Liferay 7 Portlet to Empower Your Custom Development

Though there are many implementations of MVC frameworks in Java, Liferay has created another lightweight one without any special configuration files offering some advance advantages. This section will cover:

Liferay MVC Portlet

Portlet Configuration Icon

Customize Portlet Configuration

Portlet Provider Template

Portlet Toolbar Contributor Template

1. Liferay MVC Portlet

1.1 MVC Action Command

Liferay's MVC framework lets you split your portlets action methods into separate classes. This can be very helpful in portlets that have many actions. Each action URL in your portlets JSPs then calls the appropriate action class when necessary.

```
	@Component(
			immediate = true,
			property = {
				"javax.portlet.name=
com_azilen_liferay_portlet_SampleMVCPortlet",

	"mvc.command.name=actionCommand1",
				"mvc.command.name=actionCommand2"
			},
			service = MVCActionCommand.class
		)
	public class DemoMVCActionCommand extends BaseMVCActionCommand   {
	@Override
protected void doProcessAction(ActionRequest actionRequest, ActionResponse
actionResponse) throws Exception {
		}
	}
```

1.2 MVC Render Command

Liferay's MVC framework lets you split your portlets render methods into separate classes. This can be very helpful in portlets that have many render methods. Each render URL in your portlets JSPs then calls the appropriate render class when necessary.

```
        @Component(
              immediate = true,
              property = {
"javax.portlet.name=     com_azilen_liferay_portlet_SampleMVCPortlet",
                    "mvc.command.name=renderCommand1",
                    "mvc.command.name=renderCommand2"
           },
           service = MVCRenderCommand.class
           )
      public class DemoRenderCommand implements MVCRenderCommand {

public String render(RenderRequest renderRequest, RenderResponse
renderResponse) throws PortletException {
}
```

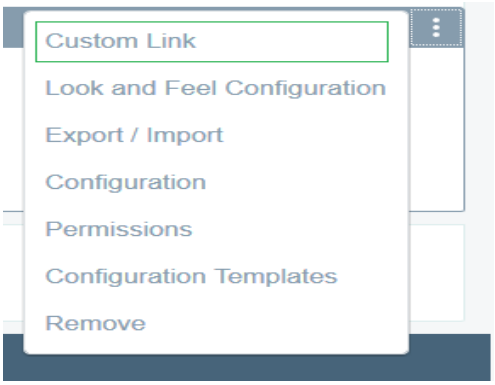
1.3 MVC Resource Command

Liferay's MVC framework lets you split your portlets resource methods into separate classes. This can be very helpful in portlets that have much resource methods. Each resource URL in your portlets JSPs then calls the appropriate render class when necessary.

```
      @Component(
                  immediate = true,
                  property = {
"javax.portlet.name= com_azilen_liferay_portlet_SampleMVCPortlet",
                        "mvc.command.name=ajax1",
            },
            service = MVCResourceCommand.class
            )
      public class DemoMVCResourceCommand extends BaseMVCResourceCommand{
                  @Override
protected void doServeResource(ResourceRequest resourceRequest,
ResourceResponse resourceResponse)  throws Exception {}
```

2. Portlet Configuration Icon

Portlet configuration icon is allowing user to add custom link on portlet toolbar menu.



To implement a configuration icon follow these steps

2.1 Create a Java class and extend **BasePortletConfigurationIcon** class.

```
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=com_azilen_liferay_toolbar_portlet_ToolBarPortlet"
        },
        service = PortletConfigurationIcon.class
)
public class SamplePortletConfigurationIcon  extends BasePortletConfigurationIcon {
}
```

2.2 Define custom link name in Language.properties file which will use by getMessage() method of BasePortletConfigurationIcon class.

```
@Override
public String getMessage(PortletRequest portletRequest) {
        ResourceBundle resourceBundle = ResourceBundleUtil.getBundle(
                "content.Language", getLocale(portletRequest), getClass());
        return LanguageUtil.get(resourceBundle, "sample-link");
}
```

2.3 override's following method's

public String getURL(PortletRequest req, PortletResponse res)

public double getWeight()

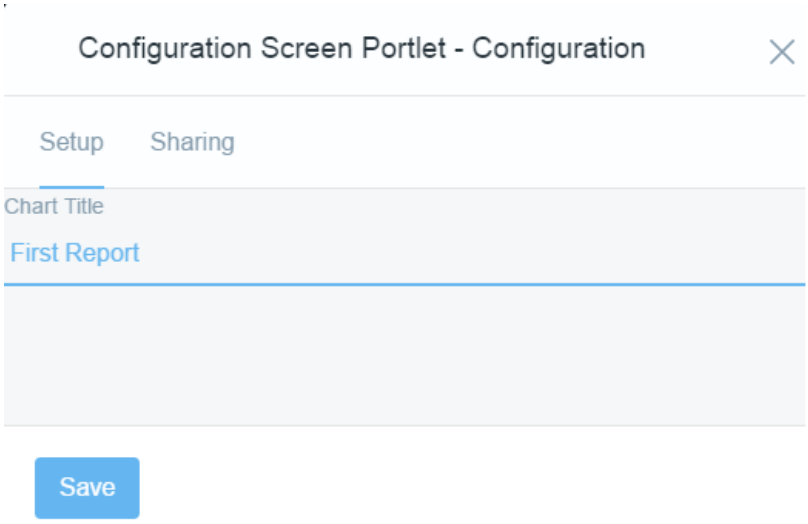
public boolean isShow(PortletRequest req)

public boolean isToolTip()

public boolean isUseDialog()

3. Customize Portlet Configuration

Portlet configuration is used for creating custom configuration for portlet and configuration page is display under portlet configuration setup tab.



<u>To implement a configuration action, follow these steps</u>

3.1 Create an interface to represent your configuration

```
@Meta.OCD(id = "com.azilen.liferay.config.ChartConfig")
public interface ChartConfig {

        public static final String FIELD_CHART_TITLE = "chartTitle";

    @Meta.AD(required = false)
     public String getChartTitle();
        }
```

3.2 Implement your configuration action class and add a reference to your configuration in your configuration action class

```
    @Component(
        configurationPid = "com.azilen.liferay.config.ChartConfig",
        configurationPolicy = ConfigurationPolicy.OPTIONAL,
                immediate = true,
                property =
{"javax.portlet.name=com_azilen_liferay_portlet_DemoConfigurationScreenPortlet"},
                service = ConfigurationAction.class
        )
        public class ChartConfigAction extends DefaultConfigurationAction  {

        }
```

3.3 Implement the user interface for configuring your application

3.4 -**metatype: *** in project bnd.bnd file (This line lets bnd use your configuration interface to generate an XML configuration file)

4. Portlet Provider Template

Portlet provider template is allowing user to provide mechanism to customize liferay in built entity behavior.

Make sure that this type of portlet is added hidden category of portlet.

If user wants to change behavior of blog portlet UI or any other existing entity behavior then we can user "*Portlet Providers* framework".

Once we deploy our custom module with specific entity which we want to customize and when we use this entity then liferay will pick the customize module which we have created instead of existing liferay portlet.

To implement a portlet provider template , follow these steps

4.1 Create a Java class and extend **BasePortletProvider** class implement the appropriate portlet provider interface based on the action type you chose your portlet to handle **AddPortletProvider,ViewPortletProvider, BrowsePortletProvider.**

```
    @Component(
        immediate = true,
        property = {
        "model.class.name=
        com.liferay.document.library.kernel.model.DLFileEntry",
               "service.ranking:Integer=" + Integer.MAX_VALUE
        },
        service = AddPortletProvider.class
)
public class ProviderTemplateAddPortletProvider
                  extends BasePortletProvider implements AddPortletProvider{
}
```

4.2 Naming conventions, name the class based on the element type and action type, followed by *PortletProvider.*

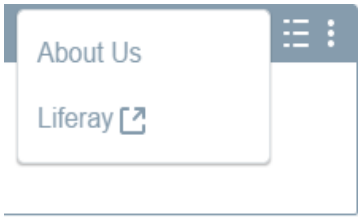
4.3 In some cases, a default portlet is already in place to handle the entity and action type requested. To override the default portlet with a custom portlet, you can assign your portlet a higher service ranking. You can do this by setting the following property in your **@Component** declaration

property= {"service.ranking:Integer=Integer.*MAX_VALUE*"}

4.4 Specify the methods you'd like to implement. Make sure to retrieve the portlet ID/portlet URL that should be provided when this service is called

5. Portlet Toolbar Contributor Template

Portlet Toolbar Contributor Template is allowing user to add extra toolbar menu in portlet.



To implement a portlet toolbar contributor template follow these steps

1. Create a Java class and implement **PortletToolbarContributor** interface

```
@Component(
                immediate = true,
                property = {
                        "javax.portlet.name=com_azilen_liferay_toolbar_portlet_ToolBarPortlet",
                        "mvc.path=-"
                },
                service = PortletToolbarContributor.class
)
public class ToolbarContributor implements PortletToolbarContributor {
@Override
public List<Menu> getPortletTitleMenus(}
                }
```


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