Fragment Modules for OOTB Customizations

This section is all about basic definition of Fragment in the exclusive perspective of Liferay 7 DXP with its usage in overriding several actions or entities majorly required for OOTB customization. This section will cover:

➔ *What is Fragment?*
➔ *How to create Fragment in Liferay 7*
➔ *Usage of Fragment in Liferay 7 DXP to:*

- *Override Liferay OOTB Module's JSP*
- *Override Language properties*
- *Override HTTP request with Servlet Filter*
- *Override Liferay Struts actions*
- *Override Liferay Modal Listener*

What is Fragment?

Fragment is a type of OSGi module similar as mvcportlet, service, activator, panel app etc. It can be said thatFragment is an extension of host module. Liferay has many out of the box modules and sometimes we need to override then based on our requirements. OSGi fragment is the way to override it.

For all versions prior to Liferay 7, Hook was being used to override Liferay portlets. But for Liferay 7, we have Fragment as Hook.

As Fragments are extends host modules, as part of host module, at the time of deployment, fragment module definitions will be merged in host module (only if, fragment module does not create any conflict with host module). If any conflict is created, fragments will not include in host module till its resolution. A fragment does not have its own class loaded or bundle activator.

Fragment jar have own OSGi manifest file and that file contains information about OSGi. Please find the significant information of MANIFEST.MF below.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Liferay Fragment
Bundle-SymbolicName: azilen.login.fragment.module
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Fragment-Host:  com.liferay.login.web;bundle-version=1.0.5
Import-Package: org.apache.commons.logging;version="1.0.4"
Export-Package: com.azilen.training;version="1.0.0"
```

Please find the details of its key properties below.

Fragment-Host: The Bundle Symbolic Name of the host module
Bundle-Version : Initial version of the bundle
Bundle-Version : Human readable fragment module name
Bundle-SymbolicName : Unique identifies the fragment in OSGi container.
Import-Package : External packages which are used in Fragment
Export-Package: Fragment packages which are visible to others modules

Fragment in Liferay 7 DXP

Liferay 7 uses Fragment for below scenarios.

1. Override Liferay OOTB module's JSP
2. Override Language properties
3. Override HTTP request with Servlet Filter
4. Override Liferay Struts actions
5. Override Liferay Modal Listener
6. Override Liferay Action Command

How to create Fragment in Liferay 7

To create a Liferay fragment as a module, use below command

blade create -t fragment [-h hostBundleName] [-H hostBundleVersion] projectName

Please note that, we have to specify the type of blade module as fragment with -t parameters. -h parameter will require host bundle Symbolic name and -H parameter will require version of the host module which is going to be overridden. And last projectName is human friendly fragment module name.

Override Liferay OOTB Module's JSP

Let's take a Use Case and I want to override Liferay Login module and I want to customize the look and feel according to my requirements. So the blade command to be created will be as below:

blade create -t fragment -h com.liferay.login.web -H 1.0.5 login-fragment-module

After this command it will create one skeleton structure of fragment and you can see host module (log in module) symbolic name and version in bnd.bnd file.

There can be a query as How to find host module symbolic name and version. So for that we have to connect gogo shell with command written.

telnet localhost 11311

Then run lb command and it will list all modules deployed in server. You can find host module with his status and version.

219|Active | 10 | Liferay Login Web (1.0.5)

Our fragment structure is ready now and we are all set to override host module jsp. As we need to override login.jsp, let's copy it from Liferay-src/modules/apps/foundation/login/login-web/src/main/resources/META-INF/resources/login.jsp and paste it in login-fragment-module/src/main/resources/META-INF/resources/.

Now modify login.jsp as your requirement and deploy login-fragment-module, a after successful deployment, you should get your changes in OOTB login module.

Override Language properties

Liferay support multi language so we have language properties for OOTB modules and suppose we want to change properties values for some labels, error messages, success messages etc then we can override those properties value and can change as our requirements.

Lets suppose we want to change Login portlet authentication failed message and properties for that message is : authentication-failed. We want to override this properties with our custom message.

blade create -t mvcportlet -p com.azilen.fragment.language -c CustomLanguageComponent language-fragment-module

Note here we are creating -t mvcportlet not -t fragment because here we are override language properties with Resource bundle class.

CustomLanguageComponent.Java

```
@Component(
		property = { "language.id=en_US" },
		service = ResourceBundle.class
)
public class CustomLanguageComponent extends ResourceBundle {

ResourceBundle bundle = ResourceBundle.getBundle("content.Language",
UTF8Control.INSTANCE);

	@Override
	protected Object handleGetObject(String key) {
		System.out.println("getting key"+key);
		return bundle.getObject(key);
	}

	@Override
	public Enumeration<String> getKeys() {
		return bundle.getKeys();
	}

}
```

It can be seen that, we are creating our custom class CustomLanguageComponent which extends ResourceBundle and we are specify property = { "language.id=en_US" } in @Component annotation. We are pointing to override Language_en.properties file.

Now Create Language_en.properties under /language-fragment-module/src/main/resources/content and add authentication-failed property with our custom messages.

authentication-failed=Authentication failed. Please try again(customized).

Now deploy language-fragment-module and you will get customized message when login failed in Login Module.

Override HTTP request with Servlet Filter

Sometimes, we need to intercept http request and need to write logic on that request. So we can achieve it with BaseFilter.

Here I will intercept every request and just print log in processFilter method.

blade create -t mvcportlet -p com.azilen.custom.filter -c CustomFilterPortlet custom-filter-fragment-module

CustomFilterPortlet.java

```
@Component(
                    immediate = true,
                    property = {
                   "dispatcher=REQUEST", "dispatcher=FORWARD",
                    "servlet-context-name=",
                    "servlet-filter-name=Custom Filter",
                    "url-pattern=/*"
                    },
                    service = Filter.class
)
public class CustomFilterPortlet extends BaseFilter {

        private static final Log _log = LogFactoryUtil.getLog(CustomFilterPortlet.class);

        @Override
         protected void processFilter(HttpServletRequest request, HttpServletResponse response,
FilterChain filterChain)
         throws Exception {

         _log.info("Intercept request successfully !!!");
         filterChain.doFilter(request, response);
    }

}
```

Here CustomFilterPortlet extends BaseFilter and override processFilter method where we can write our logic and we have to specify url pattern as url-pattern=/* in @Component property.

Override Liferay Struts actions

For Liferay 6.2 and other prior versions, all the Liferay actions are being handled by struts action defined in struts-config.xml. On the other hand, in Liferay 7 DXP most of the actions are converted in ActionCommand. Still Liferay 7 has some struts action defined in struts-config.xml. If we want to override that action then we can it override with StrutsAction.class.

Please make a note only struts actions defined in struts-config.xml file can only be overridden by StrutsAction.class.

Here I will override Terms and Conditions action which will be executed when the user logs in for the first time and he has to accept terms and conditions.

blade create -t mvcportlet -p com.azilen.custom.struts.action -c CustomTermsOfUseActionPortlet struts-action-fragment

CustomTermsOfUseActionPortlet.Java

```
@Component(
		immediate=true,
		property={
				"path=/portal/update_terms_of_use"
		},
		service = StrutsAction.class
)
public class CustomTermsOfUseActionPortlet extends BaseStrutsAction {

	private static final Log _log =
LogFactoryUtil.getLog(CustomTermsOfUseActionPortlet.class);
	@Override
	public String execute(StrutsAction originalStrutsAction, HttpServletRequest request,
HttpServletResponse response)
			throws Exception {

		_log.info("Calling Custom Termsof Use Action");

		//you logic goes here

		return originalStrutsAction.execute(request, response);
	}
}
```

Here CustomTermsOfUseActionPortlet extends BaseStrutsAction and override executes method where we can write our custom logic. We need to specify struts action path which we want to override in property “path=/portal/update_terms_of_use” in @Component.

Override Liferay Modal Listener

Sometimes, we need to override Liferay OOTB entity like User, Group, DLFileEntry etc for operation like onAfterUpdate or onBeforeUpdate. We can override those methods with BaseModelListener<T> class.

Here I want to override Liferay User’s onAfterUpdate method which will execute when any user will be updated.

```
blade create -t mvcportlet -p com.azilen.modal.listener.portlet -c
CustomUserModalListerPortlet custom-model-listener-module
```

```
@Component(
		immediate = true,
		service = ModelListener.class
```

```
		)
public class CustomUserModalListerPortlet extends  BaseModelListener<User> {

	@Override
	 public void onAfterUpdate(User model) throws ModelListenerException {
	 _log.info("user is updateing... !!!!");
	 super.onAfterUpdate(model);
	 }

	 private static final Log _log =
LogFactoryUtil.getLog(CustomUserModalListerPortlet.class);

}
```

here CustomUserModalListerPortlet extends BaseModelListener<User> and override onAfterUpdate method where we can write our custom logic. We need to define service property value as ModelListener.class in @Component.

UNITED STATES

Azilen Technologies LLC
6476, Orchard Lake Road,
Ste D, West Bloomfield,
MI 48325

Tel: +1-972-325-2243

BELGIUM

Azilen Technologies BVBA
Diestseweg 32C,
2440 Geel,
Belgium

Tel: +32 3 8082588

INDIA

Azilen Technologies Pvt. Ltd.
404/405,Iscon Mall,
Near Shivranjani Cross Road,
Satellite, Ahmedabad-380015

Tel: +91-79 4009 3121

Email : info@azilen.com | Website : www.azilen.com