

ExtJS + ExtGWT

Horváth András

NetVisor Zrt.

ExtJS

Bevezető

ExtJS:

- Kiforrott Javascript GUI Framework
- Rengeteg kész GUI komponens, egy-két trükkös Javascript kiegészítés igényesebb kód írásához
- Hivatalos homepage, példákkal, dokumentációval:
<http://www.sencha.com/products/extjs/>

Előnyei

- Hasonló előnyök mint más Javascript framework-ök esetén
- Ingyenes, browser független
- Többféle licenszelési lehetőség
- Aktívan fejlesztik
- Teljes kontrollt kapunk a GUI elkészítéséhez (vs pl.: Echo)
- Tud „Lightweight” is lenni (elég csak azokat a komponenseket betölteni, amik kellenek)

Hátrányok

- A keretrendszer nagyon könnyen bővíthető és sokoldalú funkcionalitást bocsájt rendelkezésünkre,
- Még mindig Javascript (de lehetőségünk van ExtGWT-t használni)

Getting started

1. Töltsük le az ExtJS SDK-t:
2. Készítsük el az alábbi könyvtárszerkezetet:

`/application`

`/app`

`/namespace`

`MyClass1.js`

`...`

`/extjs`

`/resources`

`/css`

`...`

`/app.js`

`/index.html`

- **application**: az alkalmazás neve
- **app**: az általunk készített osztályokat tartalmazó könyvtár
- **extjs**: ide kell kitömöríteni az extjsdk.zip tartalmát
- **resources**: a statikus erőforrásokot (például képeket) tartalmazó könyvtár
- **app.js**: az felhasználó felületet vezérlő logikát tartalmazó fájl
- **index.html (vagy index.jsp stb.)**: az alkalmazás belépési pontja

Egy egyszerű példa

- Készítsük el az alábbi egyszerűsített könyvtárszerkezetet

```
/application
```

```
  /extjs
```

```
    /resources (from extjs.zip)
```

```
    /ext-all-debug.js (from extjs.zip)
```

```
/app.js
```

```
/index.html
```

- Az index.html-t készítsük el, üres body résszel, a head-be pedig helyezzük el az alábbi hivatkozásokat:

```
<link rel="stylesheet" type="text/css"  
href="extjs/resources/css/ext-all.css">
```

```
<script type="text/javascript"  
src="extjs/ext-debug.js"></script>
```

```
<script type="text/javascript"  
src="app.js"></script>
```

- **extjs/resources/css/ext-all.css**: a keretrendszer összes formázással kapcsolatos beállítását tartalmazó css fájl
- **extjs/ext-debug.js**: a teljes ExtJS keretrendszer egy minimális részhalmozát tartalmazó javascript fájl
- **app.js**: az alkalmazásunk kódját tartalmazó fájl

Használhatnánk az ext-debug.js helyett az alábbiakat:

- **ext.js**: ugyanaz mint az ext-debug.js, csak minimalizált méretű

- **ext-all-debug.js**: a teljes keretrendszer használatához szükséges állomány

- **ext-all.js**: ugyanaz mint az ext-all-debug.js, csak minimalizált méretű

Az **ext-all.js** használata nem javasolt, hiszen az alkalmazások általában nem használják a teljes keretrendszert.

Helyette a Sencha SDK Tools segítségével készíthetünk el a testreszabott **ext-only-what-we-need.js** jellegű állományainkat.

- helyezzük el az alábbi kódot az `app.js`-ben:

```
Ext.application({
    name: 'HelloWorld',
    launch: function() {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
            items: [
                {
                    title: 'Hello World',
                    html : 'Hello World!'
                }
            ]
        });
    }
});
```

- **launch**: az itt megadott metódus fog lefutni az oldal betöltődése után
- nem szükséges explicit deklarálni, hogy mely komponenseket fogjuk használni, amikor egy olyan komponenst használunk, mely nincs letöltve, akkor az ExtJS automatikusan letölti.

Hátránya: “megakad” az oldal a letöltés idejére

Megoldás: a használt komponenseket érdemes explicit deklarálni az **Ext.require** utasítás segítségével. Pl.:

```
Ext.require('Ext.container.Viewport');
```

```
Ext.require('...');
```

...

```
Ext.application({  
    name: 'HelloWorld',  
    launch: function() {  
        ...  
    }  
});
```

ExtJS MVC bevezető

- Model: A model felelős az adatok tárolásáért és perzisztálásáért. Az ExtJS “store” objektumokat használ a model megvalósításához.
- View: tetszőleges komponens (pl.: grid)
- Controller: a model-ek inicializálásáért, view-k rendereléséért és egyéb alkalmazáslogikáért felelősek

- Részletesebb leírás az ExtJS MVC modelljéről:

http://docs.sencha.com/ext-js/4-0/#!/guide/application_architecture

- Controller: eseményekre figyelnek melyek bekövetkezésekor valamilyen tevékenységet folytatnak (saját osztállyal valósítjuk meg)

```
Ext.define('MyApp.controller.Test', {  
    extend: 'Ext.app.Controller',  
    init: function() {  
        this.control({  
            'viewport > panel': { //could be 'testlist'  
                //for every panel which is a child of a viewport  
                render: this.onPanelRendered  
            }  
        });  
    },  
    onPanelRendered: function() {  
        console.log('Panel rendered!');  
        alert('Panel rendered!');  
    }  
});
```

- View és Model:

```
Ext.define('MyApp.view.testnamespace.ListTest', {
    extend: 'Ext.grid.Panel',
    alias: 'widget.listtest',
    title: 'Test List',
    initComponents: function() {
        this.store = {
            fields: ['name', 'price'],
            data: [
                {name: 'Stuff1', price: '30000 HUF'},
                {name: 'Stuff2', price: '456 GBP'}
            ]
        };
        this.columns = [
            {header: 'Name', dataIndex: 'name'},
            {header: 'Price', dataIndex: 'price'}
        ];
        this.callParent(arguments);
    }
});
```

- Ki kell bővítenünk a control-t:

```
Ext.define('MyApp.controller.Test', {  
    (...)  
    views: [  
        'testnamespace.ListTest'  
    ],  
    (...)  
});
```

- Meg kell jelenítenünk a komponensünket a Viewport-ban:

```
Ext.application({  
    launch: function() {  
        Ext.create('Ext.container.Viewport', {  
            layout: 'fit',  
            items: {  
                xtype: 'testlist'  
            }  
        });  
    }  
});
```

```
( )
```

ExtJS komponensek

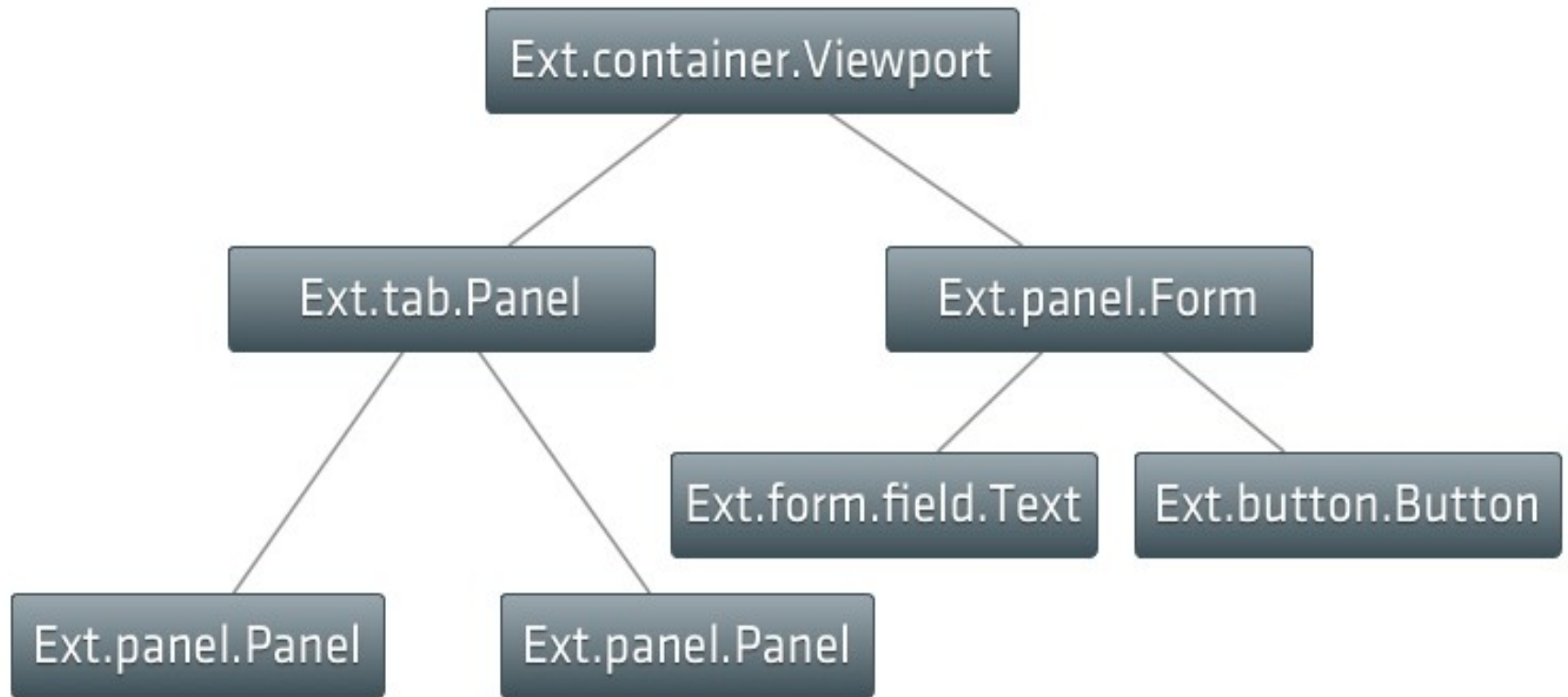
- <http://www.sencha.com/learn/components/>
- egy ExtJS alkalmazás grafikus felülete egy vagy több widget-ből épül fel, amiket Component-nek nevezünk
- minden Component az Ext.Component osztályból származik, ez az osztály biztosítja az automatizált életciklus menedzselését, beleértve:
 - Inicializálás
 - Renderelés
 - Átméretezés
 - Pozícionálás
 - Megsemmisítés
- az ExtJS rengeteg komponenst tartalmaz, továbbá lehetőségünk van saját Component-ek elkészítésére

- speciális komponens: Container
- a Container olyan Component, ami más Component-eket képes tárolni
- a Container felelős a gyermekei életciklusának kezeléséért
- egy alkalmazás általában legalább egy Container-t tartalmaz: a komponenshierarchia tetején lévő Viewport, amely az összes többi Container-t és Component-et tartalmaz:

```
var childPanel1 = Ext.create('Ext.panel.Panel', {  
    title: 'Child Panel',  
    html: 'Child Panel content'  
});
```

```
Ext.create('Ext.container.Viewport', {  
    items: [ childPanel ]  
});
```

Komponentenhierarchia



XType és Lazy Instantiation

- minden komponens rendelkezik egy szimbolikus névvel: xtype
- példa: Ext.panel.Panel komponens xtype-ja: 'panel'
- a beépített komponensek xtype-jai megtalálhatóak az API dokumentációban:

<http://docs.sencha.com/ext-js/4-0/#!/api/Ext.Component>

- eddig mindig explicit létrehoztuk a komponenseinket, ez egy nagy alkalmazásnál nem hatékony:

```
var panel = new Ext.Panel({  
    (...)  
    items: [  
        Ext.create('Ext.button.Button', {  
            text: 'OK'  
        })  
    ]  
};
```

- lehetőségünk van implicit módon létrehozni komponenseket, így spórolva a memóriával és növelve az alkalmazás sebességét (csak akkor inicializálódnak és kerülnek megjelenítésre, amikor éppen szükség van rájuk):

```
var panel = new Ext.Panel({  
    (...)  
    items: [{  
        xtype: 'button',  
        text: 'OK'  
    }]  
};
```

- lehetőségünk van saját xtype-ot definiálni:

```
Ext.define('CustomButton', {  
    extend: 'Ext.button.Button',  
    alias: 'widget.mycustombutton',  
    text: 'Custom text'  
});
```

Fontos a widget prefix az alias-nál!

- Egy bonyolultabb példa:

```
Ext.create('Ext.tab.Panel', {  
    renderTo: Ext.getBody(),  
    height: 100,  
    width: 200,  
    items: [  
        {  
            // this will be rendered  
            xtype: 'panel',  
            title: 'Tab One',  
            html: 'The first tab',  
            listeners: {  
                render: function() {  
                    Ext.MessageBox.alert('Rendered One', 'Tab One was  
rendered.');                }  
            }  
        }  
    ]  
});
```

```
{  
    // this component configuration does not have an  
    xtype since 'panel' is the default xtype for all Component  
    configurations in a Container  
  
    title: 'Tab Two',  
  
    html: 'The second tab',  
  
    listeners: {  
        render: function() {  
            Ext.MessageBox.alert('Rendered One', 'Tab Two  
was rendered.');        }  
    }  
}  
];  
});
```

- a futtatás eredménye:

- az első tab eseménykezelője lefut és megjelenik a popup ablak, ennek oka, hogy az első tab default viselkedés szerint aktív, ezért azonnal automatikusan inicializálódik és megjelenik
- a második tab eseménykezelője csak akkor fut le, ha rákattintunk, vagyis az inicializálás és a render-elés csak a tab aktivizációjakor történt meg

Pozícionálás és megjelenítés

- minden komponens rendelkezik beépített `show` és `hide` metódusokkal
- a `hide` default implementációja a css-ből ismert “`display: none`”, de ezt megváltoztathatjuk a `hideMode` segítségével:

```
var panel = Ext.create('Ext.panel.Panel', {  
    renderTo: Ext.getBody(),  
    title: 'Test',  
    html: 'Test Panel',  
    hideMode: 'visibility' // use the CSS visibility property  
    to show and hide this component  
});
```

```
panel.hide();
```

```
panel.show();
```

- Lehetőségünk van arra, hogy a komponenseink ne vegyenek részt az őket tartalmazó Container layout-jában, az ilyen komponenseket Floating Component-nek nevezzük
- Egyes komponensek Floating Component-ek, mint például a `Window`, de lehetőségünk van tetszőleges Component-et Float Component-é konvertálni, a `floating` segítségével:

```
var panel = Ext.create('Ext.panel.Panel', {  
    width: 200,  
    height: 100,  
  
    floating: true, // make this panel an  
    absolutely-positioned floating component  
  
    title: 'Test',  
    html: 'Test Panel'  
});
```

- ebben az esetben a pozícionálás absolute CSS pozícionálással történik
- ahhoz, hogy egy komponens megjelenjen, általában szükséges a `renderTo` beállítása vagy pedig egy már megjelenített Container gyermekeként kell létrehoznunk a komponenst, Floating Component-ek esetén ez nincs így
- Floating Component-ek automatikusan a body gyermekeként jönnek létre, amikor meghívásra kerül a `show` metódusuk

- további pozícionálással kapcsolatos beállítási lehetőségek:
 - **draggable**: egy Floating Component esetén engedélyezhető a drag-n-drop
 - **shadow**: egy Floating Component esetén az árnyék is testreszabható
 - **alignTo**: egy adott komponenshez igazíthatjuk a Floating Component-ünk
 - **center**: a Floating Component-et az őt tartalmazó Container közepére igazítja

Saját komponensek készítése

- egy új komponens elkészítésénél a legelső döntésünk, hogy az osztály a Component egy példányával rendelkezzen vagy származzon a Component-ből
- alapszabályként érdemes arra törekedni, hogy a kívánt funkcionalitáshoz legközelebb eső osztályt érdemes ősoosztályként használni:
 - így kihasználhatjuk az ExtJS által biztosított automatikus életciklus menedzsmentet, automatikus méretezést és pozícionálást, illetve a Container-ből történő eltávolításkor a megsemmisítést

- egyszerűbb megírni egy új osztályt, amely egy Component és ezért benne van a komponenshierarchiában, szemben egy olyan osztállyal amely egy Component-et tartalmaz és a megjelenítését nekünk kell vezérelnünk
- származtatás:

```
Ext.define('My.custom.Component', {  
    extend: 'Ext.Component'  
});
```

Röviden az ExtGWT-ről

- a GWT kiterjesztése, ezért alapvető GWT ismeretek elengedhetetlenek
- új package: `com.extjs.gxt`
- hivatalos tutorial-ok:
<http://www.sencha.com/learn/extgwt>
- API:
<http://www.sencha.com/gxtdocs/#com.extjs.gxt.ui.client>

- eclipse és netbeans alá is könnyen telepíthető
- alapvetően elegendő felületes ExtJS ismeret is, de ha bővítenünk kell az ExtGWT komponenskészletét, akkor nagyon jól jön az ExtJS behatóbb ismerete
- látványos példák forráskóddal:
<http://www.sencha.com/examples/#overview>

Ellenőrző kérdések

1. Mik az előnyei az ExtJS Framework-nek?
2. Mi a könyvtárszerkezete egy ExtJS-t használó alkalmazásnak?
Az egyes könyvtárak és fájlok mit tartalmaznak?
3. Vázolja az ExtJS MVC modell-jét!
4. Milyen speciális komponest ismer?
Röviden írja le, melyik mire való és miben különbözik a Component-től!
5. Rajzolja le és magyarázza meg az ExtJS komponenshierarchiáját!
6. Magyarázza el, hogy mi az xtype és hogyan használható Lazy Initialisation-höz! Mi az előnye a Lazy Initialisation-nek?
7. Hogyan pozícionálhatunk ExtJS komponenseket?
8. Hogyan készíthetünk saját ExtJS komponenseket?