OpenCms Days 2008

Custom Widgets In OpenCms

Welcome!

Dan Liliedahl

OpenCms 7 Development

http://www.packtpub.com/opencms-7-development/book

Extending OpenCms

Developing a Custom Widget

[PACKT] PUBLISHING

Widgets in OpenCms

- Provides Rich User Interface
- Used in Structured Content Editors
- Can be used by declaring them in XSD schema files

  <xsd:annotations>\<layouts> section
- Forms are built automatically from XSD
- Can also be used from Java within CmsWidgetDialog derived classes

Standard Widgets

- OpenCms comes with over 20:

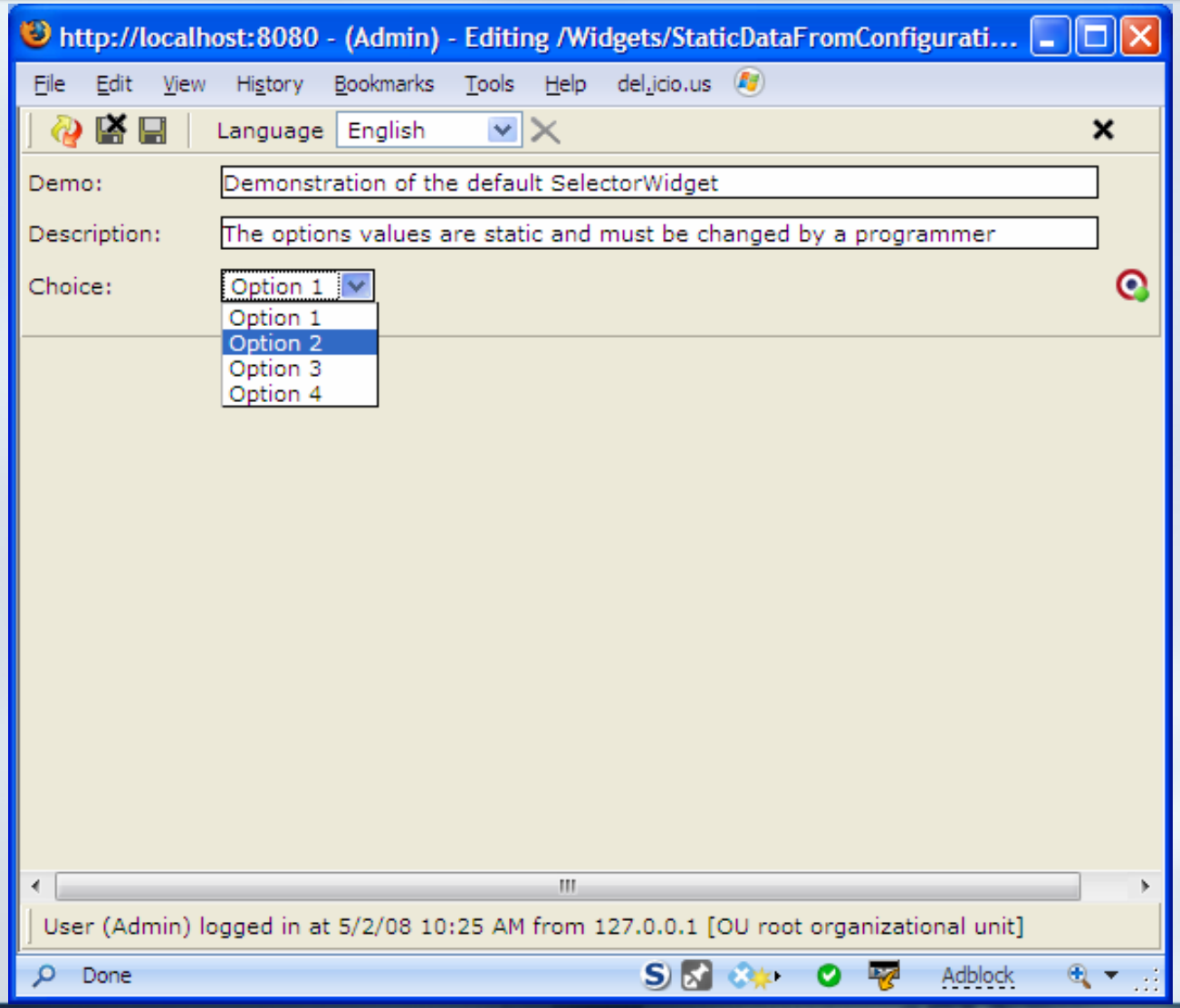
  DateTimeWidget, BooleanWidget, ColorpickerWidget, ComboWidget, DisplayWidget, DownloadGalleryWidget, GroupWidget, HtmlGalleryWidget, HtmlWidget, ImageGalleryWidget, StringWidget, StringWidgetPlaintext, LinkGalleryWidget, LocalizationWidget, MultiSelectWidget, SelectorWidget, TableGalleryWidget, TextareaWidget, TextareaWidgetPlaintext, UserWidget, VfsFileWidget
- See xmlcontentdemo/WidgetDemo
- Also check source in the org.opencms.widgets package

Example Widget Declaration

XSD Schema File Snippet:

```
<xsd:complexType name="OpenCmsWidgetDemo1">
    <xsd:sequence>
        <xsd:element name="Demo"        type="OpenCmsString" minOccurs="1" maxOccurs="1" />
        <xsd:element name="Description" type="OpenCmsString" minOccurs="1" maxOccurs="1" />
        <xsd:element name="Choice"      type="OpenCmsString" minOccurs="1" maxOccurs="25"/>
    </xsd:sequence>
    <xsd:attribute name="language" type="OpenCmsLocale" use="optional"/>
</xsd:complexType>

<xsd:annotation>
    <xsd:appinfo>
        <layouts>
        <!-- The choice fields are declared here -->
        <layout element="Choice" widget="SelectorWidget"
                configuration="Option 1|Option 2|Option 3|Option 4" />
        </layouts>
    </xsd:appinfo>
</xsd:annotation>
```

Example: SelectorWidget



Works Great, Easy to use

- Lots of UI widgets to choose from
- Easy to use inside XSD file
- Can be changed on the fly
- Limitation: may not always fit needs
  - Example: SelectorWidget
    - Data source is static
    - Changes must be made by technical person
  - Solution: Add a new Widget!

Steps to Create a Widget

- 1. Design your widget
- 2. Write widget code
  - Implement the I_CmsWidget class
- 3. Register the widget with OpenCms
  - Edit configuration file
- 4. Include widget in XSD file
- 5. Use the widget!

Step 1: Design the widget

- New Select List Widget
- Allow for dynamic list of choices
- Data sources:
  - 1. Fields within a content item
  - 2. List of OpenCms Content Types
  - 3. Database Query
  - 4. Future/Extensible

Step 1: Design Continued…

- Rather than create individual widgets, create one that has a pluggable data source
- Use configuration for controlling source
- Values separated with |
- String Format:

  configuration=source='data_source'|

  Config_parm1='some value'|

  Config_parm2='some value'

Step 2: Widget Interface

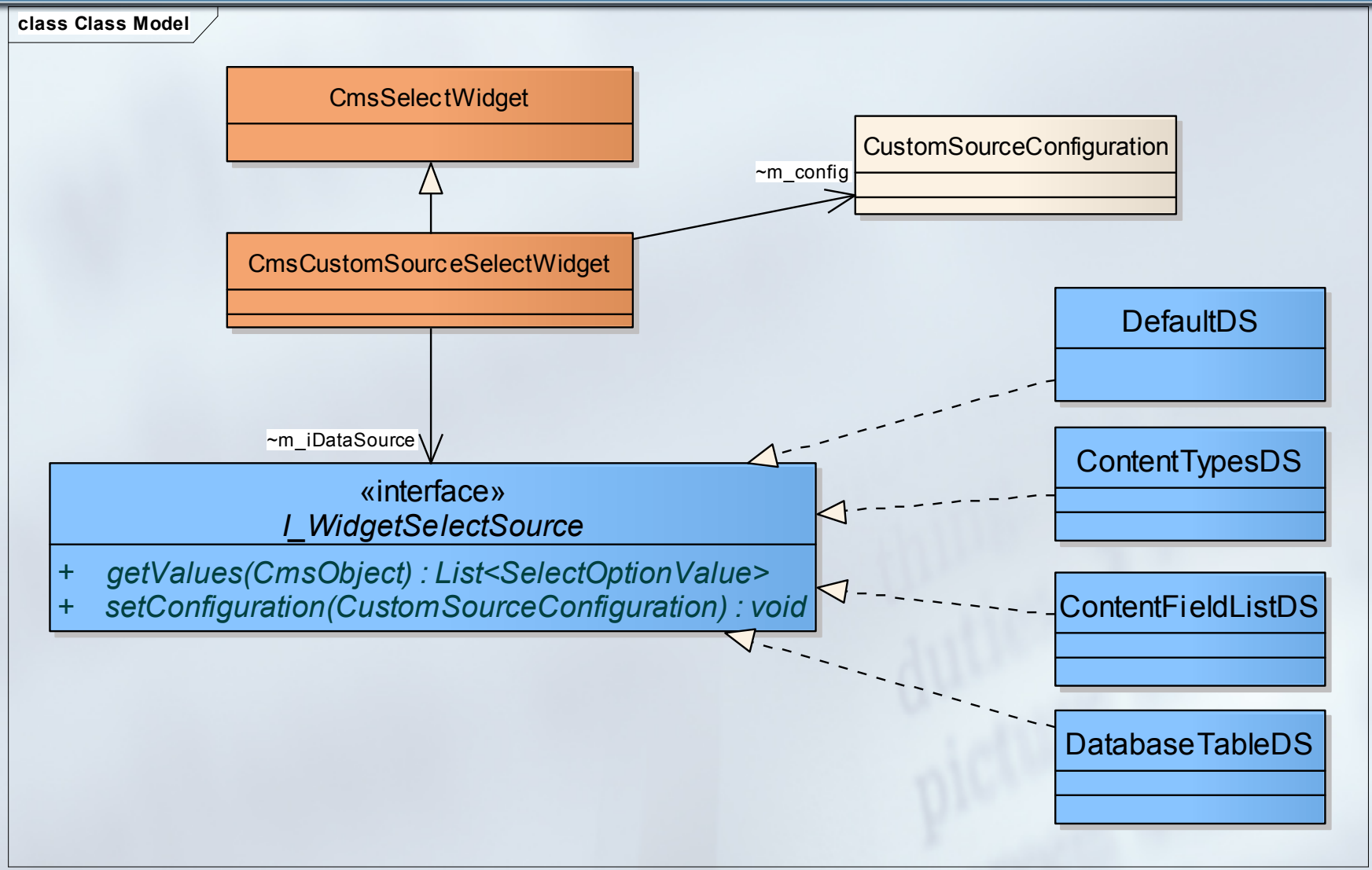
```
package: org.opencms.widgets

public interface I_CmsWidget {
    String getConfiguration();
    String getDialogHtmlEnd(…);
    String getDialogInitCall(…);
    String getDialogInitMethod(…);
    String getDialogWidget(…);
    String getHelpBubble(…);
    String getHelpText(…);
    String getWidgetStringValue(…);
    I_CmsWidget newInstance();
    void setConfiguration(…);
    void setEditorValue(…);
}
```

- Well documented in source code

Step2: Continued…

- Option1 : implement all methods
- Option 2: Subclass existing widget=Easy
  - We will subclass the CmsSelectWidget
  - Only need to override three methods:
    - newInstance – create an instance
    - setConfiguration – sets the widget configuration
    - getDialogWidget – builds widget HTML code
  - Our widget will delegate to another class that obtains the data values
  - The delegate class is pluggable

Step2: Widget implementation



Step2: Code – the Widget Class

```
public class CmsCustomSourceSelectWidget extends CmsSelectWidget {
    private static final Log LOG =
    CmsLog.getLog(CmsCustomSourceSelectWidget.class);

    /** The list of select values that will be returned */
     private List<SelectOptionValue> m_selectOptions = null;

     /** Contains the configuration options parsed from 'configuration' */
     CustomSourceConfiguration m_config;

     /** The widget data source */
     I_WidgetSelectSource m_iDataSource = null;

     /** Constructor */
     public CmsCustomSourceSelectWidget() {
         super();
     }
    /** Instantiates a new instance of the widget */
    public I_CmsWidget newInstance() {
        return new CmsCustomSourceSelectWidget();
    }
```

Step2:Code – setting the Config

```
public void setConfiguration(String configuration) {
    super.setConfiguration(configuration); // call superclass
    if (m_iDataSource == null) {
            // create the configuration
            m_config = new CustomSourceConfiguration(configuration);
            // read the class name for the data source and instantiate it
            String strClassName = m_config.getConfigValue("source");
            Class sourceClazz;
            try {
                sourceClazz = Class.forName(strClassName);
                m_iDataSource = (I_WidgetSelectSource) sourceClazz.newInstance();
            } catch (Exception e) {
                // Log the error
               LOG.error(Messages.get().getBundle().key(
               Messages.LOG_DATASOURCE_INIT_ERROR_2, strClassName), e);
               // since it failed use the default source provider to be nice
               m_iDataSource = new DefaultDS();
            }
            // give the configuration to the data source
            m_iDataSource.setConfiguration(m_config);
        }
}
```

Step2:Code – building the HTML

```
public String getDialogWidget(CmsObject cms, I_CmsWidgetDialog widgetDialog,
    I_CmsWidgetParameter param) {

    String id = param.getId();
    // build the SELECT HTML
    StringBuffer result = new StringBuffer(16);
    result.append("<td class=\"xmlTd\" style=\"height: 25px;\"><select class=\"xmlInput");
    if (param.hasError()) {
        result.append(" xmlInputError");
    }
    result.append("\" name=\"");
    result.append(id);
    result.append("\" id=\"");
    result.append(id);
    result.append("\">");

    // read the option data values - delegate to the data source
    getSelectOptionData(cms);

    ...
```

Step2:Code – building the HTML

```
    // finish the HTML
    if (null != m_selectOptions) {
        String selected = getSelectedValue(cms, param);
        Iterator<SelectOptionValue> i = m_selectOptions.iterator();
        while (i.hasNext()) {
            SelectOptionValue option = (SelectOptionValue) i.next();
            // create the option
            result.append("<option value=\"");
            result.append(option.getValue());
            result.append("\"");
            // retain SELECTED item
            if ((selected != null) && selected.equals(option.getValue())) {
                result.append(" selected=\"selected\"");
            }
            result.append(">");
            result.append(option.getName());
            result.append("</option>");
        }
    }
    result.append("</select>");
    result.append("</td>");
    return result.toString();  // return the HTML
}
```

Step 2: Obtaining select data

```
protected List<SelectOptionValue> getSelectOptionData(CmsObject cms) {
    // set the configuration in the data source
    m_iDataSource.setConfiguration(m_config);

    // read the option values
    // data values are not cached by default, but can be cached
    // by setting the configuration option "cacheData='true'"
    String strCache = m_config.getConfigValue("cacheData");
    if (null != strCache && strCache.equalsIgnoreCase("true"))  {
        if (m_selectOptions == null)  {
            m_selectOptions = m_iDataSource.getValues(cms);
        }
        return m_selectOptions;
    } else {
        // not caching, read the values again
        m_selectOptions = m_iDataSource.getValues(cms);
        return m_selectOptions;
    }
}
```

Step 3: Register with OpenCms

- Edit the opencms-vfs.xml file
- Located in:<OPENCMS>/WEB-INF/config/
- Find<widgets> section
- Add entry:

```
<widget
class="com.efoundry.widgets.CmsCustomSourceSelectWidget"
alias="DataSourceSelectWidget"/>
class = widget class name
alias = name used in XSD file
```

- Restart OpenCms!

Step 4: Include in XSD file

```
<xsd:complexType name="OpenCmsWidgetDemo1">
   <xsd:sequence>
      <xsd:element name="Demo"        type="OpenCmsString“ minOccurs="1" maxOccurs="1”/>
      <xsd:element name="Description" type="OpenCmsString“ minOccurs="1" maxOccurs="1" />
      <xsd:element name="Choice"      type="OpenCmsString“ minOccurs="1" maxOccurs="25”/>
    </xsd:sequence>
    <xsd:attribute name="language" type="OpenCmsLocale" use="optional"/>
</xsd:complexType>

<xsd:annotation>
    <xsd:appinfo>
       <layouts>
       <!-- The choice fields are declared here -->
       <layout element="Choice" widget="DataSourceSelectWidget“
          configuration="source='com.efoundry.widgets.sources.ContentFieldListDS'|
          contenttype='ChoiceList'|location='/Widgets/Seminars'|fieldname='Value'" />
          </layouts>
    </xsd:appinfo>
</xsd:annotation>
```

- Publish changes

Step 5: Use it

- Widget Sample 1

  Get list data from ChoiceList content type

```
<layout element="Choice“
  widget="DataSourceSelectWidget“
  configuration="source='com.efoundry.widgets.sources.Con
  tentFieldListDS'|contenttype='ChoiceList'|location='/Wi
  dgets/Seminars'|fieldname='Value'"  />
```

  Uses : ChoiceList content type

  Location and Field specified

  Result not cached

Step 5: Use it

- Widget Sample 2

  Get list of Content Types

```
<layout element="Choice“
  widget="DataSourceSelectWidget“
  configuration="source='com.efoundry.widgets.sources.Con
  tentFieldListDS‘|exclude=‘xmlpage’|cacheData=‘true’" />
```

  Uses : ChoiceList content type

  Excludes: xmlpage

  Caches result

Step 5: Use it

- Widget Sample 3

  Get data from DB query

```
<layout element="Choice“
  widget="DataSourceSelectWidget“
  configuration="source='com.efoundry.widgets.sources.Dat
  abaseTableDS'|jndiname='jdbc/states'|qry='select
  state,name from states‘|cacheData=‘true’"/>
```

  Uses JNDI resource: jdbc/states

  First 2 columns of returned query are used

  Cache result

Future Improvements

- Contextual/dependency based widgets not addressed
- AJAX - by jQuery?

Questions / Download

Thank you

Source Code Available in Book or Download:

http://code.google.com/p/opencmswidgets/