

Javascript + Java EE

Horváth András

NetVisor Zrt.

Bevezető

- Cél: szoftver eladása / support-álása minél hosszabb ideig
- Eszközünk: Java EE 3-tier alkalmazás:
 - 1.Database tier (nálunk: Oracle)
 - 2.(Business) Logic tier (nálunk: J2EE)
 - 3.Presentation tier (jellemzően web-es)**

- Az első két réteg határozza meg döntően az alkalmazás teljesítményét
- A felhasználók viszont (szinte) csak a GUI-n keresztül kerülnek kapcsolatba az alkalmazással
- Hiába cutting-edge az alkalmazás, ha csúnya/lassú a GUI, sokkal nehezebben eladható
- Emiatt törekedni kell arra, hogy a GUI minél „jobb” legyen

Milyen a „jó” GUI?

- Gyors
- Még gyorsabb :)
- Kényelmes
- „Szép” (erről főleg a grafikusok gondoskodnak)

Milyen eszközeink vannak?

- Plain JSP / JSF: elavult
- Plain JavaScript: **ne, hatalmas szívás!!!**
- Javascript Framework: ExtJs, Dojo etc.
- JSF Framework + Javascript: Richfaces etc.
- Portlet + Javascript: Liferay
- Java kódot írunk és abból generálódik a Javascript alapú GUI: pl: GWT, Echo 2 és 3

DOJO Framework

Bevezető

DOJO:

- Javascript GUI Framework
- Rengeteg kész GUI komponens, egy-két trükkös Javascript kiegészítés igényesebb kód írásához
- Hivatalos homepage, példákkal, dokumentációval:
<http://www.dojotoolkit.org/>

Előnyei

- Ingyenes, nagyjából browser független
- Open source
- Aktívan fejlesztik
- Teljes kontrollt kapunk a GUI elkészítéséhez (vs pl.: Echo)
- Tud „Lightweight” is lenni (elég csak azokat a komponenseket betölteni, amik kellenek)

Hátrányok

- Vannak bugok
- A dokumentáció sok helyen hiányos (fórumok / tutorial-ok viszont általában használhatóak)
- Új verziók kiadása esetén nincs backward compatibility
- Az ExtJS népszerűbb
- Javascript => még így is van vele szívás

Getting started

1. A dojo hivatalos oldalán lévő release-re hivatkozunk

```
<script src="
http://ajax.googleapis.com/ajax/libs/dojo/1.6.1/dojo/dojo.xd.js">
</script>
```

2. Letöltjük az egész src-t és azt használjuk

Gyors teszt:

```
<script>
    dojo.addOnLoad(function() {alert("dojo running")})
</script>
```

```
<script type="text/Javascript" src="../../dojo/1.4.2/dojo/dojo.js"
djConfig="parseOnLoad: true, isDebug: true,
gfxRenderer:'svg,silverlight,vml', myParam: true"></script>
```

- **parseOnLoad**: true vagy false

kétféle módon lehet létrehozni dojo-s gui elemeket:

a) html markup

b) Javascript kódból

Ha true akkor parse-olja betöltés után a dom fát, ha talál dojo-s gui komponenseket, akkor létrehozza őket

```
<div dojoType="dijit.TitlePane" title="Title Pane #1"
    tooltip="I'm the tooltip for Title Pane #1's title bar"
    style="width: 300px;" jsId="panel" id="testPanel">
    ...
</div>
```

vagy

```
var x = new dijit.TitlePane({id: "myTitlePane", title: "test",
style: "...", stb});
```

//A dom-ban lennie egy div-nek melynek „myTitlePane” az id property-je!!!

- isDebug: true vagy false (firebug-hoz)

Log-olás:

```
console.log(Obj)
```

- gfxRenderer: bizonyos komponens-ek (pl.: chart-ok) és a saját, egyszerű alakzatokra (circle stb.) épülő komponensek rendereléséhez

Ne felejtjük el leellenőrizni, hogy van-e megfelelő plugin:

```
function checkForSilverlight(){
    if (navigator.appName=="Microsoft Internet Explorer") {
        var isSilverlightInstalled = false;
        try{
            var silverlightControl = new ActiveXObject('AgControl.AgControl');
            isSilverlightInstalled = true;
        }catch (e){
            //Either not installed or not IE 7 or newer.
            if(navigator.plugins["Silverlight Plug-In"])
                isSilverlightInstalled = true;
        }
        if(!isSilverlightInstalled)
            dijit.byId('dialogNoSilverLight').show();
    }
}
```

- Saját paraméter használata:

```
dojo.config.myParam ? "something" : "something different"
```

Dojo, Dijit, Dojox

- Dojo: core működés, Drag and Drop support, dátumok kezelése (date), ajax support (dojo.xhrGet és dojo.xhrPost), ...
- Dijit: hétköznapi control-ok, amik kellenek egy egyszerű gui-hoz: gombok, menük, panel-ek
- Dojox:
 - minden olyan komponens, amihez valamilyen render-elés kell (chart, stb.)
 - dojox.math (BigInteger, Matrix stb.)
 - dojox.collections (arraylist, binarytree, set, stack, sortedlist, ...)
 - dojox.lang.oo.mixin
 - dojox.lang.aspect
 - dojox.lang.functional

- Nem mindegy, hogyan hivatkozunk a komponensekre:

```
dojo.byId("return_cats") <= DOM elemet ad vissza
```

```
dijit.byId("return_objects") <= Dijit objektum-ot ad  
vissza
```

```
dojox.byId("return_dogs") nincs!!!
```

- Miután létrejön egy dijit elem, az bekerül a dijit.registry-be, innen ha már nincs rá szükségünk, el kell távolítani a
`dijit.registry.remove("id")`-vel

- A dijit és dojox objektumoknak van `.destroy()` metódusa, ez megszünteti őket, mint Javascript objektumokat (kb. `this = null`) és kitörli őket a dom fából is
(de nem hív `dijit.registry.remove("id")`-t!!!)

Hasznos DOJO nyelvi elemek

- Query: kb. mint a jQuery

```
dojo.query("[name^=myCheckBox]").forEach(function(node, index, arr) {  
    console.log(node.innerHTML);  
}); //mindent aminek a name property-je myCheckBox-al kezdődik
```

Rövidebben:

```
dojo.query("select", document).forEach("node.disabled = true;");  
//minden select tag-re
```

- eseménykezelés: `dojo.connect` **//dojo.disconnect ha már nem használjuk!!!**

```
function init(){  
    button = dojo.byId('myButton');  
    dojo.connect(button, 'onclick', 'myButtonOnClickFunction');  
}
```

- dojo.filter:

```
var filteredArray = dojo.filter(array, function(item) {  
    return item.property == "myStringCondition";  
});
```

- dojo.some:

```
if (dojo.some(array, function(item) {return item >= 42})) {  
    result = 'there are some, which are greater than 42';  
} else {  
    result = 'no element is greater than 42';  
}
```

- dojo.every: mint a dojo.some

- dojo.map:

```
var changedArray = dojo.map(array, function(item) {  
    return item + (item / 100) * 10; //10% increase  
});
```

AJAX

- Get:

```
dojo.xhrGet({  
    url: myurl,  
    handleAs: "json", //can be: text, Javascript or even xml  
    load: function(data, ioArgs){  
        for(var i in data){ //we know it's an array  
            console.log("key:" + i + " value: " + data[i]);  
        }  
    },  
    error: function(error, response){  
        console.log("Error message: " + error.message + " Url: " +  
response.url + " Status: " + response.xhr.status)  
    }  
});
```

- Post:

```
dojo.xhrPost({  
  url: myUrl,  
  postData: postData,  
  handleAs: "text",  
  load: function(data){ ... },  
  error: function(error, response){ ... }  
});
```

postData helyett lehet akár form:"someForm"

- Delete:

```
dojo.xhrDelete
```

- Put:

```
dojo.xhrPut
```

Data store

Bizonyos GUI elemek feltöltéséhez data store-okat használhatunk / kell használnunk (pl.: dijit.Tree)

- `dojo.data.ItemFileReadStore`

- json-t olvas be

- `dojo.data.ItemFileWriteStore`

- json-t olvas, a változtatásokat megpróbálja elküldeni a szervernek

Dijit, Dojox

- Minden dijit, dojox komponenst be kell töltenünk:

```
dojo.require("dijit.form.Button"); //it will  
download this first
```

```
dojo.require("dijit.form.TextBox"); //starts after  
the previous!!
```

//lehet a /util/buildscripts könyvtárban lévő script-ekkel egy-egy nagy dojo.js, dijit.js, dojox.js-t csinálni, amiben megmondhatjuk melyik javascript-ek kerüljenek ezekbe az állományokba (gyorsabban szed le 1 nagy js-t a browser, mint sok picit) + a változóneveket is cseréli \$1, \$2 -re + kiszedi a whitespace-eket, etc.

- Létrehozzuk őket:

```
<table dojoType="dojox.grid.DataGrid"
jsid="alarmsGrid" id="alarmsGrid"

    store="alarmsStoreTable" query="{ severity:
'*' }" rowsPerPage="35"

    onClick="viewChartForAlarm"

    style="height: 300px; overflow: auto;">
```

jsid: létrehoz egy globális változót ezzel a névvel, az az objektum van benne amit a `dijit.byId()` ad vissza

- Ha már nem kellene, akkor töröljük őket

Saját komponensek

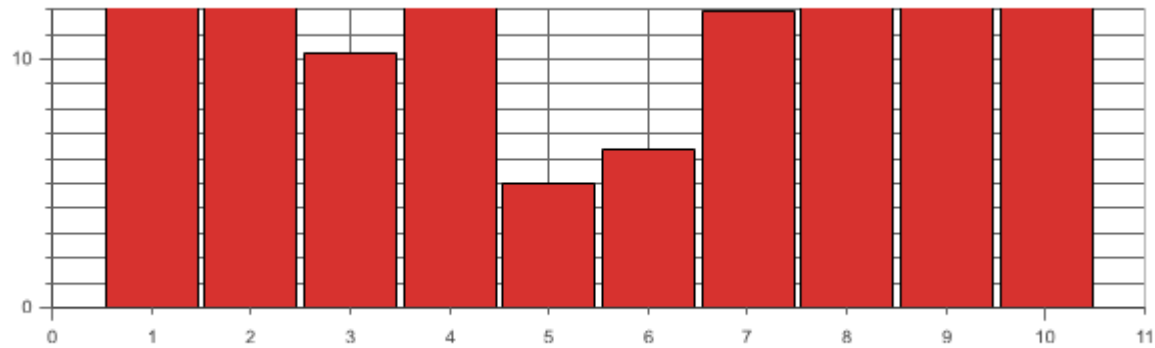
Probléma:

- Van egy komponens, ami majdnem jó nekünk.
- De csak majdnem.
- Kicsit bele kellene írni itt-ott, új funkciók, meglévők átalakítása.

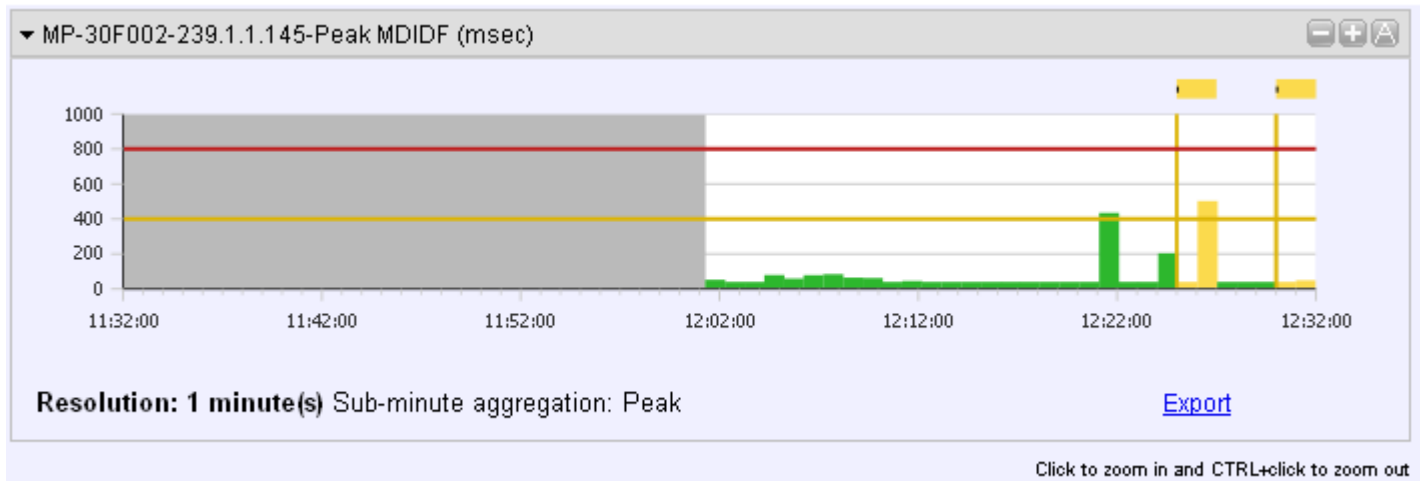
Megoldás:

- Elkezdjük átírni a komponenst => inkább ne :)
- Származtassunk belőle és írjuk át amit kell => ez a tuti

Ezt tudja a framework:



Az ügyfél pedig ezt akarja:



Öröklődés példa

```
if(!dojo._hasResource["netvisor.charting.NetvisorChart2D"]){  
  //like #ifndef  
  
  dojo._hasResource["netvisor.charting.NetvisorChart2D"] = true;  
  
  dojo.provide("netvisor.charting.NetvisorChart2D");  
  
  dojo.require(...); //include  
  
  (function() { //useful js methods  
  
    var df = dojox.lang.functional, dc = dojox.charting,  
  
    clear = df.lambda("item.clear()"),  
  
    purge = df.lambda("item.purgeGroup()"),  
  
    destroy = df.lambda("item.destroy()"),  
  
    makeClean = df.lambda("item.dirty = false"),  
  
    makeDirty = df.lambda("item.dirty = true");
```

```
dojo.declare("netvisor.charting.NetvisorChart2D",
dojox.charting.Chart2D, [inheritance] {

    //null => there is no parent

    //dojox.charting.Chart2D => inherits from this component only

    //[dojox.charting.Chart2D, dojox.charting.Chart3D] => multiple
inheritance

constructor: function(node, kwArgs){ ... }},

destroy: function(){

    dojo.forEach(this.series, destroy);

    dojo.forEach(this.stack, destroy);

    df.forIn(this.axes, destroy);

    this.surface.destroy();

},

...properties/methods...

})

})();

}
```

Ellenőrző kérdések

1. Mik az előnyei a Dojo Framework-nek?
2. Milyen csomagokra osztja a Dojo a komponenseket? Soroljon fel néhány komponenst mindegyik csomagból!
3. Ismertessen néhány (legalább 2) fontosabb djConfig paramétert.
4. Milyen módon hozhatunk létre Dojo komponenseket?
5. Ismertessen néhány hasznos Dojo nyelvi elemet.
6. Írjon egy rövid Dojo-s AJAX hívásra példát (get vagy post).
7. Hogyan írhatunk saját komponenst?