Image Manager and Image Sets

Image Manager

The `ImageManager` class is the recommended way to load and display images in Odysseus Studio. It provides advanced features like lazy loading and caching while it is still easy to use.

If you want to provide images with your bundle, the bundle's Activator class should manage an instance of the `ImageManager`. Before you can use the image manager to access the images, you must create the instance and register your images in the bundle's start method.

Set up image manager

```
@Override
public void start(BundleContext bundleContext) throws Exception {
    super.start(bundleContext);

        // create instance of ImageManager
        imageManager = new ImageManager(bundleContext.getBundle());

        // register image
        imageManager.register("someImageID", "icons/someImage.png");

        // ...
}
```

You must pass the current bundle to the constructor of the image manager. Otherwise it won't be able to access the image files located in your bundle. After the initialization of the image manager, you need to register each image using the `register(...)` method. The first parameter is the image ID which must be unique for the image manager instance. The second parameter is the path of your image file inside your bundle.

After you've registered your images, you can easily access and load the image. All you need to do is to access the image manager from your bundle (or any other available bundle's activator providing an image manager) and call the get(...) method passing the image ID of the requested image.

Retrieving image from image manager

```
Image image = MyRCPPlugIn.getImageManager().get("someImageID");
```

Image sets

Registering each image indvidually in the start method of the plugins Activator class tends to be very cumbersome if you have a huge number of images. Besides the many lines of code you have to write, it is requires code changes to update or add new images. Alternatively, you can use image sets to avoid these drawbacks.

An image set is a collection of one or more images that is defined in a XML-serialized Java property file. Each image is represented as a key-value pair consisting of the image ID (as the key) and the relative file path (as the value):

Image set property file example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
        <comment>Image set configuration file. Edit only if you know what you are doing</comment>
        <entry key="DEFAULT">operator-images/default/gear.png</entry>
        <entry key="JOIN">operator-images/default/join2.png</entry>
        <!-- more image definitions -->
</properties>
```

Similar to single images, you must register an image set at the Image Manager before you can use it:

Register image set example

```
@Override
public void start(BundleContext bundleContext) throws Exception {
    super.start(bundleContext);

        // create instance of ImageManager
        imageManager = new ImageManager(bundleContext.getBundle());

        // register image set
        imageManager.registerImageSet("white", "operator-images/icons_white.xml");

        // ...
}
```

The `registerImageSet(...)` method requires two arguments. The first argument is the name of the image set which can be considered as the namespace of the images contained in the image set. It is added as prefix seperated by a dot (".") to each image id in order to avoid name clashes between different image sets. The second argument is the relative path to the property file that specifies the image set.

You can access images from registered image sets in the same way like you can do for individually registered images. However, you need to qualify the image ID with image set name as the prefix.

Retrieving an image from an image set

```
Image image = MyRCPPlugIn.getImageManager().get("imageSetName.someImageID");
```

The `OdysseusRCPPlugIn` class provides different image sets (`white`, `black`, and `default`) each containing a number of illustrating icons for physical operators.

User-defined image sets

Image sets can be provided by a plugin. In this case, all images and the image set property file must be stored inside the bundle and all paths must be relative to the bundle root. However, it is also possible to create user-defined image sets. They can be stored in and loaded from the Odysseus home directory.

User-defined image sets are created in the same way like bundle-based image sets. However, there are a few differences you should consider:

- The property files that define the image sets must be stored in a folder called `icons` located in the Odysseus home directory.
- The property file name must be the same as the image set file name with the suffix ".xml".
- All paths inside the property files must be relative to the Odysseus home directory.
- Even though the images may be stored at any location inside the Odysseus home directory, it is strongly recommended to store them in a subdirectory of the `icons` directory which has the same name like the image set name.

All user-defined image sets are loaded at the start of Odysseus Studio and can be accessed via the `OdysseusRCPPlugIn` class instance.

Using user-defined image sets for operator graph visualization styles

With user-defined image sets, it is quite simple to apply your own icons in the operator graph visualization:

1. Create your user-defined image set as described above
    - To map an image to a specific physical oeprator, use the class name (without packages or generics) of the physical operator as the image ID
    - You must define a default image in the image set with image ID `DEFAULT`. This image is used as a fallback, if your icon set does not define an image for a specific operator.
2. Create you own viewer configuration file as described here.
    - Use the `SymbolElement` element of type `operator` to display the operator image.
    - Set its `iconSetName` parameter to the name of your image set (don't set the `resource` parameter!).

You can also use user-defined image sets to override specific images of bundle-based image sets. To do this, you need to create an user-defined image set of the same name and define the image ID of the orginal image as the image ID of the new image.