ext.define('ria.controller.RiaAppController', {
    extend: 'Ext.app.Controller',
    // 관장할 뷰리스트
    views: ['ria.view.pgms.RiaAppMain01',
            'ria.view.pgms.RiaGanttPanel',
            'ria.view.pgms.SchedulerPanel'],
    // Model들의 집합은 Store에 모이구 컨트롤러는 이 Store를 참조한다.
    stores: [
             'ria.store.Persons',
             'ria.store.Projects'
    ],
```

```
    init: function() {
      // 모든 뷰들을 리스닝할 수 있다.
      this.control({
            // step1. 좌측 매뉴에 추가된 패널 즉 시스템 패널에 대한 이벤트 추가.
            'RiaAppMain01': {
                  afterrender: this.firstSelect
            },
            'RiaAppMain01 button[action=common.]': {
                  click: this.btnClicked
            }
        });
    },
    firstSelect: function(){
      //console.log('aaaa');
    },
    btnClicked: function(btn){
      console.log('btn', btn);
    }
});
```

위의 코드는 Controller 클래스의 예시이다. MVC모델의 중심에는 Controller가 있고 이 Controller가 뷰와 모델을 참조하고 제어하는 역할 한다.

역할	설명
참조	관련 뷰들을 참조하고 참조를 통해 접근가능.
이벤트	참조된 뷰들에서 발생하는 모든 이벤트 리스닝
제어	Controller는 로직을 구현하기 보다 뷰와 뷰간의 제어를 담당하고 이를 위해 자체 함수를 구현할 수 있다.

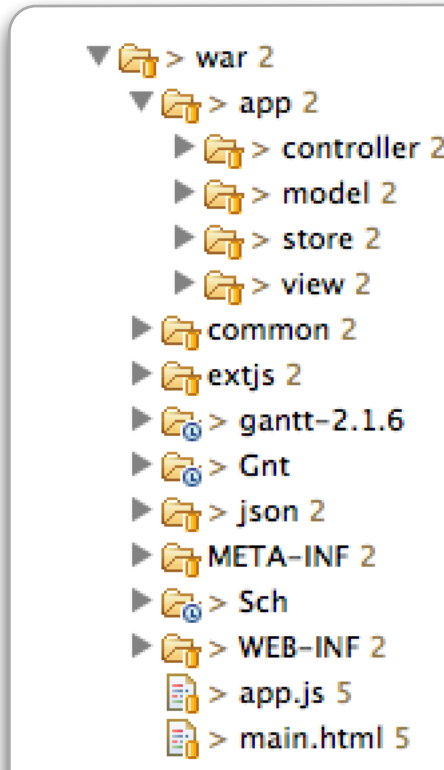
3. ExtJS시작하기

3.1 준비물

준비물	설명
개발툴	Eclipse , Text Editor
Application Server	Tomcat 7
Sencha Api	http://sencha.com에서 다운로드

3.2. 디렉토리 구조(MVC)

최초 MVC디렉토리 구조를 잡고 시작한다. 각 디렉토리는 각자의 의미를 가지고 있고 최종 배포시에 각 디렉토리의 클래스를 통합하여 압축 및 최적화를 하게 된다.



파일 및 디렉토리	설명
app/controller	뷰간의 이벤트 및 제어 역할
app/model	Store와 함께사용 Bean객체역할
app/store	model의 집합(array)
app/view	사용자정의 뷰클래스
extjs	extjs core라이브러리
app.js	어플리케이션의 시작클래스
main.html	app.js와 모든 리소스명시
app/view/ViewPort.js	기본 뷰포트(오직하나만 사용)

```
<html> <!--// main.html // -->
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title id="page-title">ExtJS시작하기</title>
    <!-- // extjs core가 사용하는 css // -->
    <link rel="stylesheet" type="text/css" href="/extjs/resources/css/ext-all-gray.css"/>
    <!-- // 개발에 필요한 ext라이브러리 // -->
    <script type="text/javascript" src="/extjs/ext-debug.js"></script>
    <!-- // 우리가 만든 어플리케이션 // -->
    <script type="text/javascript" src="app.js"></script>
</head>
<body>
</body>
</html>
```

MVC모델 방식의 main.html

3.3. ext.js와 ext-all.js의 차이점

ExtJs 프로젝트를 시작할 때 해야할 첫번째 일은 HTML페이지에 ExtJS 파일을 포함 시키는 것이다. ext-all파일을 선택하면 브라우저는 ExtJS프레임웍 전체를 로딩한다. 반면 ext.js버전을 선택하면 브라우저는 어플리케이션의 실행을 위한 최소코드를 로딩하며 추가기능을 로딩하기 위해 동적 로딩 기능을 사용한다.

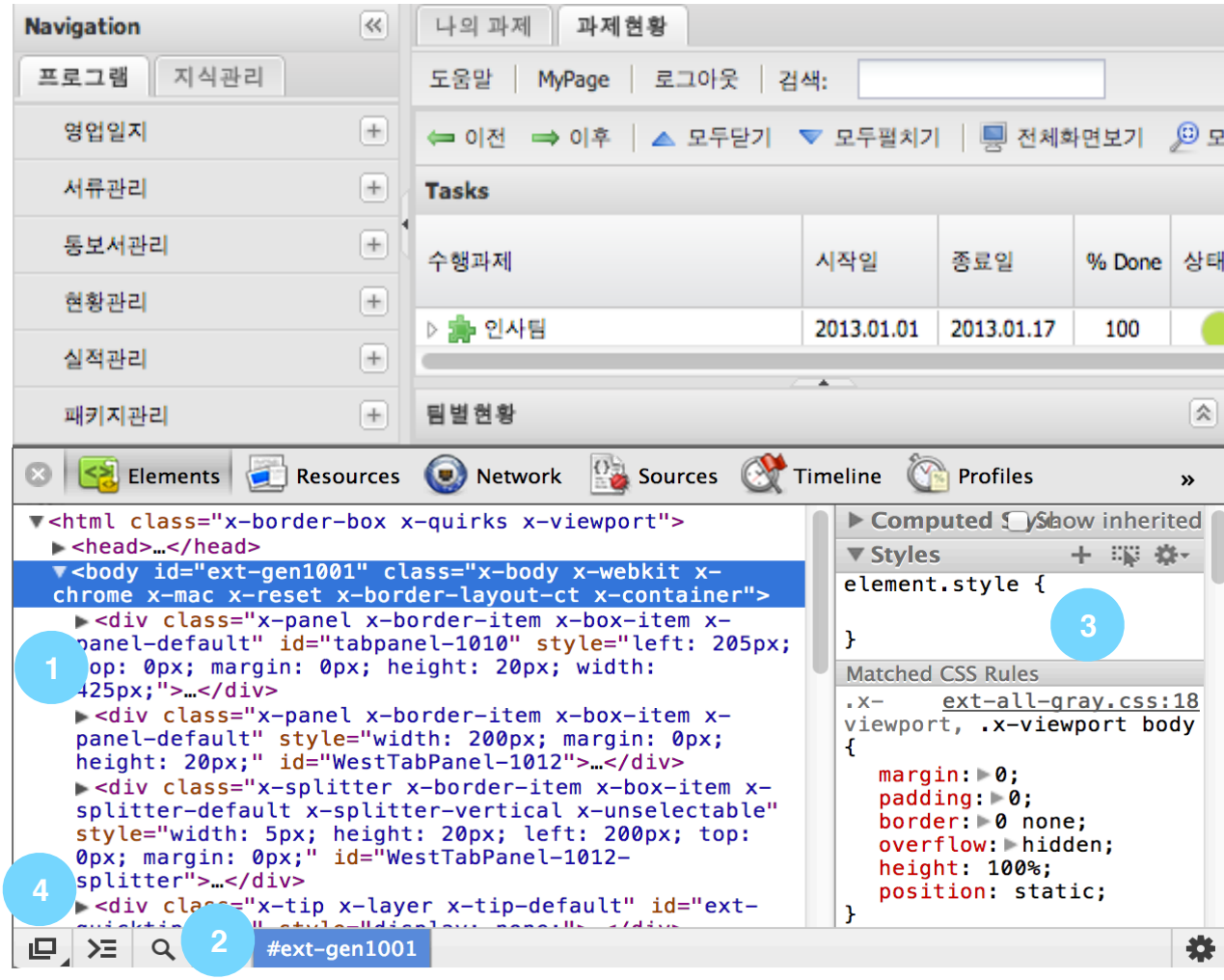
boostrap.js에 대하여:ext-all.js 대신해서 bootstarp.js파일을 HTML페이지에 포함 시킬 수 있다. 이 파일은 사용하고 있는 환경에 따라서 ext-all.js 또는 ext-all-debug.js를 페이지에 포함 시킨다. ext-all-debug.js를 포함 시킬 경우는 다음과 같다.
1. hostname(호스트네임)이 localhost 일 때
2. hostname이 IP 주소일 때
3. 현 프로토콜이 파일일 경우
4. 다른 경우에 boostrap.js는 ext-all.js 파일을 로딩한다.

Name	종류	로더	축소됨	주석	디버그	사용법
ext-debug.js	Core					로컬 개발 시 사용
ext.js	Core					Custom빌드와 함께 사용
ext-all.js	All					Custom빌드가 없을 경우
ext-all-debug.js	All					Custom빌드가 없고 디버그 사용

3.3. 디버그 모드 사용하기

ExtJS는 브라우저에 탑재된 디버그 프로그램을 통해 에러에 대한 핸드링, 로그확인 등을 지원한다. 우리는 크롬에 포함된 디버그모드를 통해 디버그를 할 수 있다.
크롬 메뉴 : 보기 -> 개발자정보 -> 자바스크립트 콘솔을 클릭하면 아래 그림과 같이 디버그 모드를 확인 할 수 있다.

디버그모드는 중요한 개발 도구로 이를 잘 활용하여야만 문제점을 쉽고 빠르게 파악하고 Dom에 대한 구조를 알 수 있다. 또한 개발자가 작성하는 모든 로그를 보여주고 구현프로그램의 흐름등을 파악할 수 있다.

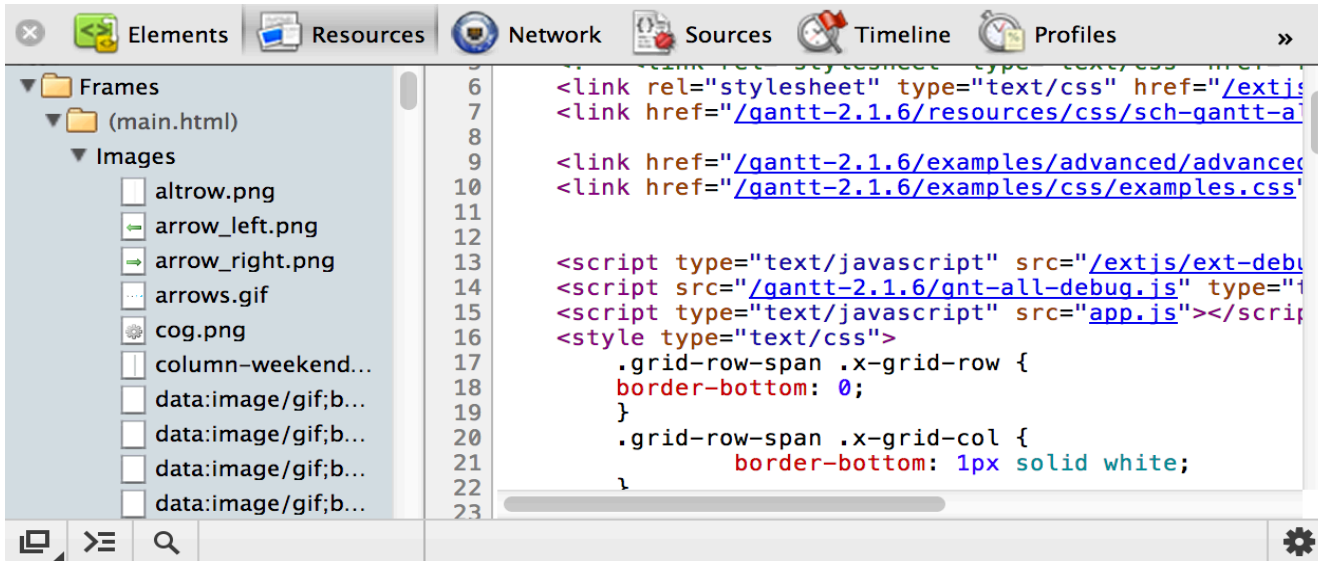


각 탭별로 기능에 대해 알아본다. 첫번째 탭인 Elements탭은 어플리케이션의 DOM Tree와 Style Sheet정보, DOM 객체의 검색 기능 등이 지원 된다.

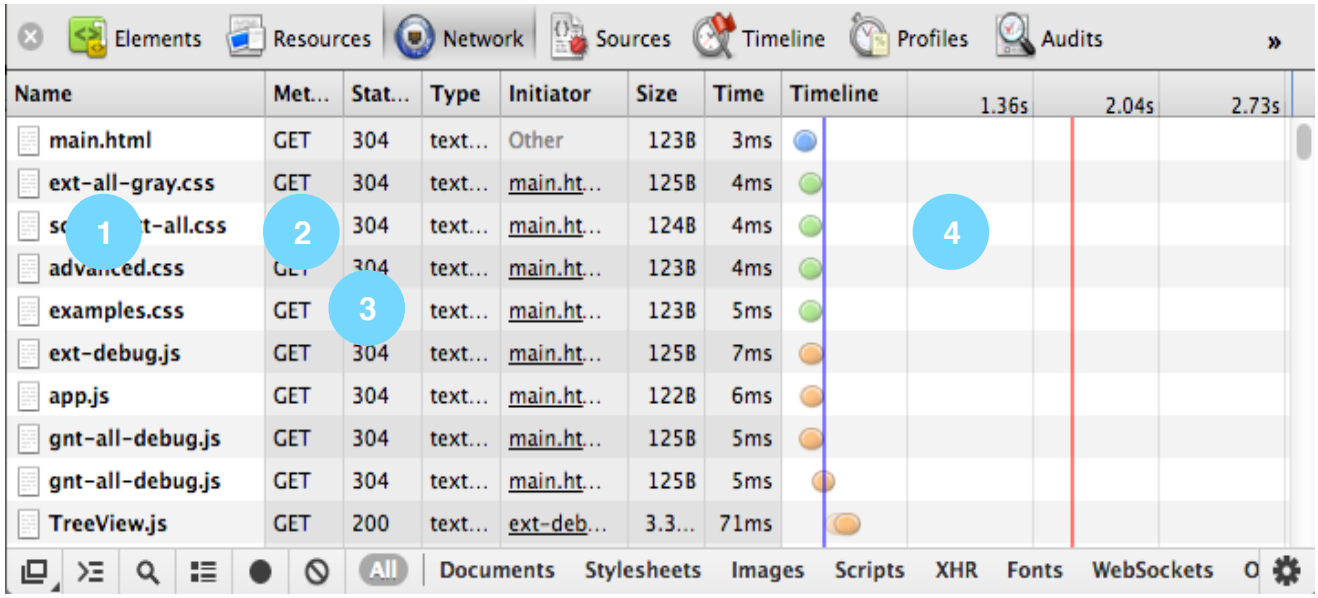
기능	설명
1. DOM트리	모든 어플리케이션은 HTML코드와 DOM객체, Style Sheet로 이루어져 있고 이 기능을 통해 구조파악을 쉽게 할 수 있다.
2. 탐색기능	돋보기 버튼을 클릭하고 화면상에서 확인하고자 하는 객체를 클릭하면 1번 화면에 블럭으로 표시된다.
3. Style Sheet	블럭으로 표시된 DOM객체가 사용하는 CSS를 표시
4. 독립창 실행	디버그 모드창을 독립적으로 실행 함.

3.3. 디버그 모드 사용하기

두번째 탭은 Resources탭으로 어플리케이션에서 사용되는 모든 소스와 이미지, 링크정보 등을 표시하고 어플리케이션의 리소스 연관성 등을 파악할 수 있다.



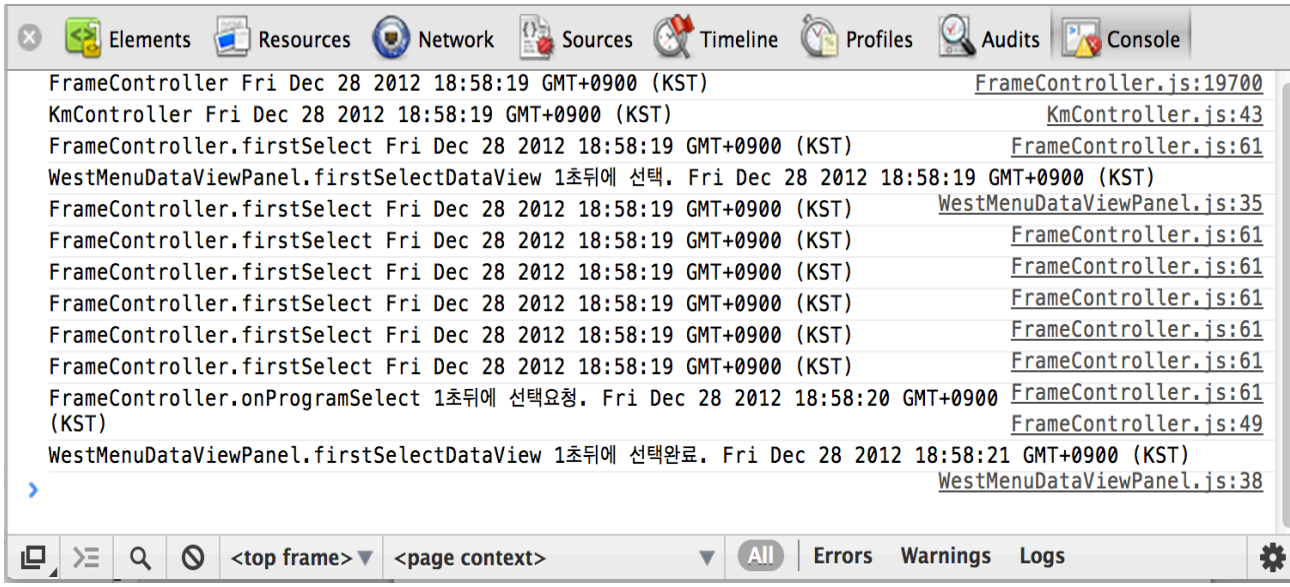
세번째 탭은 Network로 어플리케이션 실행시 로딩되는 모든 Content의 다운로드 정보와 서버와 주고 받는 Ajax 통신 정보, 통신정보 내에 포함된 Response정보, Request정보를 파악할 수 있다.



기능	설명
1. 다운로드 된 Content	하나의 어플이케이션이 실행 시 다운로드하는 이미지, 자바스크립트, CSS 등 확인
2. Method	Request방식을 표시한다. GET메소드와 POST방식의 구분을 표시한다.
3. Status	요청에 대한 결과 코드이다 쉬운예로 404에 경우 존재하지 않는 컨텐츠를 요청한 경우다. HTTP메시지 코드 참조
4. Timeline	요청된 각각의 Content의 다운로드 타임을 표시한다.

3.3. 디버그 모드 사용하기

다음으로 Console 탭을 설명합니다. Console탭에서는 어플이케이션에서 발생하는 각종 로그정보를 표기합니다. 이는 개발시 가장 많이 쓰이는 기능으로 코드 상에서는 Console.log("출력할 내용")로 실행되고 코드실행시 문제가 있을 경

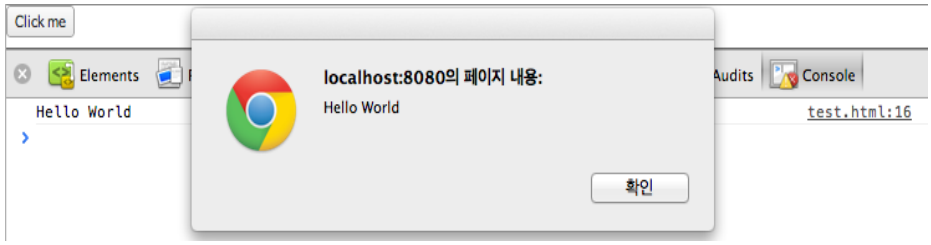


3.4. HelloWorld

MVC구조를 사용하지 않더라도 간단한 코딩으로 ExtJS를 테스트하고 로직을 구현하여 어플리케이션을 구현할 수 있다. 우선 test.html을 만들고 아래와 같이 코딩합니다.

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title id="page-title">ExtJS시작하기</title>
    <link rel="stylesheet" type="text/css" href="/extjs/resources/css/ext-all-gray.css"/>
    <script type="text/javascript" src="/extjs/ext-debug.js"></script>
    <script type="text/javascript">
    Ext.onReady(function(){     // ExtJS가 사용될 수 있는 조건이다.
        var button = Ext.create('Ext.Button', {
            text: 'Click me',
            renderTo: Ext.getBody(),
            handler: function() {
                console.log('Hello World');
                alert('Hello World');
            }
        });
    });
    </script>
</head>
<body></body>
</html>
```

위의 코드는 ExtJS가 실행 가능한 시점(Ext.onReady)에 Ext.Button을 생성하여 버튼 객체를 만들고 만들어진 버튼을 HTML의 Body에 보여지도록 하는 코드입니다. 또한 버튼을 클릭하면 경고창으로 "Hello World"를 보이도록 하고 동시에 콘솔창에 "Hello World"를 프린트 하게 됩니다.



4. ExtJS Class System

ExtJS는 프레임워크 자체 클래스 시스템을 갖고 있다. 자바스크립트는 기본 클래스가 없으므로 ExtJS클래스 시스템은 개발자가 더 객체지향적으로 코드를 작성할 수 있도록 하고 있다. ExtJS는 개발하기 쉬우며 유연한 새로운 클래스 시스템을 제공하며 하위버전의 호환성을 지킨다.

- 클래스의 선언과 생성
- 믹스인(mixin)
- 자동 생성된 getter, setter메소드
- 동적 클래스 로딩
- static

4.1 클래스의 선언과 생성

```
// app/view/MyPanel.js
Ext.define('ria.view.MyPanel', {                    ①
    extend: 'Ext.panel.Panel',                      ②
    alias: 'MyPanel',                               ③
    layout : 'border',
    items: [                                        ④
    {
        region: 'center',
        // 기존 panel에서 여러개의 패널을
        // 붙여야 하므로 tabpanel로 변경합니다.
        xtype: 'tabpanel'                           ⑤
    },{
        region: 'west',
        xtype : 'WestTabPanel',
        width: 200
    }]
});

// case 1 : 클래스의 생성 (Ext.create)
var mypanel = Ext.create('ria.view.MyPanel');       ⑥
// case 2 : 클래스의 생성 (Ext.widget)
var mypanel = Ext.widget('MyPanel');                ⑦
// case 3 : 클래스의 생성 (xtype)
items : [{                                          ⑧
    xtype : 'MyPanel',
    title : '나의 패널'
```

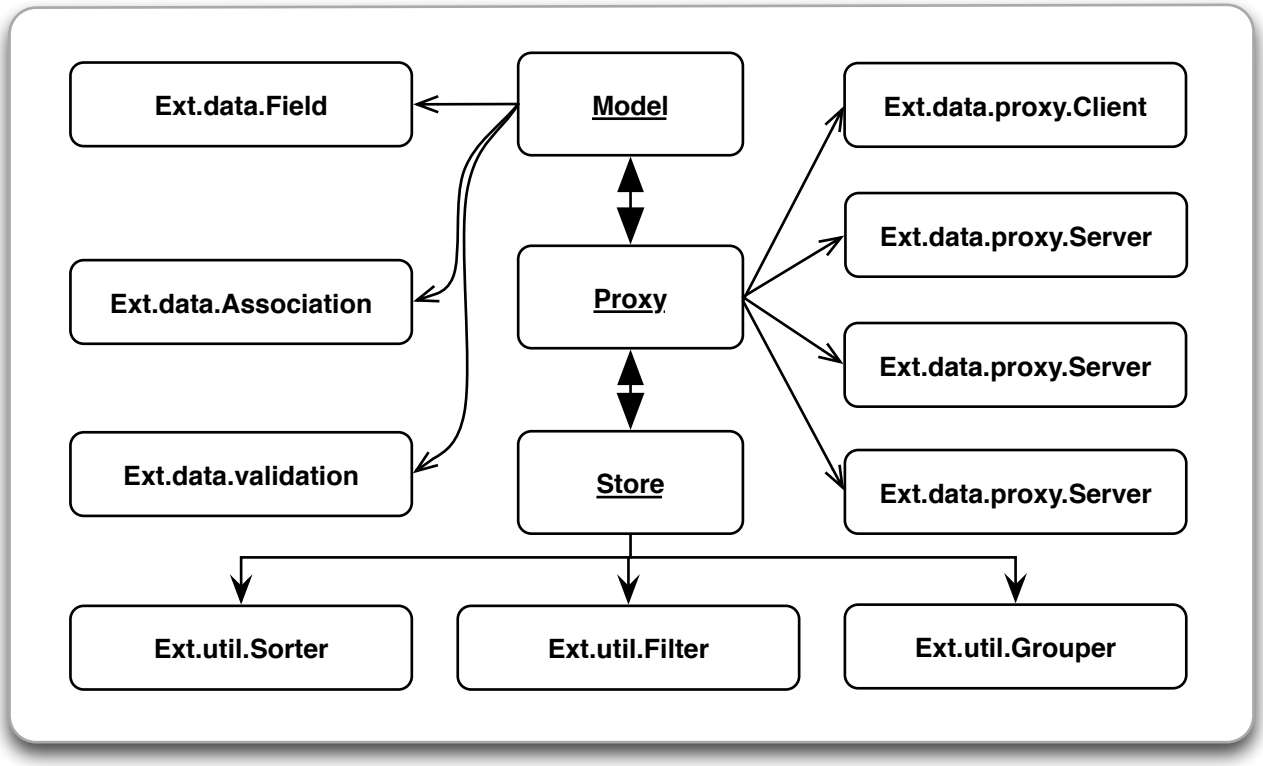
①. Ext.defined을 통해 클래스를 정의한다.
"ria.view.MyPanel"은 클래스의 풀네임이다. "."으로 어플리케이션의 이름과 폴더를 구분한다.
"MyPanel"은 클래스의 파일명으로 MyPanel.js로 생성하여 위의 내용이 채워진다.
②. extend는 상속의 의미로 'Ext.container.Container'클래스를 상속하여 만들어진 클래스이다.
객체지향에서 상속받은 클래스의 기능을 활용하고 수정하고 재정의 할 수 있다.
우리는 이 상속개념을 가장 많이 활용하게 된다.
③. alias를 통해 위젯명을 정의한다. 위젯명은 xtype으로 사용된다.
④. items는 위에서 정의된 클래스는 Ext.panel.Panel클래스를 상속 받았고 Panel과 같은
Container클래스는 자식을 가질 수 있고 자식을 효율적으로 배치할 수 있는 Layout을 지원한다.
⑤. xtype:tabpanel은 위젯명이고 풀클래스명은 Ext.tab.Panel이다.
⑥. Ext.create를 사용하여 define된 클래스를 생성한다.
⑦. Ext.widget을 사용하여 define된 클래스의 alias를 호출한다.
⑧. define된 클래스의 alias를 xtype을 통해 item으로 추가한다.

4. ExtJS Data Package

4.1. Data Package OverView

데이터패키지는 Model과 Store, Proxy를 중심으로 데이터를 포함하고 생성하고 , 표현하는 등 ExtJS에 의해 다뤄지는 모든 데이터를 관장한다.

데이터패키지의 구조는 아래의 다이어그램과 같다.



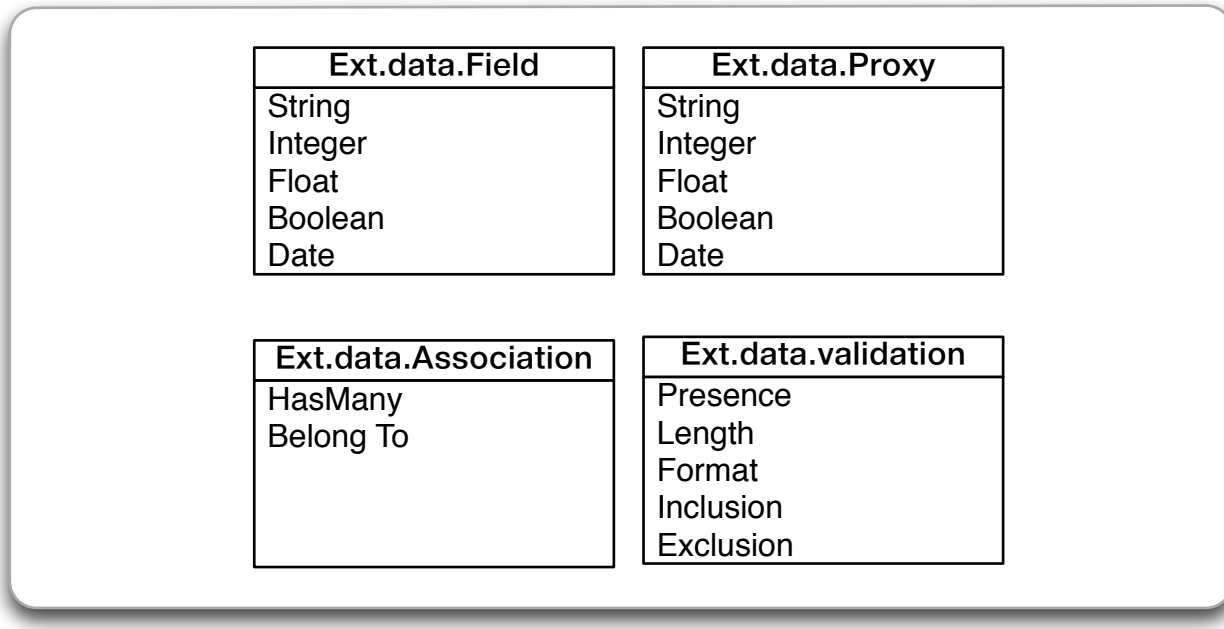
Record클래스의 새로운 버전인 Model클래스는 데이터 패키지에서 가장 중요한 클래스 중 하나이다. Model클래스를 이용해 실제 Entity를 표현할 수 잇고 Entity간의 관계 Association을 설정할 수 있고 유효성 validataion을 추가 정의할 수 있다.
이전 버전의 Record 클래스와 4버전의 Model클래스간의 비교 목록은 아래와 같다.

기능	Record Class (ExtJs3)	Model Class(ExtJs4)
필드	지원	지원
함수	지원	지원
유효성(validation)	미지원	미지원
관계(Association)	미지원	미지원
직접데이터 핸들링하기	미지원	미지원

Proxy는 Model이나 Store에 직접 접근할 수 있다. 또한 데이터를 디코딩하는 리더와 라이터 인스턴스를 얻을 수 있다. Proxy는 Client와 Server Proxy로 나뉘어 제공된다. Store는 정렬, 필터, 그룹핑과 같은 기능이 ExtJs4에서 추가되었다.

4.2. Ext.data.Model

Model은 객체와 엔티티를 표현한다. Model은 데이터를 처리하는 함수와 필드의 집합체이다. 이전 버전의 Record와 유사하지만 더 강ㅇ력하고 데이터를 조작하는 함수나 필드를 생성할 수 있으며 모델관의 관계가 성립되도록 설정할 수 있다.
Ext.data.Model클래스의 다이어그램은 아래와 같다.



Model은 Ext.define를 통해 선언되며 Ext.data.Model클래스를 확장한 새로운 클래스를 생성하고 필드를 정의한다. 아래는 Model의 작성예이다.

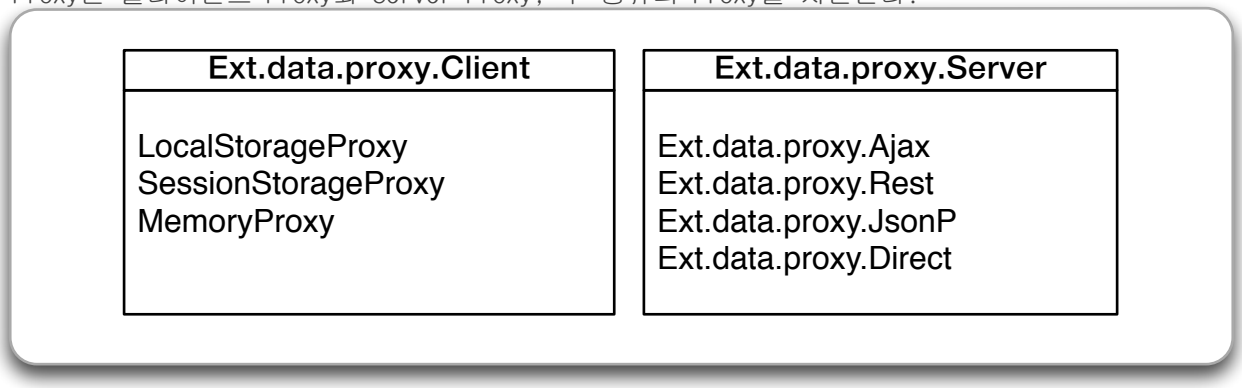
```
Ext.define('ria.model.SmplModel',{
	extend: 'Ext.data.Model',
	fields:
		[
		 {name: 'name'},
		 {name: 'juso', type: 'string'},
		 {name: 'tel', type: 'int'},
		 {name: 'email', type: 'string'},
		 {name: 'birthday', type: 'date', dataFormat: 'd/m/Y'},
		 {name: 'weight', type: 'float'},
		 {
			name: 'weightKg',
			type: 'float',
			convert: function(value, record){
				var weight = record.get('wight');
				return Math.round(wight * 0.45359237);
			}
		 }]
});
```

위 코드에서 모델 SmplModel을 정의했습니다. 7의 필드를 갖고 있는 Entity를 표현했다. 각 필드는 Ext.data.Field클래스의 instance이고 이 instance는 name과 type속성을 갖고 있다. 필드의 name은 필드의 이름이고 type는 데이터 타입을 의미한다.

4.2. Ext.data.Proxy

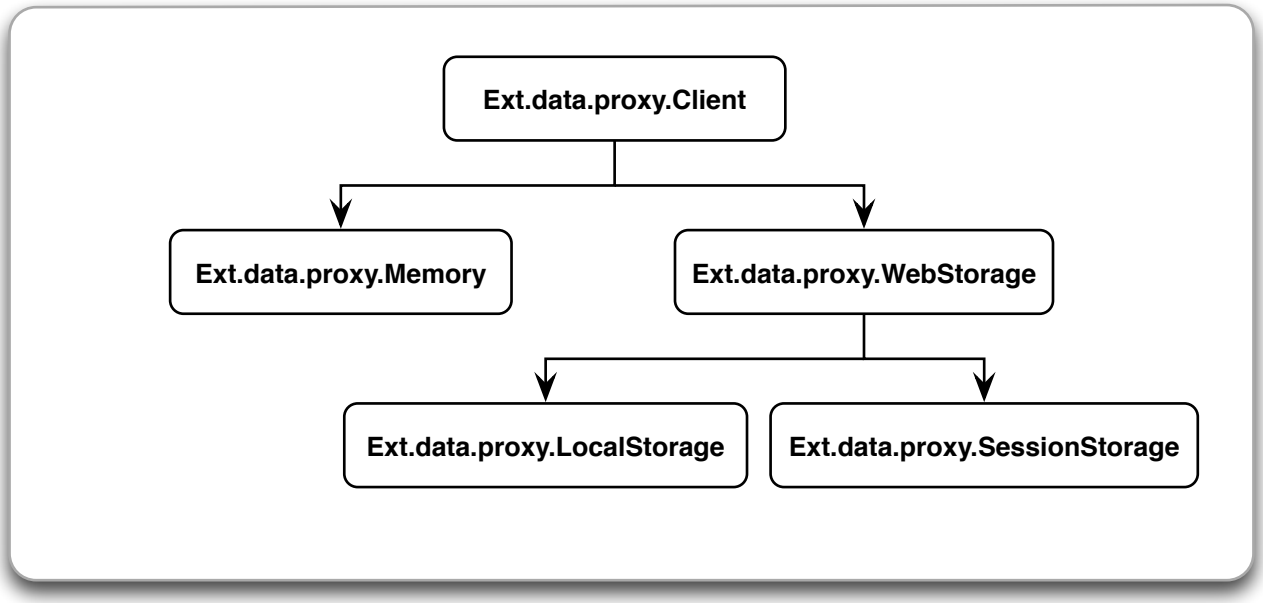
Proxy는 ExtJs에서 데이터 로딩과 저장을 담당한다. Store처럼 Proxy도 모델에서 직접 사용할 수 있다.
3버전에서는 원격서버의 데이터 로딩과 저장만 가능했지만 4버전에서는 세가자의 새로운 Proxy를 제공하여 브라우저의 로컬데이터 로딩과 저장이 가능하다.

Proxy는 클라이언트 Proxy와 Server Proxy, 두 종류의 Proxy를 지원한다.



Client Proxy

Client Proxy는 브라우저에서 지원하는 로컬저장소에 사용된다. 이 저장소는 HTML5의 새로운 기능 중 하나로 대부분의 브라우저가 지원하고 있다. 마치 쿠키와 유사하지만 HTML5저장소는 쿠키와 다르게 로컬에 저장된 데이터를 수동으로 실행하지 않는 이상 웹서버로 전송되지 않는다.

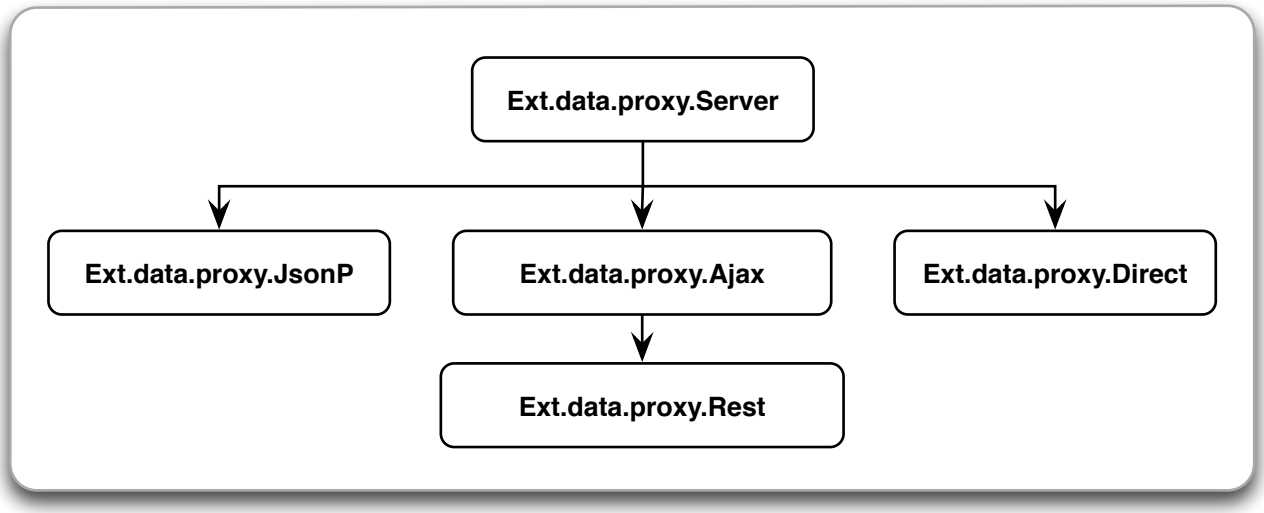


패키지	설명
Ext.data.proxy.Client	기본 클래스
Ext.data.proxy.WebStorage	Ext.data.proxy.LocalStorage, SessionStorage의 슈퍼클래스로 직접사용 안함.

4.2. Ext.data.Proxy

Server Proxy

Server Proxy는 웹서버로부터 HTTP요청으로 데이터를 로딩하거나 저장할 때 사용되고 우리와 같은 기업환경에서 절대적으로 사용되는 Proxy이므로 주목하기 바란다.



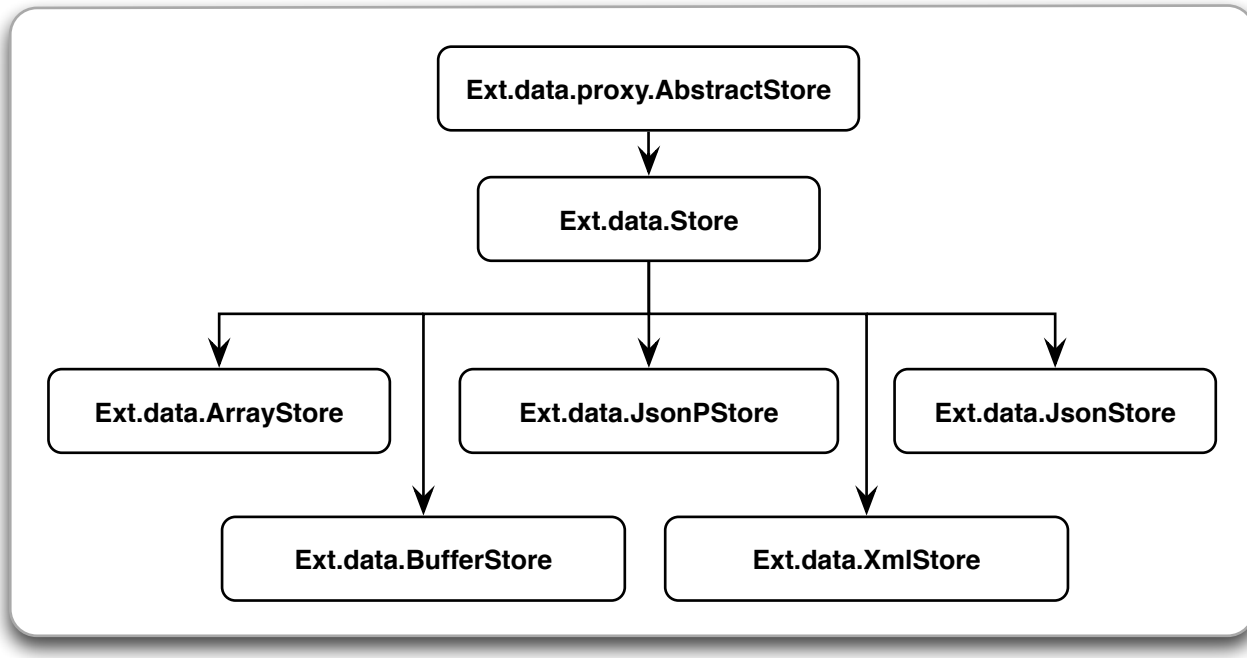
Ajax Proxy

Ajax Proxy는 Server Proxy의 한종류이다. Ajax Proxy는 가장많이 사용되는 Proxy이고. 이전 ExtJs3의 Ext.data.HttpProxy처럼 서버로부터 데이터를 로딩하고 저장할 때 Ajax요청을 사용한다.
Type은 ajax로 설정하면 된다.

```
/**
 * Date : 2012.12.14
 * Desc : 프로젝트 리스트
 */
Ext.define('ria.store.project.Projects', {
    extend: 'Ext.data.Store',       // 당연히 store상속
       autoLoad : false,                  // 자동 로드는 꺼놓자.
    model: 'ria.model.ProjectMaster',      // 모델은 동일하게 사용 세팅
    groupField: 'team_nm',
       proxy: {
        type: 'ajax',
        url: '/json/project/projectlist.json',
        reader: {
               type: 'json',
               root: 'entitys',
               totalProperty: 'totalCount',
               messageProperty: 'message'
        },
        listeners: {
            exception: function(proxy, response, operation){
              // 나중에 구현할 부분 모든 ajax 통신에 공통으로 쓸 수 있는
              // 에러 캐치 함수를 만들 것이다.
                     }
           }
    }
});
```

4.3. Ext.data.Store

Store는 모델을 감싸고 데이터를 로딩하고 저장하는 역할을 하며 정렬과 필터링, 그룹핑과 같은 추가기능이 있다. Store Class의 구조는 아래와 같다.



아래는 Store의 사용의 예이다. Store는 서버와 통신할 수 있는 수단들이 탑재되어 있다. Store는 모델과 Proxy를 포함하여 Store를 호출하므로 해서 데이터통신 후 데이터 집합을 저장할 수 있다. Store는 그리드나 트리 등에 포함되어 사용된다. 즉 그리드와 같이 데이터 집합을 표현하는 콤포넌트에 사용되는 것이다.
store는 load함수를 통해 데이터를 가져올 수 있게된다. 이는 autoLoad가 false일 경우 최초 로딩하지 않기 때문에 특정 이벤트를 받아 store를 load()함수를 통해 데이터를 가져오는 것이다.

```
// 그리드에서 사용 시
xtype: 'grid',
columnLines: true,
name : 'master',
title: '심사마스터',
frame: true,
// width: 500,
//height : 400,
store: this.store, // 생성된 store를 그리드에 세팅


// store를 로드할 때는 아래와 같이
me.store = Ext.create('ria.store.system.Programs');      // 재활용 되는 것을 막는다.
me.store.load({
       params: {
              // 아시스템 아이디를 프로그램 store에 전달하는
              // 시스템 패널별로 해당 시스템 이하의 프로그램을 가져올수 있다.
              pgm_syscd: this.pgm_syscd
       }
});
```