TOOLS

Het lijkt dat het de laatste jaren voor een beginnende Javaan er niet eenvoudiger op is geworden. Behalve de Java-taal zelf, word je ook geacht een diversiteit aan andere technieken te beheersen. (X)HTML, XML, Javascript, CSS, DOM, JSP, JSF, JPA, JSON, REST, XSLT, ANT en Maven om de meest gebruikte te benoemen. Daarbij buitelen de laatste jaren ook nog eens de verschillende frameworks over elkaar heen, die alle weer hun eigen specifieke technische kenmerken hebben.

Snel webapplicaties bouwen met Echo 2

Snel en gemakkelijk een SPA met Ajax

Echo2 bewijst dat het ook anders kan. Een GUI-framework, dat met alleen pure Java-kennis te gebruiken is en dat middels Ajax een Single Page Application (SPA) oplevert. Daar zit dan ook meteen de beperking. Het is alleen een front-end, dus alleen de View om in MVC termen te spreken. Daarentegen kan het ook juist je oplossing zijn voor het geval je een GUI laag zoekt op een gegeven Model en Controller.

Kenmerken

Echo2 van NextApp, http://echo.nextapp.com/site/, is een open-source Java-framework onder Mozilla Public License.. Gebruik van het framework voor zowel open als closed source is dan ook geheel gratis. Echo2 bestaat al een tijdje, sinds 2005, en is de opvolger van zijn voorganger, het minder bekende Echo (één) framework.

Echo2 draait op elke Java Servlet Container die de Servlet 2.3 specificatie ondersteunt. Aan de client kant ondersteunt Echo2 zowel W3C DOM / CSS Level 2 compliant browsers (Firefox, Chrome, Safari, Opera) als Internet Explorer van Microsoft.

Met Echo2 kun je webapplicaties bouwen met een interface die de mogelijkheden benadert van een moderne 'rich client interface', zoals diverse moderne JSF implementaties, Backbase en Google Web Toolkit (GWT). Het voordeel van Echo2 is een erg lage leercurve, eigenlijk heb je alleen basis Java-programmeervaardigheden nodig. Je kunt dus voor de gehele GUI-laag gebruik maken van je normale Java debugger, wat een verademing is in vergelijking met debuggen in de front end, de browser, of geïnterpreteerde View definitie-talen als JSP, Tapestry en JSF. Echo2 wordt vaak vergeleken met GWT, want met GWT bouw je ook je GUI-laag in Java. Er is echter een wezenlijk verschil. GWT biedt een beperkte Java api en 'compileert' jouw Java-code naar JavaScript dat gewoon in de browser draait. Runtime is GWT dan ook een client side framework. Bij Echo2 draait de GUI compleet op de server, waarbij via Ajax requests DOM updates gestuurd worden. Met Echo2 heb je dan ook 'gewoon' de beschikking over de normale, complete Java api en kun je ook gewoon Java libraries gebruiken, zoals je gewend bent. Dit in tegenstelling tot GWT.

Hoe doet Echo2 dit? Kort samengevat gebruikt Echo2 een class model op de server voor de GUI-laag, waarbij een servlet (met onderliggende Echo2 jar libraries) ervoor zorgt dat de updates op de GUI-laag naar de client (browser) worden gestuurd door middel van AJAX requests. Hier heb je als ontwikkelaar geen omkijken naar en dit wordt netjes voor je geregeld. Het enige dat je hoeft te doen, is je eigen Java class model van je GUI-laag schrijven op een wijze die je wellicht nog van 'vroeger' van Swing kent.

Eerste opzet

De overkoepelende class die de applicatie representeert is een ApplicationInstance. Voor elke webgebruiker wordt hiervan een instantie gecreëerd en op de sessie gezet. Deze abstract class is je startpunt. Door deze te extenden en de methode init te implementeren heb je je eigen Echo2 applicatie al. Deze init methode dient een Window object te retourneren die het startscherm van je applicatie bevat.

Emiel Paasschens
is senior software engineer bij AMIS en gespecialiseerd in Java en SOA oplossingen. Hij is te bereiken op Emiel.Paasschens@amis.nl.

Dynamisch toevoegen van buttons of tabellen is een 'piece of cake'!

```
public class EchoWorldApplication extends
ApplicationInstance {
    private final Window window = new Window();
    public Window init() {
        window.setTitle("Echo World");
        window.setContent(new EchoPane());
        return window;
    }
}
```

Aangezien je een Window weer vult met een ContentPane en deze telkens wisselt, is het in de praktijk niet nodig meerdere Window objecten te definiëren. Om dit wisselen gemakkelijk te maken, ligt een setContent methode natuurlijk erg voor de hand.

Het is raadzaam zelf een Controller te schrijven, die deze methode gebruikt. Deze controller heeft natuurlijk een referentie naar je ApplicationInstance nodig. Dit is te doen middels de static methode getActive van de abstracte ApplicationInstance class. Voor je zojuist gemaakte setContent methode zul je de ApplicationInstance toch moeten casten naar je eigen class definitie. Dit kunnen we dan ook meteen in een static methode van onze eigen ApplicationInstance doen.

Ook is dit een goede plek voor het zetten van een Echo2 stylesheet voor de look & feel. Hierover later meer.

```
public class EchoWorldApplication extends
ApplicationInstance {
    private final Window window = new Window();
    public Window init() {
        setStyleSheet(Styles.DEFAULT_STYLE_SHEET);
        window.setTitle("Echo World");
        window.setContent(new EchoPane());
        return window;
    }
    public void setContent(ContentPane content){
         window.setContent(content);
    }
    public static EchoWorldApplication getApplication(){
       return (EchoWorldApplication)getActive();
    }
}
```

Andere typische functionaliteit die zich leent voor je ApplicationInstance is het zetten van de user en zijn autorisaties. Je hebt immers een instantie per gebruiker!

Verdere opbouw

Om bovenstaande applicatie te voltooien hebben we de EchoPane en de Styles class nodig.

In de EchoPane , subclass van ContentPane, willen we een invoerveld hebben met een knop om de ingevoerde tekst te echo-en.

```
public class EchoPane extends ContentPane {
    private TextField txtName;
    private Label lblEcho;
    private Button btnEcho;
    public EchoPane() {
        super();
        Column column = new Column();
        add(column);
        Row row = new Row();
        txtName = new TextField();
        btnEcho = new PushButton("Echooo");
        btnEcho.setActionCommand("doEcho");
        btnEcho.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEventevent) {
              if (event.getActionCommand().equals("doEcho")) {
            lblEcho.setText(txtName.getText() + "..." + txtName.
getText());
                }
            }
        });
        row.add(txtName);
        row.add(btnEcho);
        column.add(row);
        lblEcho = new Label("hoi");
        column.add(lblEcho);
    }
}
```

Een ContentPane dient eerst gevuld te worden met één 'container' object, een Column, Row of een Grid. Aan deze containers kunnen we vervolgens visuele componenten toevoegen, maar ook weer container objecten of een mix van beide. Zodoende kun je complete hiërarchieën opbouwen en alle denkbare schermindelingen bouwen op een vrij eenvoudige manier. Het spreekt voor zich dat je door het zetten van properties allerlei eigenschappen kunt beïnvloeden, zoals uitlijning, randen, styles, kleuren, zichtbaarheid etc. Aangezien dit gewoon Java-code is, is dit ook heel gemakkelijk dynamisch, op runtime te doen. Dit geldt niet alleen voor het geven van verschillende waardes aan properties onder verschillende omstandigheden, maar ook voor het creëren of opruimen van object instanties en het toevoegen of verwijderen aan windows is mogelijk! Zo is het dynamisch toevoegen of verwijderen van bijvoorbeeld buttons of tabellen dan ook een 'piece of cake'! Een ander treffend voorbeeld is het maken van evenveel invoervelden als items in een list.

```
        List<String> list = getNames();
        TextField textfield;
        Label label;
        Row row;
        Map<String, TextField> map = new
HashMap<String,
TextField>(list.size());
        for (Iterator<String> i = list.iterator(); i.
hasNext();){
         String name = i.next();
         label = new Label(name);
         textfield = new TextField();
         map.put(name, textfield); //voor uitlezen
         row = new Row();
         row.add(label);
         row.add(textfield);
         column.add(row);
        }
```

Schermnavigatie en caching met Controller

Zoals al benoemd is het raadzaam om zelf een centraal Controller object te schrijven voor navigatie. Deze controller kun je in geval van autorisatie ook gebruiken om navigatieregels af te dwingen en/of om contentPanes in een zelfgeschreven (autorisatie)mode te zetten, bijvoorbeeld een readonly mode waarbij je controls disabled of onzichtbaar maakt. Een valkuil waar je bij Echo2 voor moet waken, is dat je snel geneigd bent om business logica in de pagina's, Panes, te stoppen in

plaats van dit te beperken tot schermafhandeling. Laten we in bovenstaand voorbeeld navigeren naar een Paasei scherm als de gebruiker paasei intypt. De actionlistener van een button gaat af bij het indrukken van de button. We controleren in de listener de waarde van het textveld en in het geval van een paasei, vragen we de controller de navigatie te doen.

```
        btnEcho.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent
event) {
                if (event.getActionCommand().
equals("doEcho")) {
                    if ("paasei".
equalsIgnoreCase(txtName.getText())){
                            EchoWorldApplication.
getController().goEasternEgg();
                    }
                    else{
                            lblEcho.setText(txtName.
getText() + "..." + txtName.getText());
                    }
                }
            }
        });
```

De Controller stuurt onze navigatie. Dit gebeurt simpelweg door de window met een andere Pane te vullen.

```
package nl.amis.echoworld;
public class Controller {
        private final EchoWorldApplication app =
EchoWorldApplication.getApplication();
        Controller(){
        }
        public void goEasternEgg(){
                app.setContent(new PaaseiPane());
        }
}
```

Je kunt nu ook heel gemakkelijk schermen gaan cachen, hergebruiken door de schermreferenties in een Hashtable te zetten. Bij een paginawisseling hoeft dan niet meer een nieuwe Pane instantie gecreëerd te worden, maar kun je de vorige weer herbruiken. Dit scheelt tijd, maar kost natuurlijk weer wel wat geheugen. Implementeer hiervoor in elke pane een methode die een unieke naam teruggeeft (definieer hiervoor bij voorkeur een interface). Let op! Hiermee zul je dus ook de state van een scherm bewaren en krijgt de gebruiker zijn scherm terug in dezelfde staat als toen hij de pagina verliet. Dit is soms prettig en gewenst, maar soms ook niet. In het laatste geval kan een zelf te implementeren refresh methode (interface!) uitkomst bieden.

Echo2 Style

Echo2 heeft zijn eigen manier om een style te definiëren. Dit is een xml bestand, dat wordt ingelezen. Het komt erop neer dat alle properties die op een component zijn te zetten omtrent layout en gedrag in een xml variant zijn gedefinieerd. Opmerkelijk is dat de standaard extensie van het bestand .stylesheet. is.

```
<style name="ControlPane.Button" type="nextapp.echo2.
app.Button">
  <properties>
    <property name="lineWrap" value="false"/>
    <property name="foreground" value="#000000"/>
    <property name="rolloverForeground"
 value="#6f0f0f"/>
    <property name="rolloverEnabled" value="true"/>
    <property name="insets" value="8px 0px"/>
    </properties>
</style>
```

Het inlezen gebeurt middels een stukje Java code. Hieronder de implementatie van de Styles class uit ons voorbeeld:

```
public class Styles {

    public static final String STYLE_PATH = "/nl/amis/
echoworld/resource/style/";

    public static final StyleSheet DEFAULT_STYLE_SHEET;

    static {
        try {

DEFAULT_STYLE_SHEET = StyleSheetLoader.load(STYLE_PATH +
"Default.stylesheet",
                                    Thread.currentThread().
getContextClassLoader());
        }
        catch (ComponentXmlException ex) {
            throw new RuntimeException(ex);
        }
    }
}
```

Nu bestaat er gelukkig een uitbreiding op echo2, genaamd echopointing. Deze library bevat allerlei handige classes, waaronder rijkere visuele componenten, zoals bijvoorbeeld een autosuggest invoerveld, maar ook een CssStyleSheetLoader class die een css kan inlezen in een Echo2 StyleSheet class.
Ook binnen een echo2 stylesheet kun je namen geven aan een style en ook deze dynamisch wisselen!
Laten we in ons voorbeeld een Ugly style definiëren van een invoerveld:

```
<style name="Ugly" type="nextapp.echo2.app.TextField">
<properties>
  <property name="background" value="#000000"/>
  <property name="foreground" value="#ffff00"/>
</properties>
```

Nu kunnen we deze gaan gebruiken in ons EchoPane voorbeeld:

```
    private boolean isUgly = false;

    private void toggleUgly(){
        if (isUgly)
            txtName.setStyleName("Default");
        else
            txtName.setStyleName("Ugly");

        isUgly = !isUgly;
    }
```

Aanroep van toggleUgly in de ActionListener:

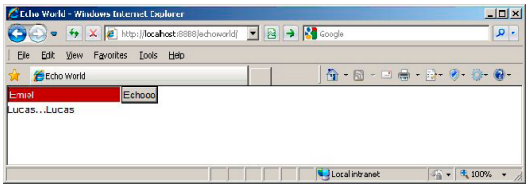
```
        btnEcho.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent
event) {
                    if (event.getActionCommand().
equals("doEcho")) {
                      if ("paasei".
equalsIgnoreCase(txtName.getText())){
```

Properties die je op een component kunt zetten zijn in een xml variant gedefinieerd.

Voor Eclipse is tegen betaling een plugin verkrijgbaar

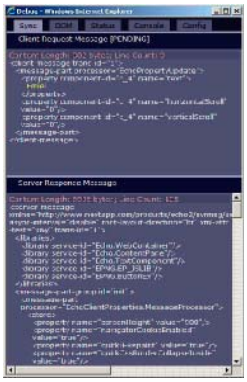
```
                                    EchoWorldApplication.
getController().goEasternEgg();
                        }
                        else{
                                    lblEcho.setText(txtName.
getText() + "..." + txtName.getText());
                                    toggleUgly();
                        }
                    }

                }
            });
```



Ajax debugging

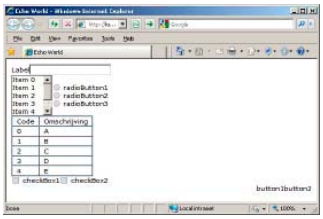
Echo2 biedt een debug feature waarmee je kunt bekijken wat er over de lijn gestuurd wordt. Dit doe je door eenvoudig '?debug' achter je url in de browser te plakken. Er verschijnt dan een popup venster waarmee je het Ajax request en de response kunt inspecteren alsmede de DOM structuur van je pagina en nog wat additionele informatie. Zelf heb ik dit eigenlijk nog nooit nodig gehad hetgeen een goed teken is dat het synchroniseer mechanisme van Echo2 blijkbaar feilloos werkt!



Echo2 debug window.

Visuele Componenten

De visuele componentenset van Echo2 is op zich vrij beperkt: Button, Checkbox, Label, ListBox, PasswordField, RadioButton, SelectField, Table, TextArea, TextField, Image. Verder zijn er wat layoutcomponenten, Pane, SplitPane, Panel, Column, Row en Grid.



Standaard componenten zonder enige styling.

Gelukkig bevat de echopointng library een flinke uitbreiding, van een Separator (spacer) tot KeyStrokeListeners en van een AutoLookupTextField tot eenTree component.
Wat daarbij zeer krachtig is, is dat het erg eenvoudig is om een component uit te breiden met functionaliteit en je zo je eigen componenten kunt bouwen. Te denken valt aan eenvoudige default properties die je in de constructor zet, bijvoorbeeld een button met standaardbreedte. Maar ook een samengesteld component dat meerdere andere componenten bevat is mogelijk, zoals een MessageDialog die als een 'DIV' embedded in je pagina verschijnt of een pageable table met de bekende knoppen om een set records vooruit, achteruit, naar de eerste, de laatste of een zelf in te geven nummer te bladeren.

Eclipse plugin

Voor Eclipse is tegen een lichte betaling een plugin verkrijgbaar waarmee je visueel pagina's en Echo2 xml Stylesheet kunt maken en wijzigen.

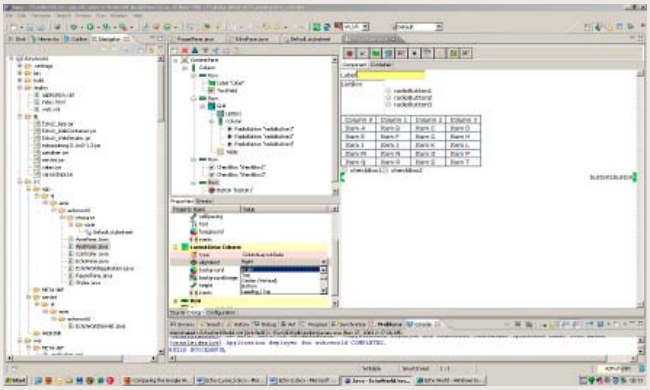
Met de visuele pagina editor, kun je in de linkerbovenkant met behulp van de componenten structuur je pagina in elkaar klikken. Eronder kun je de properties van het geselecteerde component zetten en wijzigen. In de rechterkant zie je direct het resultaat.
De visuele editor genereert bij opslaan de bijbehorende Java-code in de methode initComponents. Deze methode wordt dan weer aangeroepen in de constructor van de pagina. Van deze methode met gegenereerde Java dien je wel af te blijven, aangezien de methode weer wordt overschreven bij een volgende opslaan actie en je dan je eigen code kwijt bent. Dit is natuurlijk eenvoudig te omzeilen door je post generatie wijzigingen in een andere methode te zetten die je in de constructor aanroept na de initComponents methode. De visuele editor werkt onder water eigenlijk stiekem met een xml file. De visuele editor leest deze xml in bij opstarten en schrijft wijzigingen er ook weer in weg waarna vervolgens de Java code gegenereerd wordt. Het is dus belangrijk om dit bestand met de extensie .form mee te nemen in een source control tool.

Bovenstaand geldt niet voor de visuele Style Editor. Deze wijzigt wel direct de onderliggende xml.

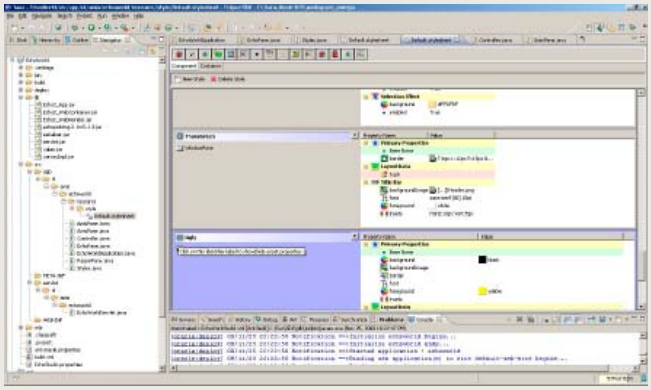
Web huishouding

We moeten de applicatie natuurlijk web enabled maken door hem 'in te pluggen' in een web applicatie. Dit is vrij eenvoudig door in de web.xml de binnenkomende requests naar een servlet te mappen. Deze servlet is een subclass van een abstract echo2 servlet waarbij we simpelweg refereren naar onze ApplicationInstance.

```
<web-app>
```



Echo2's visuele component editor in Eclipse.



Echo2's visuele Style edtior in Eclipse.

```
    <display-name>echoworld</display-name>
     <servlet id="echoworld-servlet">
       <servlet-name>echoServlet</servlet-name>
<servlet-class>nl.amis.echoworld.EchoWorldServlet</
servlet-class>
     </servlet>
     <servlet-mapping id="echoworld-servlet-mapping">
       <servlet-name>echoServlet</servlet-name>
       <url-pattern>*</url-pattern>
     </servlet-mapping>

</web-app>
```

web.xml

```
public class EchoWorldServlet extends
WebContainerServlet {
    public ApplicationInstance newApplicationInstance()
{
        return new EchoWorldApplication();
    }
}
```

De echo2 subclass servlet

Voor de volledigheid hieronder de opsomming van de benodigde libraries:

- Echo2_WebRender.jar
- Echo2_WebContainer.jar
- Echo2_App.jar
- servlet.jar
- echopointng-2.1rc5-1.3.jar (optional, containing some more extended gui components).

PDF Generatie

Een nadeel van Echo2 dat uit de praktijk bleek, was dat het zo goed als onmogelijk is om een tweede browser scherm in de applicatie te openen. Nu is dit binnen een SPA eigenlijk ook vrijwel nooit nodig. De oplossing voor ons praktijk probleem van een printer vriendelijke versie van een pagina, was een pdf te genereren. Dit bleek erg eenvoudig te implementeren te zijn, juist doordat de componenten 'gewone' Java classes waren! Door een eigen interface te definiëren met methode

```
public com.lowagie.text.Section toPdf(com.lowagie.text
Section activeSection);
```

en deze te implementeren op de componenten die geprint moesten worden, waren binnen een dag alle rapportage pagina's naar pdf te renderen!

Conclusie

Echo2 is een zeer eenvoudig GUI framework waarmee gemakkelijk en relatief snel een webapplicatie kan worden gebouwd. Hierbij moet worden opgemerkt dat het Echo2 alleen de voorkant dekt, alleen de View van MVC. Dit hoeft geen probleem te zijn bij niet al te ingewikkelde applicaties. Een controller en het data model kun je gemakkelijk zelf schrijven, al dan niet gebruik makend van ondersteunde frameworks en technieken.

Ook dient opgemerkt te worden dat de gehele GUI laag op de web server draait hetgeen bij hoge belastingen een probleem kan worden. De grafische- en Ajax mogelijkheden zijn enigszins beperkt als ze worden vergeleken met moderne Java Server Faces frameworks.

Toch ben ik ervan overtuigd dat voor menig intranet applicatie binnen een bedrijf Echo2 een goede en verdedigbare keuze kan zijn. Het is een gratis framework waarmee je snel en gemakkelijk mooie interfaces kunt bouwen. Daarbij is de leercurve van Echo2 extreem kort. Met de bijkomende voordelen van server side ontwikkelen, ben je dan al snel en productief aan de slag. Echo2 is dan ook ideaal voor een web applicatie waarbij je wel een acceptabele GUI wenst, maar deze niet 'top of the bill' hoeft te zijn; waarbij het aantal gebruikers en dus de load niet extreem hoog is en waarbij de business logica enigszins beperkt is zodat een Controller en Model snel zelf te bouwen zijn.

Echo3

Inmiddels is Echo3 al een tijdje beschikbaar in een Beta versie. Echo3 belooft een mix van Echo2 en client side frameworks, zoals GWT en Wicket. NextApp, de maker van Echo2 en 3, belooft op de site dat de severside api ten opzichte van Echo2 zo goed als ongewijzigd gebleven, hetgeen een overstap voor ontwikkelaars gemakkelijk maakt. Ook het migreren van een Echo2 applicatie naar Echo3 zou volgens de makers niet al te moeilijk moeten zijn. Wellicht eens tijd om daar eens mee te gaan spelen. To be continued dus … «

Eenvoudig framework om snel een applicatie voor het web te bouwen.

Links

Al met al is het toch mogelijk om met Echo2 een zeer acceptabele en mooie interface te bouwen. Een demo applicatie die de vele mogelijkheden van de Echo2 GUI toont, is te bekijken op de site van NextApp: http://demo.nextapp.com/Demo/app.