OpenKM
Extension guide

Contents

Articles

References

Extension Guide

OpenKM has a lot of interesting features, but it is not possible to meet all the special user requirements. For this reason, you can expand application features by implementing extensions. There are two families of extensions:

- Core extensions
- Frontend extensions
- Database Metadata
- Javascript API

Core extensions

This kind of extension enables you to add features at OpenKM Core level, implementing extension points. Actually there are a couple of methods which can be extended in this way but the list will grow as needed.

This kind of extensions are in development and only available in trunk. We expect to be included in a future OpenKM 5.2 release.

First of all, lets create an extension sample. Suppose you want to disable the uploading of certain type of documents in a folder. You need to create a Java project with your favorite IDE (or CLI), create a class which implements an interface and create a JAR archive. This generated archive should be placed in the **$JBOSS_HOME/plugins** directory.

That's all! The plugin will be used next time you restart JBoss. But more interesting. You can also refresh your plugins without restarting the application server. To do so, go to Administration > Scripting and execute this command:

```
com.openkm.extension.core.ExtensionManager.getInstance().reset();
```

Let's see how you can implement this plugin.

Plugin implementation

We will create a class called MyDocumentExtension, extending from DocumentExtension:

```
package com.openkm.sample.extension;

public class MyDocumentExtension implements DocumentExtension {
  @Override
  public int getOrder() {
    return 0;
  }

  @Override
  public void preCreate(Session session, Ref<Node> parentNode, Ref<File> content,
Ref<Document> doc) {
    // To be implemented
  }

  @Override
```

```
  public void postCreate(Session session, Ref<Node> parNode, Ref<Node> docNode, Ref<Document>
 doc) {
    // To be implemented
  }
}
```

These two methods represent two extension points in the document created action. The first one (**preCreate**) will be executed BEFORE the document is created. The second one (**postCreate**) will be executed AFTER the document creation. The **getOrder** method actually is not important but if you have many registered DocumentExtensions, it may be useful to force an execution order.

So, we want to reject a document creation when placed in a folder. Our objective is the **preCreate** method:

```
@Override
public void preCreate(Session session, Ref<Node> parentNode, Ref<File> content, Ref<Document>
 doc) throws
    AccessDeniedException, ExtensionException {
  try {
    if (parentNode.get().getPath().equals("/okm:root/forbidden") &&
doc.get().getMimeType().equals("application/pdf")) {
      throw new AccessDeniedException("Can't upload PDF documents
here");
    }
  } catch (RepositoryException e) {
    new ExtensionException(e.getMessage());
  }
}
```

The hard work has been done. Now, to declare this class as an extension you need to add the **@PluginImplementation** annotation. To use this annotation you need to include the library **jspf-1.0.1.jar** as a project dependency. You can download this library from http://code.google.com/p/jspf/.This is the class once all the steps have been finished:

```
package com.openkm.sample.extension;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class MyDocumentExtension implements DocumentExtension {
  @Override
  public int getOrder() {
    return 0;
  }

  public void preCreate(Session session, Ref<Node> parentNode, Ref<File> content,
Ref<Document> doc) throws
      AccessDeniedException, ExtensionException {
    try {
      if (parentNode.get().getPath().equals("/okm:root/forbidden") &&
doc.get().getMimeType().equals("application/pdf")) {
```

```
        throw new AccessDeniedException("Can't upload PDF documents
here");
      }
    } catch (RepositoryException e) {
      new ExtensionException(e.getMessage());
    }
  }

  @Override
  public void postCreate(Session session, Ref<Node> parNode, Ref<Node> docNode, Ref<Document>
 doc) {
  }
}
```

Frontend extensions

OpenKM plugin extensions makes it easy to extend the OpenKM end user interface by encapsulating plugin code and making it reusable between OpenKM versions.

The OpenKM plugin extension architecture is based on:

- Extensions
- Events
- Handlers
- Comunicators

Extensions are available widget definitions that allows the developer to make extensible panels and widgets (for example adding new tab panel on tab document)

Events are a collection of events that OpenKM UI fires each time any changes occur (for example when is added new keyword in tab document is fired event HasDocumentEvent.KEYWORD_ADDED)

Handlers are a collection of methods called internally by OpenKM. Handlers must be implemented into your extensions in order to collect OpenKM fired events. Each extension you make can have one or several handlers, that are automatically registered by OpenKM on loading process. OpenKM internally fire events to each declared handler.

Comunicators are a collection of methods avaliable as OpenKM Comunitators API to accessing transparently with some internal UI values. There are several Comunitators, for example with GeneralComunicator can access some general actions like refreshing UI as GeneralComunicator.refreshUI(). OpenKM Communicators API has read and write methods to interact with internal OpenKM UI objects.

Creating OpenKM plugin extensions is easy but you need some java knowledge and pay special attention to the Google Web ToolKit API that's used to build OpenKM UI. [1]

- HelloWorld Example
- Enable example extensions

Extension

- TabWorkspaceExtension
- TabDocumentExtension
- ToolBarBoxExtension
- TabFolderExtension
- TabMailExtension
- TabRecordExtension
- ToolBarButtonExtension
- MenuItemExtension
- MenuBarExtension

Handlers

- WorkspaceHandlerExtension
- NavigatorHandlerExtension
- DocumentHandlerExtension
- FolderHandlerExtension
- MailHandlerExtension
- RecordHandlerExtension
- ToolBarHandlerExtension
- PropertyGroupHandlerExtension
- DashboardHandlerExtension
- WidgetHandlerExtension

Comunicators

- GeneralComunicator
- WorkspaceComunicator
- NavigatorComunicator
- FileBrowserComunicator
- TabDocumentComunicator
- TabFolderComunicator
- TabMailComunicator
- TabRecordComunicator
- UtilComunicator
- DashboardComunicator
- SearchComunicator
- ToolBarComunicator

UI Constants

- UIMenuConstants
- UIDesktopConstants
- UIDockPanelConstants
- UISearchConstants
- UIFileUploadConstants

Misc

- RPCService
- Widget library

Best practices

- Adding RPC (Remote Process Calls) to server
- Internationalization

References

[1] http://code.google.com/webtoolkit/

HelloWorld Example

HelloWord example will add a new Widget in tab documents user interface.

Create a file called **HelloWorld.gwt.xml** into **src/main/resources/com/openkm/extension/frontend**

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE module SYSTEM "http://google-web-toolkit.googlecode.com/svn/releases/2.0/distro-source/core/src/gwt-module.dtd">

<module>

  <!-- Inherit the core Web Toolkit stuff -->

  <inherits name='com.google.gwt.user.User'/>

  <inherits name="com.google.gwt.http.HTTP"/>

</module>
```

Edit **Customization.gwt.xml** into **src/main/resources/com/openkm/extension/frontend** and add the new module

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE module SYSTEM "http://google-web-toolkit.googlecode.com/svn/releases/2.0/distro-source/core/src/gwt-module.dtd">

<module>

  <!-- Inherit the extension widgets -->

  <inherits name="com.openkm.extension.frontend.HelloWorld"/>

</module>
```

Create a file called **HelloWorld.java** into **src/main/java/com/openkm/extension/frontend/client**

```
public class HelloWorld extends TabDocumentExtension {
	Button refresh;
	VerticalPanel vPanel;

	public HelloWorld() {
```

```
            HTML html = new HTML("Hello Word");
            refresh = new Button("refresh UI");
            refresh.addClickHandler(new ClickHandler() {
                  @Override
                  public void onClick(ClickEvent event) {
                        GeneralComunicator.refreshUI();
                  }
            });
            vPanel = new VerticalPanel();
            vPanel.add(html);
            vPanel.add(refresh);

            refresh.setStyleName("okm-Input");

            initWidget(vPanel);
      }

      @Override
      public String getTabText() {
            return "Hello tab";
      }

      @Override
      public void set(GWTDocument doc) {
            // TODO Auto-generated method stub
      }

      @Override
      public void setVisibleButtons(boolean visible) {
            // TODO Auto-generated method stub

      }

        @Override
        public String getExtensionUUID() {
                return
String.valueOf("d9dab640-d098-11df-bd3b-0800200c9a66");
        }
}
```

Edit Customization.java OpenKM class and add the HelloWorld widget

```
public class Customization {

      /**
       * getExtensionWidgets
       *
       * @return
```

```
        */
      public static List<Object> getExtensionWidgets() {
            List<Object> extensions = new ArrayList<Object>();

            // Declare here your widget extensions
            extensions.add(new HelloWorld());

            return extensions;
      }
}
```

Now you only need to recompile project.

Enable example extensions

In order enable OpenKM extension go to **Administration** tab and click on **Database query** button. Then select JDBC and register the extensions in your DBMS.

ToolBarButtonExample

```
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME)
VALUES('9f84b330-d096-11df-bd3b-0800200c9a66','Toolbar button
example');
```

TabWorkspaceExample

```
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME)
VALUES('44f94470-d097-11df-bd3b-0800200c9a66','Tab workspace example');
```

TabFolderExample

```
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME)
VALUES('d95e01a0-d097-11df-bd3b-0800200c9a66','Tab folder example');
```

HelloWorld

```
INSERT INTO OKM_EXTENSION (EXT_UUID,EXT_NAME)
VALUES('d9dab640-d098-11df-bd3b-0800200c9a66','Hello world example');
```

After that, you also need to enable the proper extensions in **Administration > Profiles**.

TabWorkspaceExtension

Methods

getTabText

Used by OpenKM to get the tab text.

getExtensionUUID()

Return the unique extension id

Example

```
public class TabWorkspaceExample extends TabWorkspaceExtension {
	private VerticalPanel vPanel;

	/**
	 * TabWorkspaceExample
	 */
	public TabWorkspaceExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("new workspace example"));

		vPanel.setStyleName("okm-Input");

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "tab workspace";
	}

	  @Override
	public String getExtensionUUID() {
		return
String.valueOf("44f94470-d097-11df-bd3b-0800200c9a66");
	}
}
```

TabDocumentExtension

Methods

getTabText

Used by OpenKM to get the tab text.

Example

```
public class TabDocumentExample extends TabDocumentExtension {
	VerticalPanel vPanel;
	  String tabText = "Tab example";

	public TabDocumentExample() {
		HTML html = new HTML("Content tab example");
		vPanel = new VerticalPanel();
		vPanel.add(html);

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return tabText;
	}

}
```

ToolBarBoxExtension

Methods

getWidget()

Returns the associated widget to the tool box that will be showed when it'll be selected.

getExtensionUUID()

Return unique extension id

Example

```
public class ToolBarBoxExample extends ToolBarBoxExtension {
      public ToolBarBoxEx(Image img, String text) {
            super(img, text);
      }

      @Override
      public Widget getWidget() {
            // TODO Auto-generated method stub
            return null;
      }

            @Override
        public String getExtensionUUID() {
                        return
String.valueOf("d9dab640-d098-11df-bd3b-0800200c9a66");
        }
}
```

TabFolderExtension

Methods

getTabText

Used by OpenKM to get the tab text.

set(GWTFolder folder)

Any time there's some folder selected (in tree or browser) is executed this method by OpenKM in order to refreshing folder tab panel information.

setVisibleButtons(visible)

Show or hide buttons (for example in trash view normally edit button should not be visible).

Example

```
public class TabFolderExample extends TabFolderExtension {
	VerticalPanel vPanel;

	public TabFolderExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("hello world"));

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "New folder tab";
	}

	@Override
	public void set(GWTFolder folder) {
		// TODO Auto-generated method stub

	}

	@Override
	public void setVisibleButtons(boolean visible) {
		// TODO Auto-generated method stub
	}

}
```

TabMailExtension

Methods

getTabText

Used by OpenKM to get the tab text.

set(GWTMail mail)

Any time there's some mail selected (in browser) is executed this method by OpenKM in order to refreshing mail tab panel information.

setVisibleButtons(visible)

Show or hide buttons (for example in trash view normally edit button should not be visible).

Example

```
public class TabMailExample extends TabMailExtension {
	VerticalPanel vPanel;

	public TabMailExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("hello world"));

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "New folder tab";
	}

	@Override
	public void set(GWTMail mail) {
		// TODO Auto-generated method stub
	}

	@Override
	public void setVisibleButtons(boolean visible) {
		// TODO Auto-generated method stub
	}

}
```

TabRecordExtension

Methods

getTabText

Used by OpenKM to get the tab text.

set(GWTRecord record)

Any time there's some record selected (in browser) is executed this method by OpenKM in order to refreshing record tab panel information.

setVisibleButtons(visible)

Show or hide buttons (for example in trash view normally edit button should not be visible).

Example

```
public class TabRecordExample extends TabRecordExtension {
	VerticalPanel vPanel;

	public TabRecordExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("hello world"));

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "New folder tab";
	}

	@Override
	public void set(GWTRecord record) {
		// TODO Auto-generated method stub
	}

	@Override
	public void setVisibleButtons(boolean visible) {
		// TODO Auto-generated method stub
	}
}
```

ToolBarButtonExtension

Methods

checkPermissions(GWTFolder folder, GWTFolder folderParent, int originPanel)

Check button permisions depending on folder grants in order to evaluate enable or disabling button

checkPermissions(GWTDocument doc, GWTFolder folder)

Check button permisions depending on document grants in order to evaluate enable or disabling button

checkPermissions(GWTMail mail, GWTFolder folder)

Check button permisions depending on mail grants in order to evaluate enable or disabling button

isEnabled()

Return boolean value indicating if button is enabled or disabled

enable(boolean enable)

Enables or disables button

getExtensionUUID()

Return unique extension id

Example

```
public class ToolBarButton extends ToolBarButtonExtension {

	public ToolBarButton(Image image, String title, ClickHandler
handler) {
		super(image, title, handler);
	}

	@Override
	public void checkPermissions(GWTFolder folder, GWTFolder
folderParent, int originPanel) {
		// TODO Auto-generated method stub
	}

	@Override
	public void checkPermissions(GWTDocument doc, GWTFolder folder) {
		// TODO Auto-generated method stub
	}

	@Override
	public void checkPermissions(GWTMail mail, GWTFolder folder) {
		// TODO Auto-generated method stub
```

```
	}

	@Override
	public void enable(boolean enable) {
		// TODO Auto-generated method stub
	}

	@Override
	public boolean isEnabled() {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public String getExtensionUUID() {
		return "9f84b330-d096-11df-bd3b-0800200c9a66";
	}
}
```

MenuItemExtension

Constructors

MenuItemExtension(String imageURL, String text, Command cmd)

Normal menuitem

MenuItemExtension(String imageURL, String text, MenuBar menuBar)

A menuitem that has a menu bar. Normally used to submenu entry.

MenuItemExtension(String text, MenuBar menuBar)

Normally used to define the main menuItem. Menu root that has other menu options as child.

Example

```
public class MainMenuExample {
	private MenuItemExtension exampleMenu;
	private MenuBarExtension subMenuExample;
	private MenuBarExtension subMenuExample2;
	private MenuItemExtension subMenuItem;
	private MenuItemExtension option1;
	private MenuItemExtension option2;
	private MenuItemExtension option3;
	private MenuItemExtension option4;

	/**
```

```
        * MainMenuExample
        */
       public MainMenuExample() {
             // All menu items
             option1 = new MenuItemExtension("img/box.png", "Option 1",
option1Action);
             option2 = new MenuItemExtension("img/box.png", "Option 2",
option2Action);
             option3 = new MenuItemExtension("img/box.png", "Option 3",
option3Action);
             option4 = new MenuItemExtension("img/box.png", "Option 4",
option4Action);

             // Secondary submenu
             subMenuExample2 = new MenuBarExtension();
             subMenuExample2.addItem(option3);
             subMenuExample2.addItem(option4);
             subMenuItem = new MenuItemExtension("img/box.png","Sub
menu", subMenuExample2); // is a secondary submenu

             // Principal submenu
             subMenuExample = new MenuBarExtension();
             subMenuExample.addItem(option1);
             subMenuExample.addItem(option2);
             subMenuExample.addItem(subMenuItem);

             // Principal menuitem
             exampleMenu = new MenuItemExtension("New Menu",
subMenuExample); // is not a secondary submenu
       }

       public MenuItemExtension getNewMenu() {
             return exampleMenu;
       }

       /**
        * option1Action
        */
       Command option1Action = new Command() {
             public void execute() {
                   Window.alert("option1 action");
             }
       };

       /**
        * option2Action
        */
```

```
        Command option2Action = new Command() {
                public void execute() {
                        Window.alert("option2 action");
                }
        };

        /**
         * option3Action
         */
        Command option3Action = new Command() {
                public void execute() {
                        Window.alert("option3 action");
                }
        };

        /**
         * option4Action
         */
        Command option4Action = new Command() {
                public void execute() {
                        Window.alert("option4 action");
                }
        };
}
```

MenuBarExtension

Constructor

MenuBarExtension()

Used to create submenu extension.

Example

Complete example at MenuItemExtension.

WorkspaceHandlerExtension

Any extension that implements WorkspaceHandlerExtension will be watching all events fired from workspace tab panel.

Method

onChange(WorkspaceEventConstant event)

Each time it'll be some new tab panel event the method onChange it'll be fired by OpenKM

Example

```
public class HandlersTest implements WorkspaceHandlerExtension {
	@Override
	public void onChange(WorkspaceEventConstant event) {
		Window.alert("workspace event fired");
	}
}
```

NavigatorHandlerExtension

Any extension that implements NavigatorHandlerExtension will be watching all events fired from navigator stack panel

Method

onChange(NavigatorEventConstant event)

Each time it'll be some new stack panel event the method onChange it'll be fired by OpenKM.

Example

```
public class HandlersTest implements NavigatorHandlerExtension {
	@Override
	public void onChange(NavigatorEventConstant event) {
		Window.alert("navigator event fired");
	}
}
```

DocumentHandlerExtension

Any extension that implements DocumentHandlerExtension will be watching all events fired from document tab.

Method

onChange(DocumentEventConstant event)

Each time it'll be some new document event the method onChange it'll be fired by OpenKM

Example

```
public class TabDocumentExample extends TabDocumentExtension implements
 DocumentHandlerExtension {
	VerticalPanel vPanel;
	  String tabText = "Tab - example";

	public TabDocumentExample() {
		HTML html = new HTML("Content tab example");
		vPanel = new VerticalPanel();
		vPanel.add(html);

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
```

```
            return tabText;
      }

      @Override
      public void set(GWTDocument doc) {
            // TODO Auto-generated method stub
      }

      @Override
      public void setVisibleButtons(boolean visible) {
            // TODO Auto-generated method stub

      }

        @Override
      public void onChange(DocumentEventConstant event) {
            if (event.equals(HasDocumentEvent.DOCUMENT_CHANGED)) {
                  Window.alert("document changed - " +event.getType());
            } else if (event.equals(HasDocumentEvent.KEYWORD_ADDED)) {
                  Window.alert("keyword added - " +event.getType());
            } else if (event.equals(HasDocumentEvent.KEYWORD_REMOVED))
{
                  Window.alert("keyword removed - " +event.getType());
            } else if (event.equals(HasDocumentEvent.CATEGORY_ADDED)) {
                  Window.alert("category added - " +event.getType());
            } else if (event.equals(HasDocumentEvent.CATEGORY_REMOVED))
 {
                  Window.alert("category removed - " +event.getType());
            } else if (event.equals(HasDocumentEvent.TAB_CHANGED)) {
                  Window.alert("tab changed - " +event.getType() + " -
actual tab " + TabDocumentComunicator.getSelectedTab());
            } else if (event.equals(HasDocumentEvent.PANEL_RESIZED)) {
                  Window.alert("panel resized - " +event.getType());
            } else if (event.equals(HasDocumentEvent.SECURITY_CHANGED))
 {
                  Window.alert("security changed - " +event.getType());
            } else if (event.equals(HasDocumentEvent.NOTE_ADDED)) {
                  Window.alert("note added - " +event.getType());
            }
      }
}
```

FolderHandlerExtension

Any extension that implements FolderHandlerExtension will be watching all events fired from folder tab

Method

onChange(FolderEventConstant event)

Each time it'll be some new folder event the method onChange it'll be fired by OpenKM

Example

```
public class TabFolderExample extends TabFolderExtension implements
FolderHandlerExtension {
	VerticalPanel vPanel;

	public TabFolderExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("hello world"));

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "New folder tab";
	}

	@Override
	public void set(GWTFolder doc) {
		// TODO Auto-generated method stub
	}

	@Override
	public void setVisibleButtons(boolean visible) {
		// TODO Auto-generated method stub
	}

	@Override
	public void onChange(FolderEventConstant event) {
		if (event.equals(HasFolderEvent.TAB_CHANGED)) {
			Window.alert("tab changed - " + event.getType());
		}
	}
}
```

MailHandlerExtension

Any extension that implements MailHandlerExtension will be watching all events fired from mail tab

Method

onChange(MailEventConstant event)

Each time it'll be some new mail event the method onChange it'll be fired by OpenKM

Example

```
public class TabMailExample extends TabMailExtension implements
MailHandlerExtension {
	VerticalPanel vPanel;

	public TabMailExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("hello world"));

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "New folder tab";
	}

	@Override
	public void set(GWTMail mail) {
		// TODO Auto-generated method stub
	}

	@Override
	public void setVisibleButtons(boolean visible) {
		// TODO Auto-generated method stub
	}

	@Override
	public void onChange(MailEventConstant event) {
		if (event.equals(HasMailEvent.TAB_CHANGED)) {
			Window.alert("tab changed - " + event.getType());
		}
	}
}
```

RecordHandlerExtension

Any extension that implements RecordHandlerExtension will be watching all events fired from record tab

Method

onChange(RecordEventConstant event)

Each time it'll be some new Record event the method onChange it'll be fired by OpenKM

Example

```
public class TabRecordExample extends TabRecordExtension implements
RecordHandlerExtension {
	VerticalPanel vPanel;

	public TabRecordExample() {
		vPanel = new VerticalPanel();
		vPanel.add(new HTML("hello world"));

		initWidget(vPanel);
	}

	@Override
	public String getTabText() {
		return "New Record tab";
	}

	@Override
	public void set(GWTRecord rec) {
		// TODO Auto-generated method stub
	}

	@Override
	public void setVisibleButtons(boolean visible) {
		// TODO Auto-generated method stub
	}

	@Override
	public void onChange(RecordEventConstant event) {
		if (event.equals(HasRecordEvent.TAB_CHANGED)) {
			Window.alert("tab changed - " + event.getType());
		}
	}
}
```

ToolBarHandlerExtension

Any extension that implements ToolBarHandlerExtension will be watching all events fired from toolbar

Method

onChange(ToolBarEventConstant event)

Each time it'll be some new toolbar event the method onChange it'll be fired by OpenKM

Example

```
public class ToolBarButton extends ToolBarButtonExtension implements
ToolBarHandlerExtension {

	public ToolBarButton(Image image, String title, ClickHandler
handler) {
		super(image, title, handler);
	}

	@Override
	public void checkPermissions(GWTFolder folder, GWTFolder
folderParent, int originPanel) {
		// TODO Auto-generated method stub
	}

	@Override
	public void checkPermissions(GWTDocument doc, GWTFolder folder) {
		// TODO Auto-generated method stub
	}

	@Override
	public void checkPermissions(GWTMail mail, GWTFolder folder) {
		// TODO Auto-generated method stub
	}

	@Override
	public void enable(boolean enable) {
		// TODO Auto-generated method stub
	}

	@Override
	public boolean isEnabled() {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
```

```
	public void onChange(ToolBarEventConstant event) {
		if (event.equals(HasToolBarEvent.EXECUTE_ADD_DOCUMENT)) {
			Window.alert("executed add document - " +
event.getType());
		}
	}
}
```

PropertyGroupHandlerExtension

Any extension that implements PropertyGroupHandlerExtension will be watching all events fired from property group (metadata) tab.

Method

onChange(PropertyGroupEventConstant event)

Each time it'll be some new property group (metadata) event the method onChange it'll be fired by OpenKM

Example

```
public class Example implements PropertyHandlerExtension {

	public Example() {
	}

	  @Override
	public void onChange(PropertyEventConstant event) {
		if
(event.equals(HasPropertyGroupEvent.EVENT_PROPERTYGROUP_CHANGED)) {
			Window.alert("propertyGroup changed - "
+event.getType());
		} else {
				Window.alert("other property group event - "
+event.getType());
			}
	}
}
```

DashboardHandlerExtension

Any extension that implements DashboardHandlerExtension will be watching all events fired from folder tab

Method

onChange(DashboardEventConstant event)

Each time it'll be some new dashboard event the method onChange it'll be fired by OpenKM

Example

```
public class Example implements DashboardHandlerExtension {

	public Example() {
	}

	@Override
	public void onChange(DashboardEventConstant event) {
		if (event.equals(HasDashboardEvent.TOOLBOX_CHANGED)) {
			// Do something here
		}
	}
}
```

WidgetHandlerExtension

Any extension that implements WidgetHandlerExtension will be watching all events fired from widgets

Method

onChange(WidgetEventConstant event)

Each time it'll be some new widget event the method onChange it'll be fired by OpenKM

Example

```
public class Example implements WidgetHandlerExtension {

	public Example() {
	}

	@Override
	public void onChange(WidgetEventConstant event) {
		if (event.equals(HasWidgetEvent.FINISH_STARTUP)) {
			// Do something here
		}
	}
}
```

GeneralComunicator

Read Methods

- refreshUI()
- ToolBarOption getToolBarOption()
- String getLang()
- downloadDocument(boolean checkout)
- downloadDocumentPdf()
- downloadFile(String path, String params)
- extensionCallOwnDownload(String url)
- resetStatus()
- showError(String callback, Throwable caught)
- logout()
- refreshUserDocumentsSize()
- List<String> getUserRoleList()
- String getUser()
- String i18nExtension(String property)
- openPath(String path, String docPath)
- String getAppContext()
- showNextWizard()
- boolean isDigitalSignature()
- GWTDocument getDocumentToSign()
- String getToken()
- GWTWorkspace getWorkspace()
- String i18n(String property)
- enableKeyShorcuts()
- disableKeyShorcuts()
- openPathByUuid(String uuid)
- Main get()

getToolBarOption() must be only used for reading purposes

Write methods

- setStatus(String msg)

Deprecated

- openAllFolderPath(String path, String docPath)

Example

```
// Call to refreshing user interface
GeneralComunicator.refreshUI();
```

WorkspaceComunicator

Read Methods

- int getSelectedTab()
- changeSelectedTab(int selectedTab)
- int getSelectedWorkspace()
- GWTWorkspace getWorkspace()
- getTabExtensionIndex(TabWorkspaceExtension widget)
- String getAppVersion()

Example

```
int tab = WorkspaceComunicator.getSelectedTab();
if (tab==0) {
  Window.alert("First tab enabled");
} else {
  Window.alert("Other stack");
}
```

NavigatorComunicator

Read Methods

- int getStackIndex()
- boolean isTaxonomyShown()
- boolean isCategoriesShown()
- boolean isMetadataShown()
- boolean isMetadataShown
- boolean isThesaurusShown()
- boolean isTemplatesShown()
- boolean isPersonalShown()
- boolean isMailShown()
- boolean isTrashShown()
- GWTFolder getFolder()
- String getActualPath()
- GWTFolder getRootFolder()
- GWTFolder getCategoriesRootFolder()
- GWTFolder getMetadataRootFolder()
- GWTFolder getThesaurusRootFolder()
- GWTFolder getTemplatesRootFolder()
- GWTFolder getPersonalRootFolder()
- GWTFolder getMailRootFolder()
- GWTFolder getTrashRootFolder()

Example

```
int stack = NavigatorComunicator.getStackIndex();
if (stack==0) {
  Window.alert("First stack enabled");
} else {
  Window.alert("Other stack");
}
```

FileBrowserComunicator

Read Methods

- boolean isDocumentSelected()
- boolean isFolderSelected()
- boolean isMailSelected()
- GWTDocument getDocument()
- GWTFolder getFolder()
- GWTMail getMail()
- boolean isPanelSelected()
- refreshOnlyFileBrowser();

Example

```
if (FileBrowserComunicator.isDocumentSelected()) {
    Window.alert("document selected");
} else {
    Window.alert("document not selected");
}
```

TabDocumentComunicator

Read Methods

- int getSelectedTab()
- GWTDocument getDocument()
- Collection<String> getKeywords()
- Collection<GWTNote> getNotes()
- boolean isVisibleButton()
- refreshPreviewDocument()
- boolean isWidgetExtensionVisible(Widget widget)

Write Methods

- addKeyword(String keyword)
- removeKeyword(String keyword)
- addCategory(GWTFolder category)
- removeCategory(String UUID)
- setRefreshingStyle()
- unsetRefreshingStyle()

Example

```
int selectedTab = TabDocumentComunicator.getSelectedTab();
if (selectedTab==0) {
  Window.alert("First tab enabled");
} else {
  Window.alert("Other tab");
}
```

TabFolderComunicator

Read Methods

- int getSelectedTab()
- GWTFolder getFolder()
- boolean isVisibleButton()
- boolean isWidgetExtensionVisible(Widget widget)

Example

```
int selectedTab = TabFolderComunicator.getSelectedTab();
if (selectedTab==0) {
  Window.alert("First tab enabled");
} else {
  Window.alert("Other tab");
}
```

TabMailComunicator

Read Methods

- int getSelectedTab()
- GWTMail getMail()
- boolean isWidgetExtensionVisible(Widget widget)

Example

```
int selectedTab = TabMailComunicator.getSelectedTab();
if (selectedTab==0) {
  Window.alert("First tab enabled");
} else {
  Window.alert("Other tab");
}
```

TabRecordComunicator

Read Methods

- int getSelectedTab()
- GWTRecord getRecord()
- boolean isWidgetExtensionVisible(Widget widget)

Example

```
int selectedTab = TabRecordComunicator.getSelectedTab();
if (selectedTab==0) {
  Window.alert("First tab enabled");
} else {
  Window.alert("Other tab");
}
```

UtilComunicator

Read Methods

- String formatSize(double size)
- String imageItemHTML(String imageUrl, String title, String align)
- String createHeaderHTML(String imageURL, String caption)
- String menuHTML(String imageUrl, String text)
- String imageItemHTML(String imageUrl)
- String getTextAsBoldHTML(String text, boolean mark)
- String getUserAgent()
- String getParent(String path)
- String getName(String path)
- String mimeImageHTML(String mime)

Example

```
// Call to formating some document size number ( gb, mb etc... )
UtilComunicator.formatSize(15);
```

DashboardComunicator

Read Methods

- getUserSubscribedDocuments()
- getUserSubscribedFolders()
- refreshAllSearchs()
- showToolBoxExtension(ToolBarBoxExtension extension)
- getUserSearchs(boolean refresh)

Example

```
// Call to refreshing subscribed documents
DashboardComunicator.getUserSubscribedDocuments();
```

SearchComunicator

Read Methods

- getAllSearchs()
- getUserSearchs()
- int getSelectedRowSearchSaved()
- int getSelectedRowUserNews()
- GWTQueryParams getSavedSearch()
- GWTQueryParams getSavedUserNews()

Write Methods

- setSavedSearch(GWTQueryParams params)

Example

```
// Call to refreshing subscribed documents
GWTQueryParams params = new GWTQueryParams();
params.setName("test");
SearchComunicator.setSavedSearch(params);
```

ToolBarComunicator

Read Methods

- evaluateShowIcons()
- ToolBarOption getToolBarOption()
- Object getActualNode()
- HorizontalPanel getMainToolBarPanel()

Write Methods

- setToolBarOption(ToolBarOption toolBarOption)

Example

```
ToolBarComunicator.evaluateShowIcons();
```

UIMenuConstants

UIMenuConstant class defines contants to be used in OpenKM extensions to identify some menus where add own menu extensions.

If menu location is not defined in extension, by default is set value **NEW_MENU** location. That means it'll be created as new menu into main menu desktop view.

Example

How adding new submenu in default tools menu:

```
public class SubMenuMessage {
	private MenuItemExtension messageMenu;
	private MenuBarExtension subMenuMessage;
	private MenuItemExtension sendNewMessage;

	/**
	 * SubMenuMessage
	 */
	public SubMenuMessage() {
		// All menu items
		sendNewMessage = new
MenuItemExtension("img/icon/actions/new_message.png", "New message",
sendMessage);

		// Principal submenu
		subMenuMessage = new MenuBarExtension();
		subMenuMessage.addItem(sendNewMessage);
		messageMenu = new
MenuItemExtension("img/icon/actions/message.png", "Message",
```

```
subMenuMessage);

messageMenu.setMenuLocation(UIMenuConstants.MAIN_MENU_TOOLS);
      }

         /**
       * @return
       */
      public MenuItemExtension getMenu() {
            return messageMenu;
      }

      /**
       * option1Action
       */
      Command sendMessage = new Command() {
            public void execute() {
                  Window.alert("some action");
            }
      };
}
```

UIDesktopConstants

UIDesktopConstants class defines contants to be used in OpenKM extensions to identify some desktop widgets.

Example

```
if
(NavigatorComunicator.getStackIndex()==UIDesktopConstants.NAVIGATOR_TAXONOMY)
 {
    Window.alert("Taxonomy selected");
} else {
    Window.alert("Other navigator panel selected");
}
```

UIDockPanelConstants

UIDockPanelConstants class defines contants to be used in OpenKM extensions to identify some dock panel widgets (main widgets).

Example

```
WorkspaceComunicator.changeSelectedTab(UIDockPanelConstants.SEARCH);
```

UISearchConstants

UISearchConstants class defines contants to be used in OpenKM extensions to identify some widgets on search view.

UIFileUploadConstants

UIFileUploadConstants class defines contants to be used in OpenKM extensions to identify some operation on fileupload.

RPCService

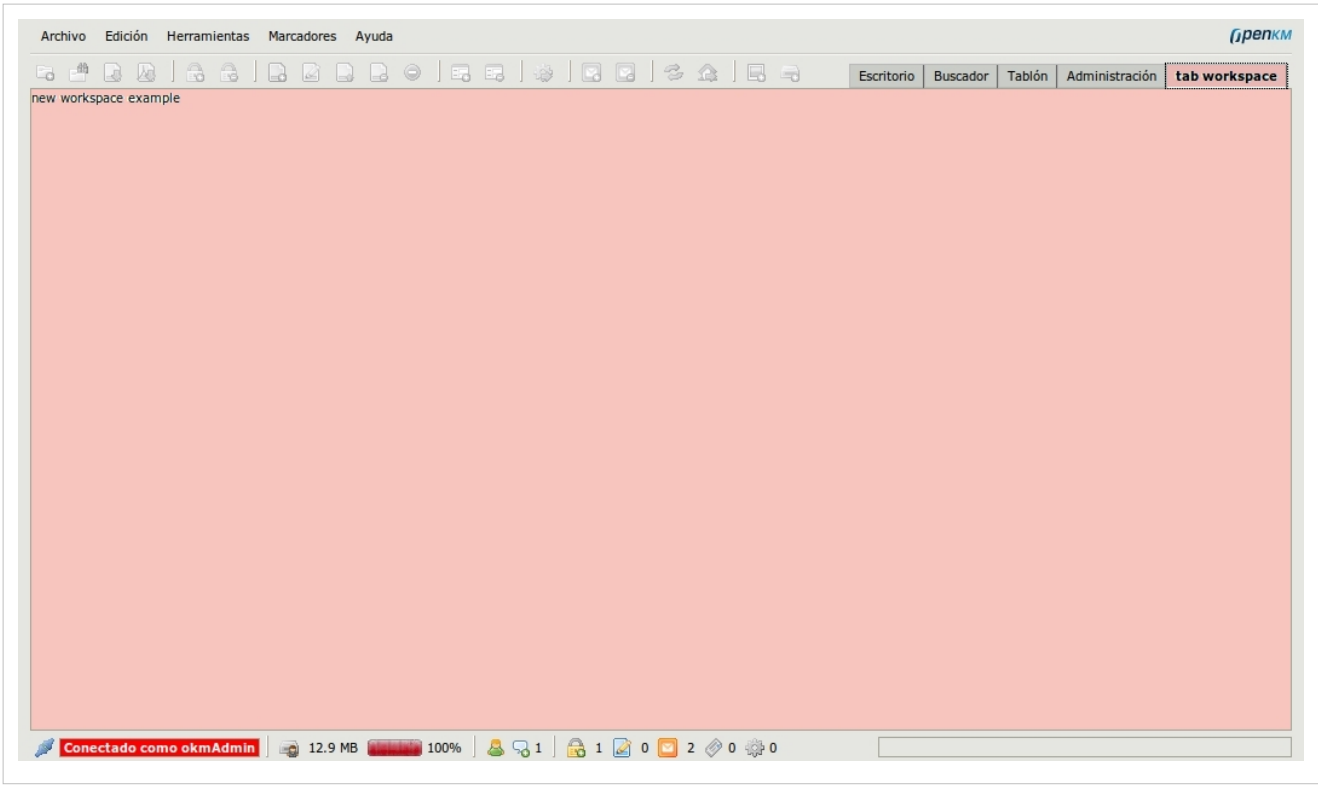
RPCService class defines all RPC constants service that can be used

Example

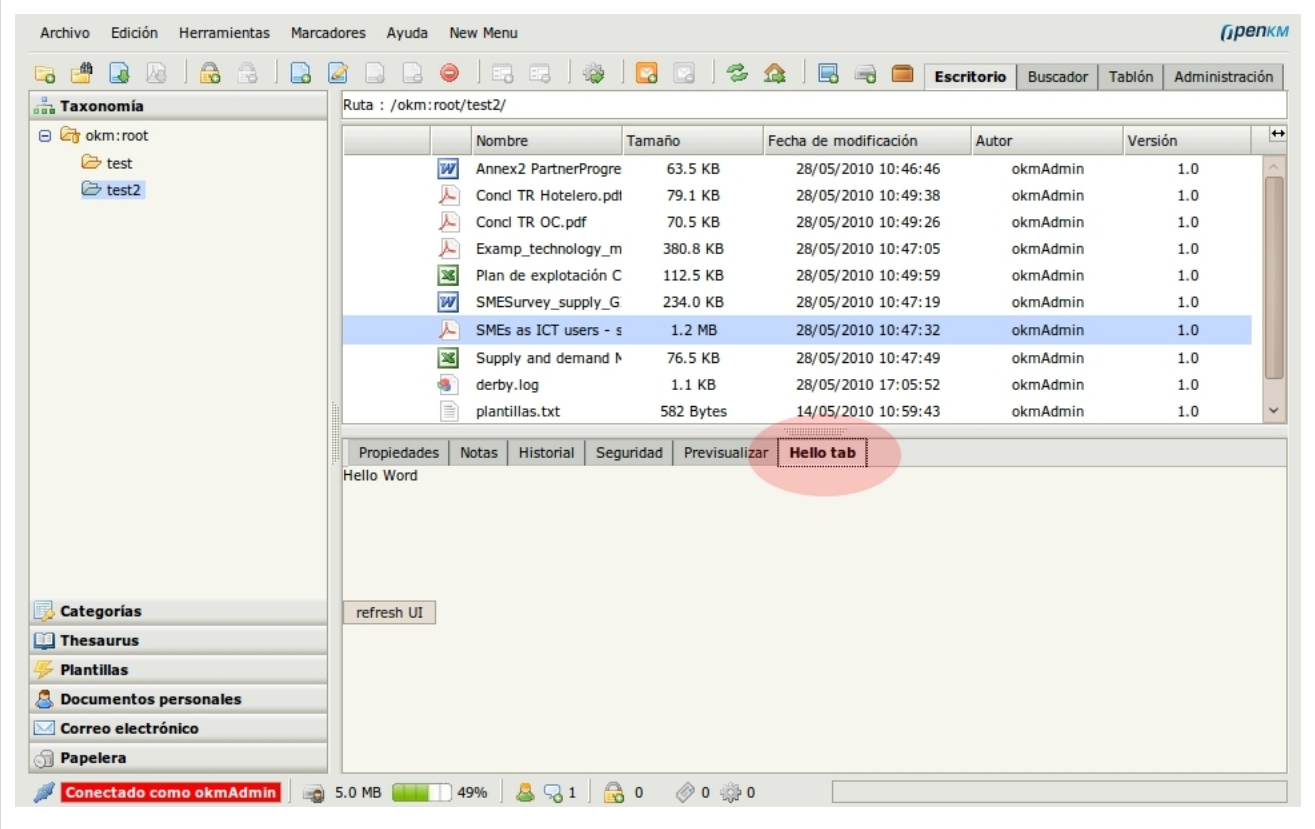
```
ServiceDefTarget endPoint = (ServiceDefTarget) messageService;
endPoint.setServiceEntryPoint(RPCService.MessageService);
messageService.deleteSent(1, new AsyncCallback<Object>() {
        @Override
      public void onSuccess(Object result) {
      }

      @Override
      public void onFailure(Throwable caught) {
      }
});
```

Widget library

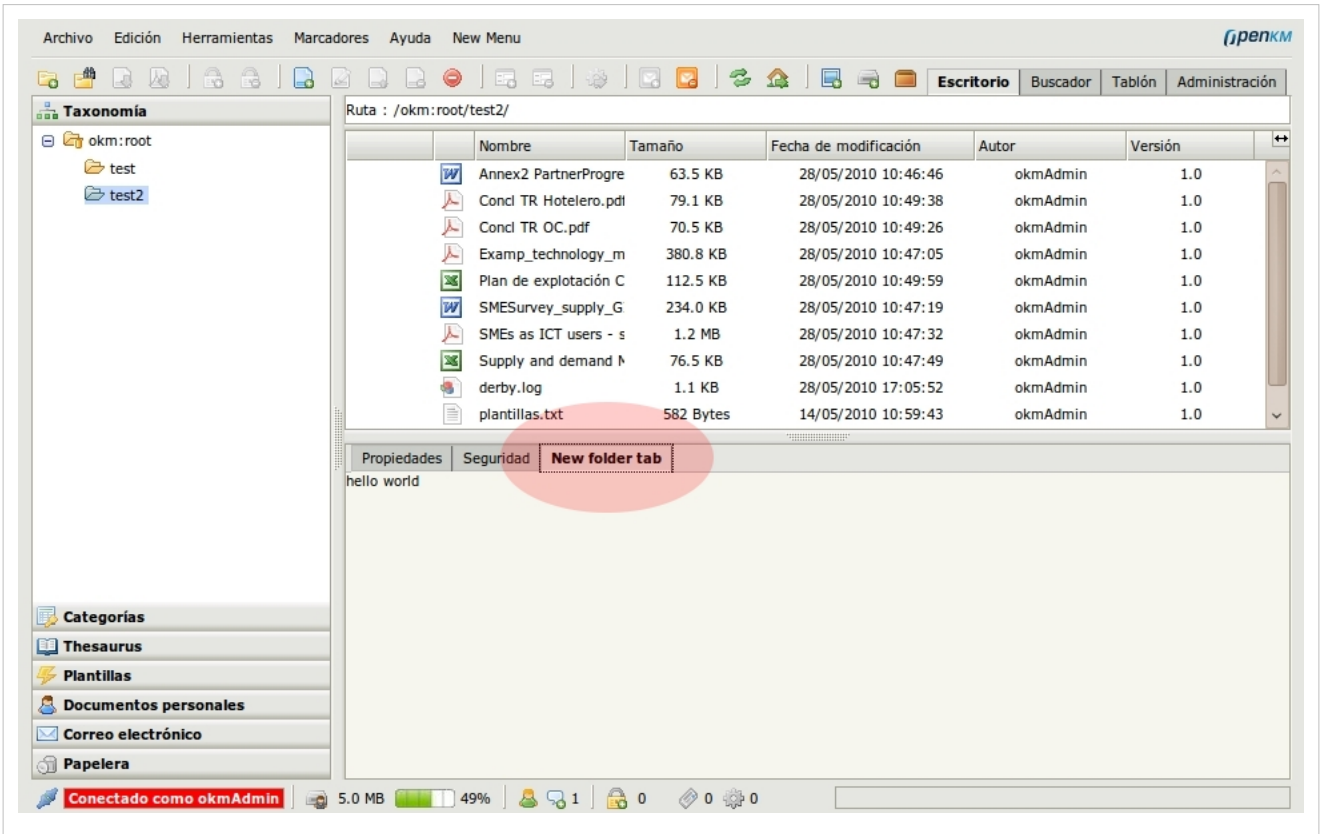
Tab Workspace



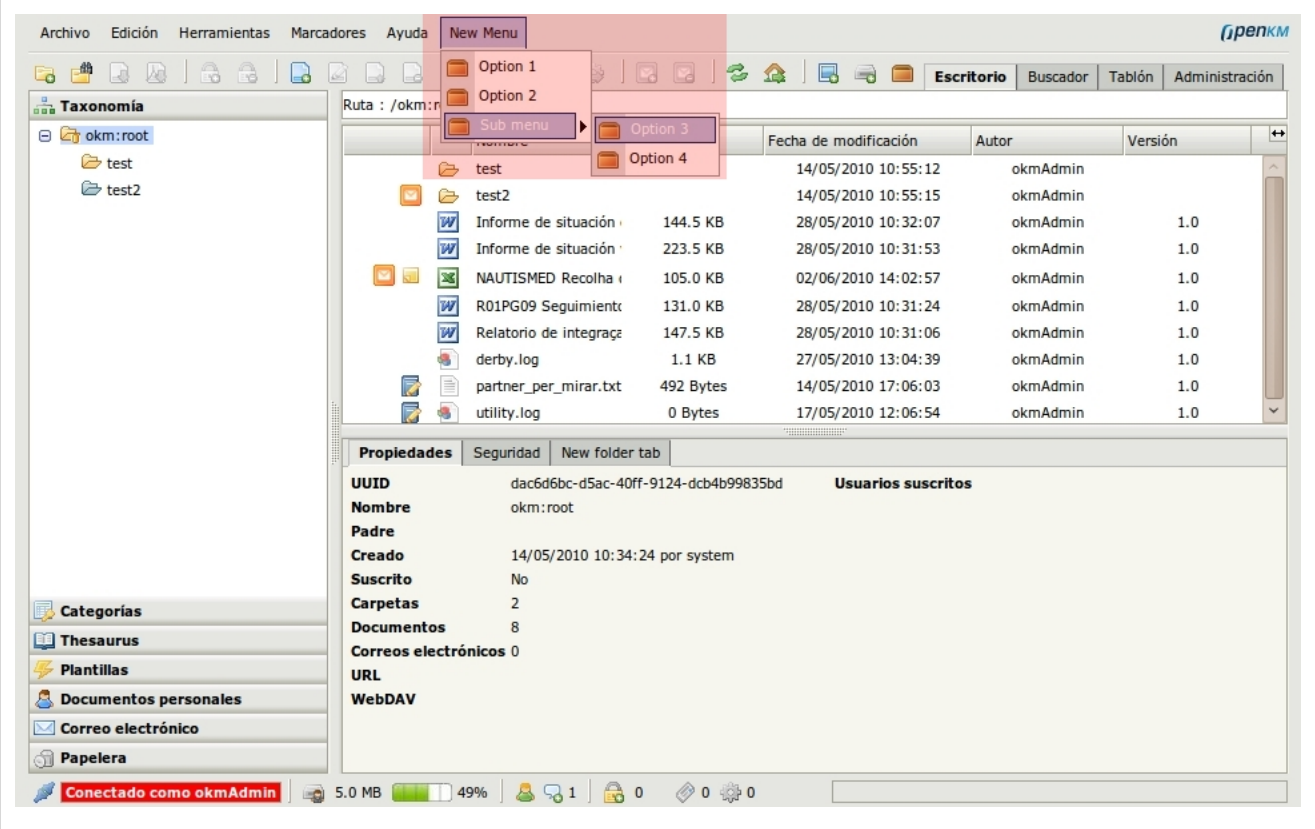
Tab document



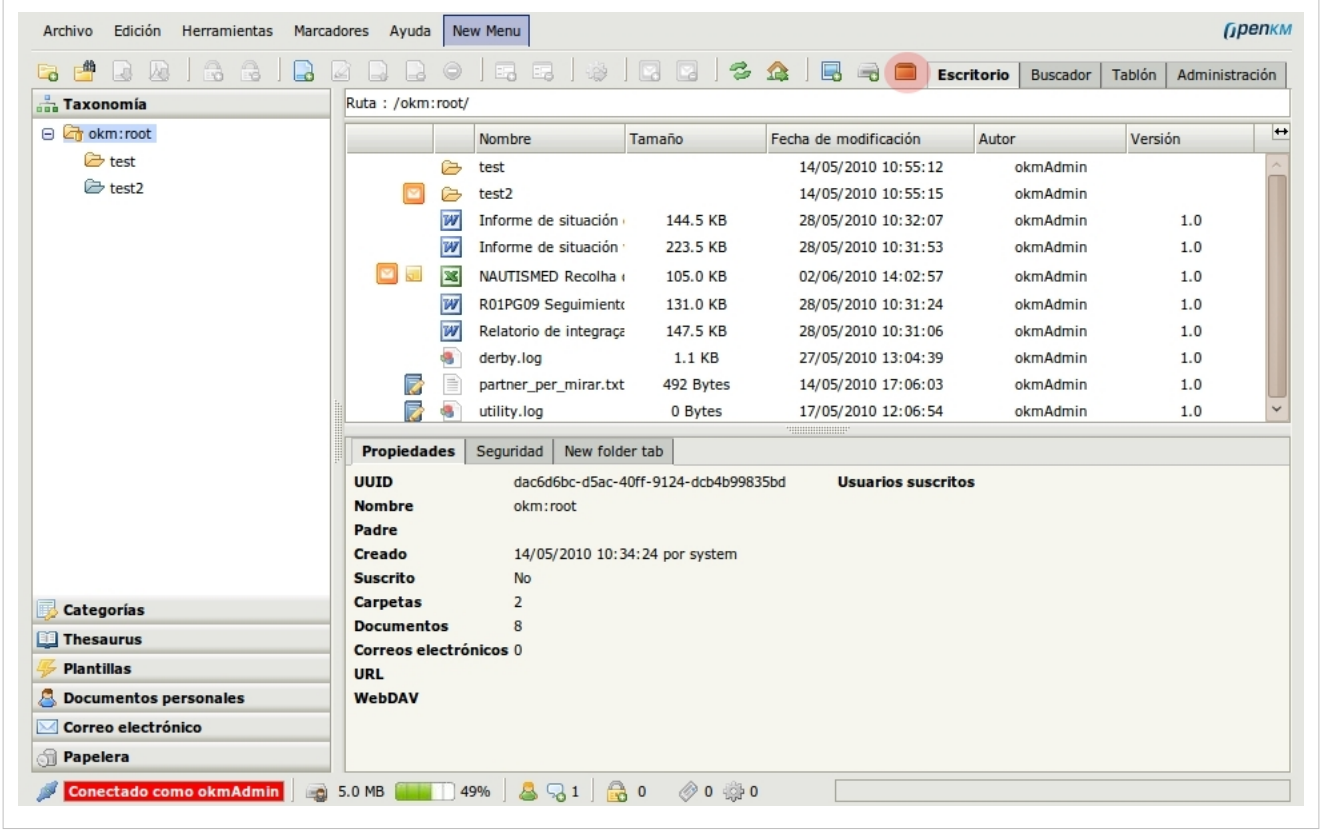
Tab folder



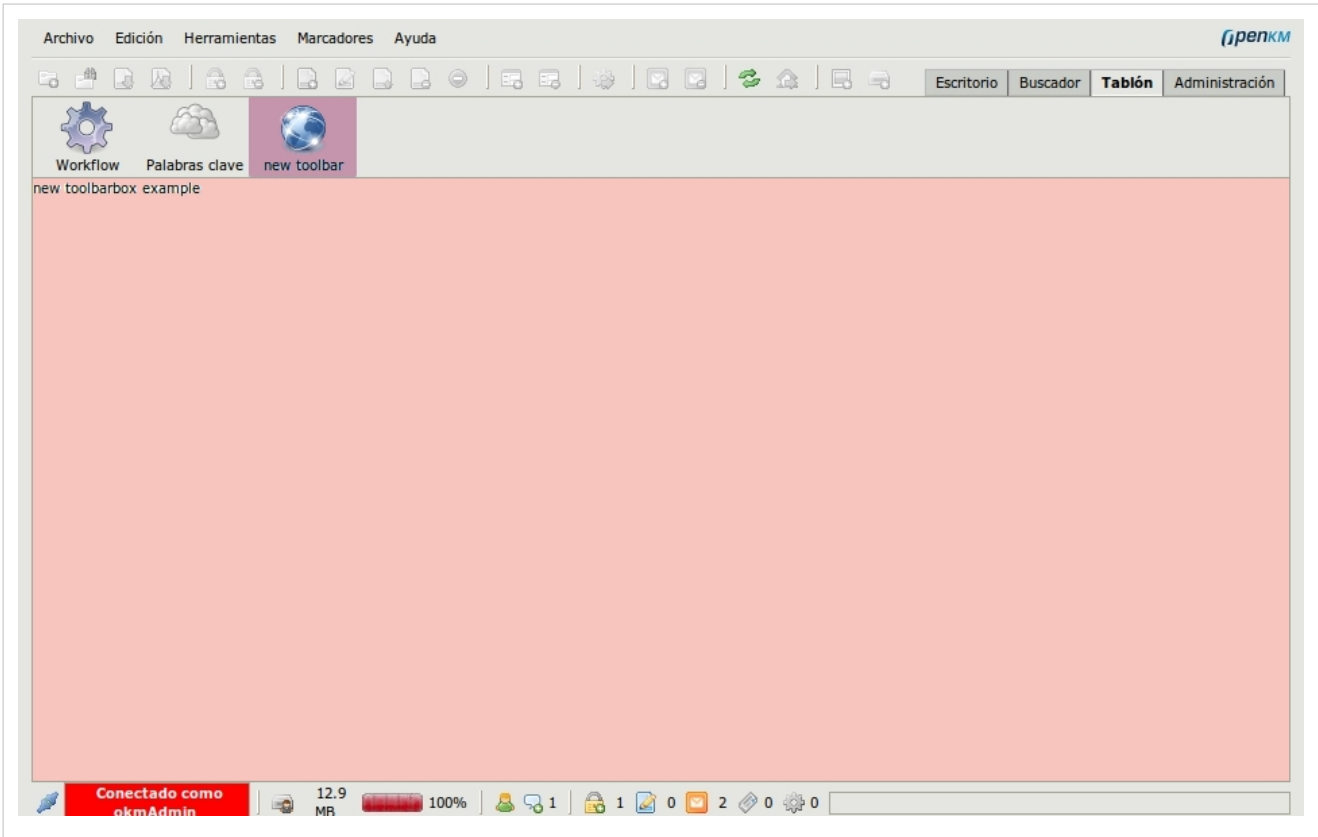
Main menu



Toolbar button



Dashboard



Adding RPC (Remote Process Calls) to server

Servlets are used to doing GWT RPC (Remote Process Calls). In order to registering a new Servlet one file (web.xml) must be modified and OpenKM re-compiled.

Into **web.xml** (src/main/webapp/WEB-INF) must be added the servlet declaration

```
<servlet>
    <servlet-name>SomeServletName</servlet-name>
    <servlet-class>SomeJavaClassName</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>SomeServletName</servlet-name>
    <url-pattern>/SomeServletName</url-pattern>
</servlet-mapping>
```

Internationalization

The idea is add in database some records like

```
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE)
VALUES ('extension', 'new_extension.button.add', 'Add', 'en-GB');
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE)
VALUES ('extension', 'new_extension.button.add', 'Añadir', 'es-ES');
```

And then call in source code with

```
GeneralComunicator.i18nExtension("new_extension.button.add");
```

or you can use some of OpenKM frontend translations with

```
GeneralComunicator.i18n("button.close");
```

OpenKM 4.x and older

It's good idea centralizing Internationalization in a single class package for all OpenKM extensions.

For example if you want to make **internationalization for english and spanish** you migh create two classes **Lang_en_GB.java** and **Lang_es_ES.java**

```
package com.openkm.extension.frontend.client.lang;

import java.util.HashMap;

public class Lang_en_GB {
	public final static HashMap<String, String> lang;
	static {
		lang = new HashMap<String, String>();

		// Download button
		lang.put("download.button.tittle", "Download document");

	}
}
```

```
package com.openkm.extension.frontend.client.lang;

import java.util.HashMap;

public class Lang_es_ES {
	public final static HashMap<String, String> lang;
	static {
		lang = new HashMap<String, String>();

		// Download button
		lang.put("download.button.tittle", "Descargar documento");
	}
}
```

And has a main **Lang.java** class to get **language mapping translations**

```
package com.openkm.extension.frontend.client.lang;

import java.util.HashMap;

public class Lang {
	// Languages
	public static final String LANG_es_ES = "es-ES";
	public static final String LANG_en_GB = "en-GB";

	public static HashMap<String, String> getLang(String lang) {
		HashMap<String, String> hLang = new HashMap<String, String>();

		if (LANG_es_ES.equalsIgnoreCase(lang) ||
LANG_es_ES.substring(0, 2).equalsIgnoreCase(lang.substring(0, 2)))  {
			hLang = Lang_es_ES.lang;
		} else if (LANG_en_GB.equalsIgnoreCase(lang) ||
LANG_en_GB.substring(0, 2).equalsIgnoreCase(lang.substring(0, 2)))  {
			hLang = Lang_en_GB.lang;
		} else {
			hLang = Lang_en_GB.lang;
		}

		return hLang;
	}
}
```

As you can see in **Lang.java** the variables **LANG_es_ES** and **LANG_en_GB** refers to **"es-ES"** and **"en-GB"**. When in your extensions makes a call to **GeneralComunicator.getLang()** to get language it returns a String with that format **"language-Country"**.

Here there's some example how you could use it in your extensions:

```
	String lang = GeneralComunicator.getLang();
	String translation =
Lang.getLang(lang).get("download.button.tittle")
```

Remember any extension internationalized must implements LanguageHandlerExtension, for example:

```
private class ToolBarButton extends ToolBarButtonExtension implements
LanguageHandlerExtension {
	public ToolBarButton(Image image, String title, ClickHandler
handler) {
		super(image, title, handler);
	}

	@Override
	public void checkPermissions(GWTFolder folder, GWTFolder
folderParent, int originPanel) {
		// TODO
```

```
	}

	@Override
	public void checkPermissions(GWTDocument doc, GWTFolder folder) {
		// TODO
	}

	@Override
	public void checkPermissions(GWTMail mail, GWTFolder folder) {
		// TODO
	}

	@Override
	public void enable(boolean enable) {
		// TODO
	}

	@Override
	public boolean isEnabled() {
		return true;
	}

	@Override
	public void onChange(LanguageEventConstant event) {
		if (event.equals(HasLanguageEvent.LANGUAGE_CHANGED)) {
			String lang = GeneralComunicator.getLang();

setTitle(Lang.getLang(lang).get("download.button.tittle"));
		}
	}
}
```

Database Metadata

When you create an extension, the need for a database to store data is very common. You can create tables, but also need to create a bean with Hibernate XML mapping or annotations, a DAO, etc. The other way is creating meta-tables. These virtual tables are part of the OpenKM 5.1 Database Metadata feature. Let's see an example.

Actually there are several data types available:

- text
- boolean
- integer
- long

Our customer wants us to create a contact management feature. For this, we are going to create the metadata structure:

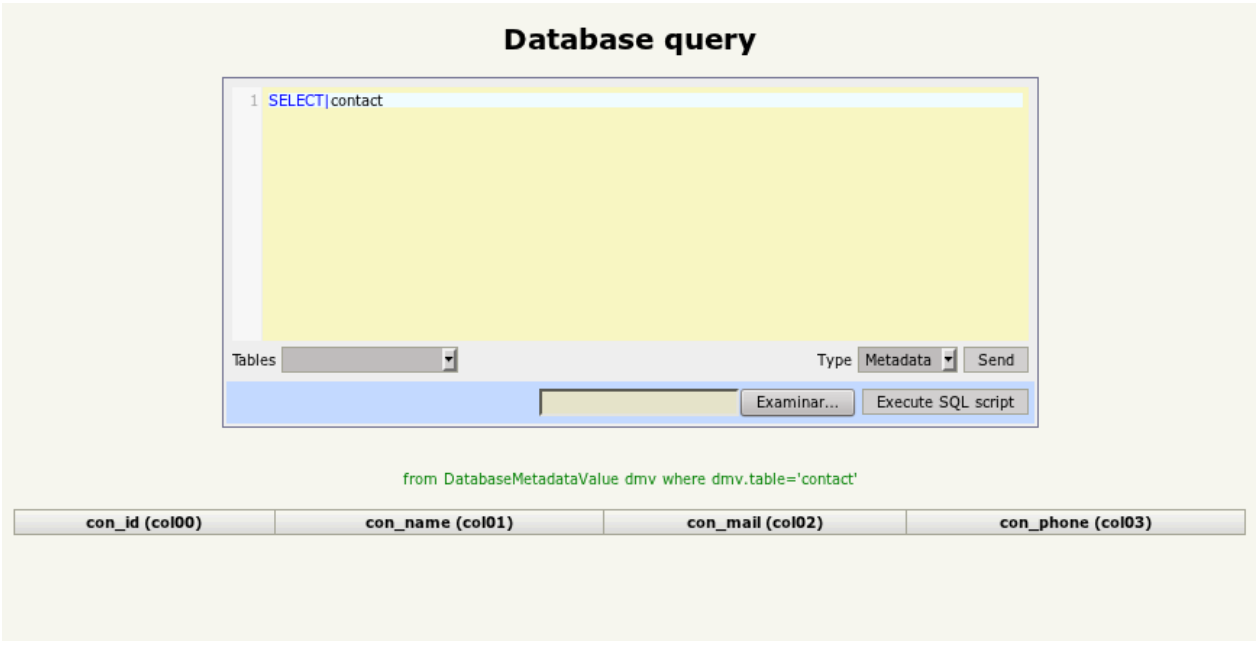
```
INSERT INTO OKM_DB_METADATA_TYPE (DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE,
 DMT_VIRTUAL_COLUMN) VALUES ('contact', 'col00', 'integer', 'con_id');
INSERT INTO OKM_DB_METADATA_TYPE (DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE,
 DMT_VIRTUAL_COLUMN) VALUES ('contact', 'col01', 'text', 'con_name');
INSERT INTO OKM_DB_METADATA_TYPE (DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE,
 DMT_VIRTUAL_COLUMN) VALUES ('contact', 'col02', 'text', 'con_mail');
INSERT INTO OKM_DB_METADATA_TYPE (DMT_TABLE, DMT_REAL_COLUMN, DMT_TYPE,
 DMT_VIRTUAL_COLUMN) VALUES ('contact', 'col03', 'text', 'con_phone');
```

In this sample, the meta-table "contact" contains 4 columns:

- COL 0 -> con_id
- COL 1 -> con_name
- COL 2 -> con_mail
- COL 3 -> con_phone

Actually a meta-table can contain no more than 15 columns.

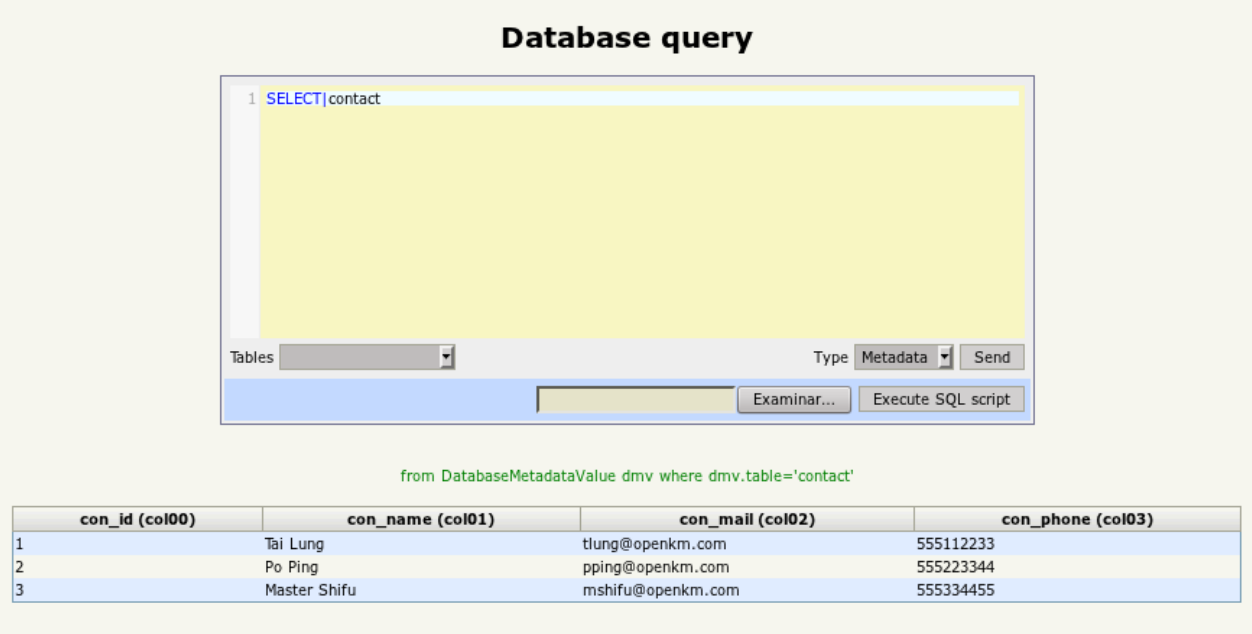
And if you go to **Administration** -> **Database Query** you can see this new empty table:



Let's insert some data:

```
INSERT INTO OKM_DB_METADATA_VALUE (DMV_TABLE, DMV_COL00, DMV_COL01,
DMV_COL02, DMV_COL03) VALUES ('contact', '1', 'Tai Lung',
'tlung@openkm.com', '555112233');
INSERT INTO OKM_DB_METADATA_VALUE (DMV_TABLE, DMV_COL00, DMV_COL01,
DMV_COL02, DMV_COL03) VALUES ('contact', '2', 'Po Ping',
'pping@openkm.com', '555223344');
INSERT INTO OKM_DB_METADATA_VALUE (DMV_TABLE, DMV_COL00, DMV_COL01,
DMV_COL02, DMV_COL03) VALUES ('contact', '3', 'Master Shifu',
'mshifu@openkm.com', '555334455');
```

This is the executed query again:



As you can see, now the inserted data is shown under its correct column. But this is not all, you can also filter these results using this syntax:

```
SELECT|contact|$con_name='Po Ping'
```

Which will display only results with virtual column "con_name" has the value "Po Ping". Not the $ symbol to refer to a virtual column. You can learn more on this in the next section.

Database Query syntax

The syntax used in the Database Query is defined as:

```
SENTENCE|TABLES|QUERY
```

Where TABLES is a list of meta-tables separated by a comma.

```
SELECT|TABLE
SELECT|TABLE|FILTER
```

where TABLE is an unique meta-table.

```
UPDATE|TABLE
UPDATE|TABLE|VALUES
```

```
UPDATE|TABLE|VALUES|FILTER
```

where TABLE is an unique meta-table.

```
DELETE|TABLE
DELETE|TABLE|FILTER
```

where TABLE is an unique meta-table.

This is a sample JOIN query using metadata sintax:

```
SENTENCE|expediente,municipio|from DatabaseMetadataValue expe,
DatabaseMetadataValue mun
where expe.table='expediente' and mun.table='municipio' and
expe.$exp_mun_id=mun.$mun_id
```

Use from Java

Obviously Database metadata can also be used from Java. This way you can implement your own extensions which make use of this feature. This can be achieved by making use of these static methods:

```
String DatabaseMetadataUtils.buildQuery(String table, String filter,
String order)

String DatabaseMetadataUtils.buildUpdate(String table, String values,
String filter)

String DatabaseMetadataUtils.buildDelete(String table, String filter)
```

Each one of these methods will return a Hibernate query with the $xxx columns already replaced by its real-column counterpart. And this Hibernate query can be executed, for example. by:

```
List<Object> LegacyDAO.executeQuery(String query)
```

The returned list, in this case, will be a list of DatabaseMetadataValue objects.

Javascript API

There're several UI javascript function public exposed. In some OpenKM customization cases is interesting accessing some public UI javascript functions (take a look at this video tab embedded web application [1]).

General Functions

- jsI18n(String) - Returns frontend trasnlation where String is the translation key.
- jsOpenWikiPage(String) - Open wiki page where String is wiki title.
- jsOpenUserTaskInstance(String) - Open dashboard workflow view with user taskinstance, where String is the taskinstance.
- jsOpenPath(String, String) - Open some path in OpenKM UI. First String is folder path, second string is document path.
- jsopenPathByUuid(String) - Open some path in OpenKM UI where String indicates uuid.
- jsRefreshFolder() - Refresh ui, same as toolbar refresh button
- jsWizard(String, String) - Call wizard where first String is docPath and second String is json to string serialization of GWTFileUploadResponse bean.
- i18n(String) - Returns frontend trasnlation where String is the translation key (deprecated from openkm 6.x).
- openWikiPage(String) - Open wiki page where String is wiki title (deprecated from openkm 6.x).
- openPath(String, String) - Open some path in OpenKM UI. First String is folder path, second string is document path. (deprecated from openkm 6.x).
- openPathByUuid(String) - Open some path in OpenKM UI where String indicates uuid. (deprecated from openkm 6.x).
- refreshFolder() - Refresh ui, same as toolbar refresh button (deprecated from openkm 6.x).
- jsOpenUserTaskInstance(taskInstanceId) - Open workflow with taskInstanceId
- jsCancelCheckout() - Cancel selected document checkout.

Specific Cryptography Functions

- cryptographyLoaded() - Hide UI status indicating cryptography applet is yet loaded.
- digitalSignatureEnded() - Indicates digital signature has finished.
- digitalSignatureCanceled() - Cancelling digital signature has been canceled.
- startDigitalSignature() - Start digital signature.

Specific Applets Functions

- destroyScannerApplet() - Indicates should be cleaned the scanner applet
- destroyUploaderApplet() - Indicates should be cleaned the uploader applet

Specific HTMLEditor Functions

- jsSearchDocumentHTMLEditorPopup - Open search document popup
- jsSearchFolderHTMLEditorPopup - Open search folder popup

specific Wiki Functions

- jsOpenWikiPage(String) - Open wiki page with some title
- openWikiPage(String - Open wiki page with some title (deprecated from openkm 6.x).

How expose GWT public methods to JS

Example 1- how to expose **static method** called **i18n** to public js as jsI18n

```
/**
 * initJavaScriptApi
 */
native void initJavaScriptApi() /*-{
      $wnd.jsI18n = function(s) {
            return
@com.openkm.frontend.client.Main::i18n(Ljava/lang/String;)(s);
      };
}-*/;
```

Example 2- How to expose **non static method** from class Toolbar.java to public js as jsRefreshFolder.

After Toolbar.java has been created can be called the initJavaScriptApi:

```
ToolBar toolBar = new ToolBar();
mainPanel.topPanel.toolBar.initJavaScriptApi(mainPanel.topPanel.toolBar);
```

Toolbar.java code:

```
/**
 * initJavaScriptApi
 */

public native void initJavaScriptApi(ToolBar toolBar) /*-{
      $wnd.jsRefreshFolder =
toolBar.@com.openkm.frontend.client.widget.toolbar.ToolBar::executeRefresh();
}-*/;
```

For more information take a look at [latest GWT DevGuideCodingBasicsJSNI [2]]

References

[1] http://www.openkm.com/es/openkm-extensibilidad.html
[2] https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsJSNI

Article Sources and Contributors

Extension Guide *Source*: http://wiki.openkm.com/index.php?oldid=7010 *Contributors*: Anonymous, Jllort, Megamansgo.ok, Pavila

Core extensions *Source*: http://wiki.openkm.com/index.php?oldid=5229 *Contributors*: Anonymous, Pavila

Frontend extensions *Source*: http://wiki.openkm.com/index.php?oldid=7585 *Contributors*: Jllort, Pavila

HelloWorld Example *Source*: http://wiki.openkm.com/index.php?oldid=3101 *Contributors*: Jllort, Pavila

Enable example extensions *Source*: http://wiki.openkm.com/index.php?oldid=4798 *Contributors*: Jllort, Pavila

TabWorkspaceExtension *Source*: http://wiki.openkm.com/index.php?oldid=3111 *Contributors*: Jllort, Pavila

TabDocumentExtension *Source*: http://wiki.openkm.com/index.php?oldid=7208 *Contributors*: Jllort, Pavila

ToolBarBoxExtension *Source*: http://wiki.openkm.com/index.php?oldid=3113 *Contributors*: Jllort, Pavila

TabFolderExtension *Source*: http://wiki.openkm.com/index.php?oldid=7207 *Contributors*: Jllort, Pavila

TabMailExtension *Source*: http://wiki.openkm.com/index.php?oldid=7202 *Contributors*: Jllort, Pavila

TabRecordExtension *Source*: http://wiki.openkm.com/index.php?oldid=7205 *Contributors*: Jllort

ToolBarButtonExtension *Source*: http://wiki.openkm.com/index.php?oldid=6745 *Contributors*: Jllort, Pavila

MenuItemExtension *Source*: http://wiki.openkm.com/index.php?oldid=3117 *Contributors*: Jllort, Pavila

MenuBarExtension *Source*: http://wiki.openkm.com/index.php?oldid=3118 *Contributors*: Jllort, Pavila

WorkspaceHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3119 *Contributors*: Jllort, Pavila

NavigatorHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3120 *Contributors*: Jllort, Pavila

DocumentHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3121 *Contributors*: Jllort, Pavila

FolderHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3122 *Contributors*: Jllort, Pavila

MailHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=7209 *Contributors*: Jllort

RecordHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=7210 *Contributors*: Jllort

ToolBarHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3123 *Contributors*: Jllort, Pavila

PropertyGroupHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3124 *Contributors*: Jllort, Pavila

DashboardHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=3125 *Contributors*: Jllort, Pavila

WidgetHandlerExtension *Source*: http://wiki.openkm.com/index.php?oldid=6749 *Contributors*: Jllort

GeneralComunicator *Source*: http://wiki.openkm.com/index.php?oldid=7392 *Contributors*: Jllort, Pavila

WorkspaceComunicator *Source*: http://wiki.openkm.com/index.php?oldid=5684 *Contributors*: Jllort, Pavila

NavigatorComunicator *Source*: http://wiki.openkm.com/index.php?oldid=5611 *Contributors*: Jllort, Pavila

FileBrowserComunicator *Source*: http://wiki.openkm.com/index.php?oldid=6760 *Contributors*: Jllort, Pavila

TabDocumentComunicator *Source*: http://wiki.openkm.com/index.php?oldid=6063 *Contributors*: Jllort, Pavila

TabFolderComunicator *Source*: http://wiki.openkm.com/index.php?oldid=6064 *Contributors*: Jllort, Pavila

TabMailComunicator *Source*: http://wiki.openkm.com/index.php?oldid=6065 *Contributors*: Jllort

TabRecordComunicator *Source*: http://wiki.openkm.com/index.php?oldid=7211 *Contributors*: Jllort

UtilComunicator *Source*: http://wiki.openkm.com/index.php?oldid=7425 *Contributors*: Jllort, Pavila

DashboardComunicator *Source*: http://wiki.openkm.com/index.php?oldid=3147 *Contributors*: Jllort, Pavila

SearchComunicator *Source*: http://wiki.openkm.com/index.php?oldid=3135 *Contributors*: Jllort, Pavila

ToolBarComunicator *Source*: http://wiki.openkm.com/index.php?oldid=6750 *Contributors*: Jllort

UIMenuConstants *Source*: http://wiki.openkm.com/index.php?oldid=3136 *Contributors*: Jllort, Pavila

UIDesktopConstants *Source*: http://wiki.openkm.com/index.php?oldid=3204 *Contributors*: Jllort

UIDockPanelConstants *Source*: http://wiki.openkm.com/index.php?oldid=3207 *Contributors*: Jllort

UISearchConstants *Source*: http://wiki.openkm.com/index.php?oldid=3210 *Contributors*: Jllort

UIFileUploadConstants *Source*: http://wiki.openkm.com/index.php?oldid=3266 *Contributors*: Jllort

RPCService *Source*: http://wiki.openkm.com/index.php?oldid=3200 *Contributors*: Jllort

Widget library *Source*: http://wiki.openkm.com/index.php?oldid=3141 *Contributors*: Jllort, Pavila

Adding RPC (Remote Process Calls) to server *Source*: http://wiki.openkm.com/index.php?oldid=3208 *Contributors*: Jllort, Pavila

Internationalization *Source*: http://wiki.openkm.com/index.php?oldid=6751 *Contributors*: Jllort, Pavila

Database Metadata *Source*: http://wiki.openkm.com/index.php?oldid=6460 *Contributors*: Anonymous, Pavila

Javascript API *Source*: http://wiki.openkm.com/index.php?oldid=7587 *Contributors*: Jllort, Pavila

Image Sources, Licenses and Contributors

File:Nota_advertencia.png *Source*: http://wiki.openkm.com/index.php?title=File:Nota_advertencia.png *License*: unknown *Contributors*: Pavila

File:Nota clasica.png *Source*: http://wiki.openkm.com/index.php?title=File:Nota_clasica.png *License*: unknown *Contributors*: Pavila

File:Okm_extension_001.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_001.jpeg *License*: unknown *Contributors*: Jllort

File:Okm_extension_008.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_008.jpeg *License*: unknown *Contributors*: Jllort

File:Okm_extension_002.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_002.jpeg *License*: unknown *Contributors*: Jllort

File:Okm_extension_003.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_003.jpeg *License*: unknown *Contributors*: Jllort

File:Okm_extension_006.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_006.jpeg *License*: unknown *Contributors*: Jllort

File:Okm_extension_007.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_007.jpeg *License*: unknown *Contributors*: Jllort

File:Okm_extension_009.jpeg *Source*: http://wiki.openkm.com/index.php?title=File:Okm_extension_009.jpeg *License*: unknown *Contributors*: Jllort

File:Database metadata 01.png *Source*: http://wiki.openkm.com/index.php?title=File:Database_metadata_01.png *License*: unknown *Contributors*: Pavila

File:Database metadata 02.png *Source*: http://wiki.openkm.com/index.php?title=File:Database_metadata_02.png *License*: unknown *Contributors*: Pavila