TrinidadSeamAjax4JsfFaceletDetail

Configure Trinidad for use with Seam, ajax4jsf and Facelets

In a complex environment with a lot of different libraries to play together and therefor use the best of all to fullfil your requirements, there are some things to keep in mind. This is a short description on how I configured my application to make these components run together.

There have been some major changes between Seam 1.1.6 and Seam 1.2.0, so you have to have two different configurations for the use of each Seam versions. I´ll try to cover both of them - but it´s just a reference on how configured everything to make it run together. Maybe you also have some improvements - then feel free to add them here.

I´m referencing to the following environment:

- Trinidad Nightly Build from Apache Incubator http://people.apache.org/repo/m2-snapshot-repository/org/apache/myfaces/trinidad/
- Ajax4jsf 1.0.6 https://ajax4jsf.dev.java.net/
- Facelets 1.1.12 https://facelets.dev.java.net/
- MyFaces 1.1.5 http://myfaces.apache.org/
- JBoss Application Server 4.0.5 http://labs.jboss.com/

You will find both versions of JBoss Seam at http://www.jboss.com/products/seam

To make Trinidad play together with Ajax4jsf you also have to include the `a4j-trinidad.jar`. The a4j-trinidad contains an alternative AjaxContext called TrinidadAjaxContext which is not present in the main distribution, so you will need the extra library additional to the ajax4jsf-library. You will find this at the ajax4jsf-website as well.

Therefor you should have the following jars additional to the usually required jars, the Seam-jars for your version and the jars of your JSF-implementation:

- trinidad-api-incubator-m1-SNAPSHOT.jar
- trinidad-impl-incubator-m1-SNAPSHOT.jar
- a4j-trinidad.jar
- ajax4jsf.jar
- oscache-2.3.2.jar
- jsf-facelets.jar

All of these jars should go into your `/WEB-INF/lib` directory. Make sure, that your JBoss also has the libs of your JSF-implementation in the

```
<JBOSS-DIR>/server/default/deploy/jbossweb-tomcat55.sar\jsf-libs
```

. All the other jars don´t have to go there.

Configure Trinidad for use with Seam 1.2.0, ajax4jsf and Facelets

faces-config.xml

Let´s start with the `faces-config.xml`. It is very neccessary to remove all `view-handler}}s and instead set the {{default-render-kit` to Trinidad. Then the faces-config.xml should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN" "http://java.sun.
com/dtd/web-facesconfig_1_0.dtd">

<faces-config>

    <!-- Phase listener needed for all Seam applications -->
    <lifecycle>
        <phase-listener>org.jboss.seam.jsf.SeamPhaseListener</phase-listener>
    </lifecycle>

    <application>
        <default-render-kit-id>org.apache.myfaces.trinidad.core</default-render-kit-id>
    </application>

</faces-config>
```

web.xml

Let´s go over to the `web.xml`. First we have to define a special set of `listeners`. I don´t think the order is that important in this case, but I started with the SeamListener, which is declared as follows:

```
    <listener>
        <listener-class>org.jboss.seam.servlet.SeamListener</listener-class>
    </listener>
```

You don´t have to define any additional listeners for your JSF-implementation as these are listed inside the TLD for the core lib (<f:xxx>) and the web-container should see the Listener, when parsing the TLD, during deployment.

That´s why you can remove listeners like:

```
    <!-- MyFaces listener -->
    <listener>
        <listener-class>org.apache.myfaces.webapp.StartupServletContextListener</listener-class>
    </listener>
    <!-- Sun RI listener -->
    <listener>
        <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
    </listener>
```

from your web.xml.

For the most of the libraries you don´t have to set any special context-parameters, but when it comes to the Trinidad {{context-param}}s the following declaration is necessary:

```
    <context-param>
        <param-name>org.apache.myfaces.trinidad.ALTERNATE_VIEW_HANDLER</param-name>
        <param-value>com.sun.facelets.FaceletViewHandler</param-value>
    </context-param>
```

Alternatively you can also use the `org.jboss.seam.ui.facelet.SeamFaceletViewHandler` as the ALTERNATE_VIEW_HANDLER. The SeamFaceletViewHandler has an extension for the EL and therefor can handle action calls with parameters etc.

When it comes to the `filter-mapping` the order is very important. You have to define the ajax4jsf-filter-mapping as the very first one, followed by the others. Be careful, that you don´t mix up `url-pattern` and `servlet-name` with your `filter-mapping` but only stay with one of them or you might run into some trouble. (See note from Carsten Höhne at the very bottom of this page)

This is the sequence of filter-mappings I used in my web.xml:

```
<!-- ********************** AJAX4JSF Filter *************************** -->
    <!-- If you have other filters declared in the web.xml, be sure that Ajax4jsf Filter is declared before the
others. -->
    <filter>
        <display-name>Ajax4jsf Filter</display-name>
        <filter-name>ajax4jsf</filter-name>
        <filter-class>org.ajax4jsf.Filter</filter-class>
        <init-param>
            <param-name>forceparser</param-name>
            <param-value>false</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>ajax4jsf</filter-name>
        <url-pattern>*.seam</url-pattern>
    </filter-mapping>

    <!-- ********************** Trinidad Filter *************************** -->
    <filter>
        <filter-name>Trinidad</filter-name>
        <filter-class>org.apache.myfaces.trinidad.webapp.TrinidadFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>Trinidad</filter-name>
        <!--<servlet-name>Faces Servlet</servlet-name>-->
```

```
    <url-pattern>*.seam</url-pattern>
</filter-mapping>


<!-- ********************** Seam Filter *************************** -->
 <filter>
    <filter-name>Seam Filter</filter-name>
    <filter-class>org.jboss.seam.web.SeamFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>Seam Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- ********************** SEAM Servlet Mapping ********************** -->
<servlet>
    <servlet-name>Seam Resource Servlet</servlet-name>
    <servlet-class>org.jboss.seam.servlet.ResourceServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>Seam Resource Servlet</servlet-name>
    <url-pattern>/seam/resource/*</url-pattern>
</servlet-mapping>


<!-- ********************** Faces Servlet Mapping ***************************** -->
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.seam</url-pattern>
</servlet-mapping>


<!-- ********************** Trinidad resources servlet ***************************** -->
<servlet>
    <servlet-name>Trinidad Resources</servlet-name>
    <servlet-class>org.apache.myfaces.trinidad.webapp.ResourceServlet</servlet-class>
</servlet>

<!-- This cannot be configured currently -->
<servlet-mapping>
    <servlet-name>Trinidad Resources</servlet-name>
    <url-pattern>/adf/*</url-pattern>
</servlet-mapping>
```

sample-web-1-2-0.xml

Configure Trinidad for use with Seam 1.1.6, ajax4jsf and Facelets

faces-config.xml

The faces-config.xml stays the same between both versions, so you can look at the faces-config.xml from the Seam 1.2.0 section.

web.xml

The main difference between 1.1.6 and 1.2.0 is in the filter mapping. As in Seam 1.2.0 they only have the `org.jboss.seam.web.SeamFilter` you have to define the
`org.jboss.seam.servlet.SeamExceptionFilter` and `org.jboss.seam.servlet.SeamRedirectFilter` in 1.1.6. For the rest you can set the same declaration as in 1.2.0 listed above.

So this is I how it worked for me:

```
    <!-- ********************** AJAX4JSF Filter ************************** -->

    <!-- If you have other filters declared in the web.xml, be sure that Ajax4jsf Filter is declared before the
others. -->

    <filter>
        <display-name>Ajax4jsf Filter</display-name>
        <filter-name>ajax4jsf</filter-name>
        <filter-class>org.ajax4jsf.Filter</filter-class>
        <init-param>
            <param-name>forceparser</param-name>
            <param-value>false</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>ajax4jsf</filter-name>
        <url-pattern>*.seam</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>INCLUDE</dispatcher>
    </filter-mapping>


    <!-- ********************** Trinidad Filter ************************** -->

    <filter>
        <filter-name>Trinidad</filter-name>
        <filter-class>org.apache.myfaces.trinidad.webapp.TrinidadFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>Trinidad</filter-name>
        <url-pattern>*.seam</url-pattern>
    </filter-mapping>


    <!-- ********************** Seam Filter ************************** -->

    <filter>
        <filter-name>Seam Exception Filter</filter-name>
        <filter-class>org.jboss.seam.servlet.SeamExceptionFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>Seam Exception Filter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <filter>
        <filter-name>Seam Redirect Filter</filter-name>
        <filter-class>org.jboss.seam.servlet.SeamRedirectFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>Seam Redirect Filter</filter-name>
        <url-pattern>*.seam</url-pattern>
    </filter-mapping>
```

This is all I did and it works quite nice and smoothly.

Hope it helps someone to run these great libraries together, which makes development really smooth and easy. For further reference I attached my complete web.xml for Seam 1.1.6, if someone doesn´t know where to put all the declarations.

Have fun!

Thomas Hamacher

sample-web.xml

I do have experienced some problem when configuring Seam and Ajax4Jsf this way. The problem was that conversation state was not propagated properly. **Why?**
Because url mapping has higher precedence than servlet mapping. So in the above config the Seam redirect filter comes before the Ajax4Jsf filter. To avoid this pitfall consistent mapping is required.

First option: Map Ajax4jsf via url mapping

```
    <filter-mapping>
        <filter-name>ajax4jsf</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Second option: configure Seam with servlet mappings.

BTW:

With Seam 1.2.0 the filter mappings have changed. Ther is now a Seam Master filter which has to be configured.

Carsten Höhne