## \\mataraserver\user\Chathurangi\Logo-final.jpg

**SE3030 – Software Architecture**

**3rd Year – Semester 01**

**Lecturer: By Udara Samaratunge**

## Apache Felix (OSGi) Lab Sheet

* **Go to the download page and download and extract the Felix release**
  + [***http://felix.apache.org/site/downloads.cgi***](http://felix.apache.org/site/downloads.cgi)
  + ***http://apache.cs.utah.edu//felix/org.apache.felix.main.distribution-4.0.3.zip***
* **Go to the Felix usage page to learn how to launch the Felix framework**
  + ***http://felix.apache.org/site/apache-felix-framework-usage-documentation.html***

**Dictionary Service Bundle**

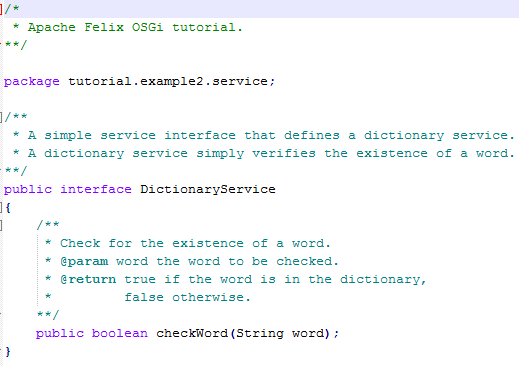
* 1. Creates a bundle that implements an OSGi service.
  2. Implementing an OSGi service is a two-step process
  3. First we must define the interface of the service and then we must define an implementation of the service interface.

**Client service bundle**

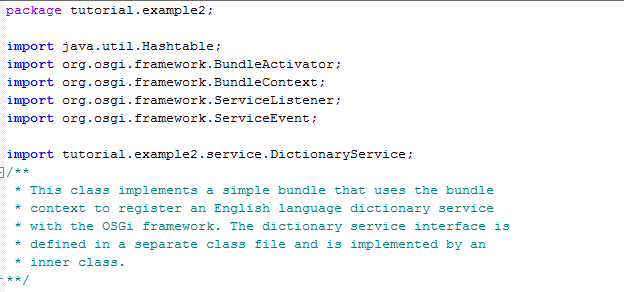
To consume the dictionary service.

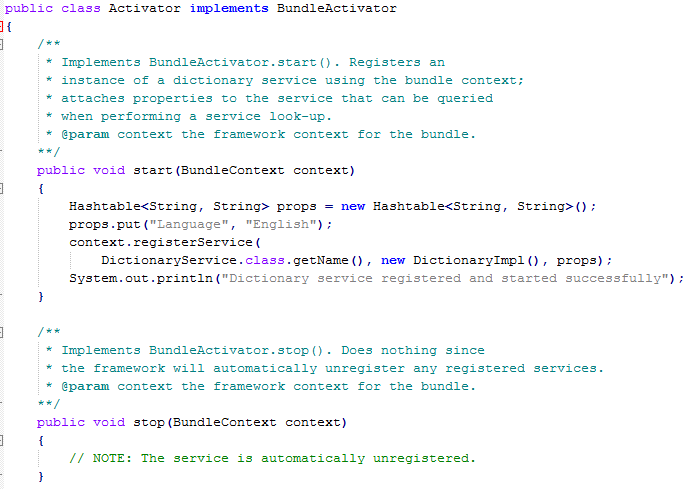
**Export dictionary service and use the service**

1. Dictionary service interface in a file called *DictionaryService.java*

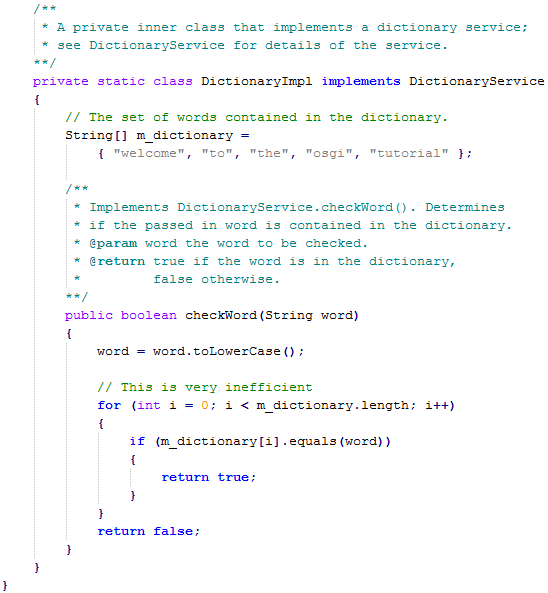


1. In the following source code, the bundle uses its bundle context to register the dictionary service. We implement the dictionary service as an inner class of the bundle activator class, but we could have also put it in a separate file. The source code for our bundle is as follows in a file called Activator.java:

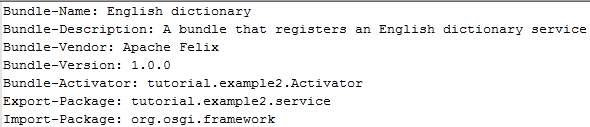




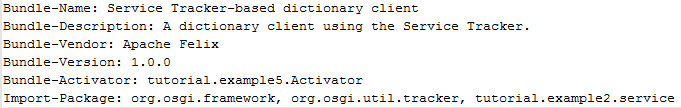
1. You have to create inner class that implements *DictionaryService* interface.



1. We must create a manifest.mf file that contains the meta-data for our bundle; the manifest file contains the following (manifest\_example2.mf) for Dictionary Service



1. Manifest class of consumer is as follow (manifest\_example5.mf)



1. We specify which class is used to activate our bundle via the Bundle-Activator attribute.
2. Export-Package attribute makes it possible for other bundles to import our dictionary service interface.
3. The Import-Package attribute informs the framework of the bundle's dependencies on external packages.
4. All bundles with an activator must import org.osgi.framework since it contains the core OSGi class definitions.
5. To compile our source code, we need the felix.jar file (found in Felix' bin directory) in our class path.

javac -cp C:\Udara\felix-framework-4.0.3\bin\felix.jar \*.java service\\*.java

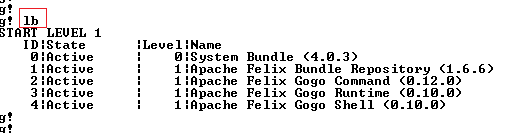
1. This command compiles all source files and generates class files.
2. After compiling, we need to create a JAR file containing the generated package directories.
3. We will also add our manifest file that contains the bundle's meta-data to the JAR file. To create the JAR file, we issue the command.

jar cfm example2.jar manifest.mf -C C:\Udara\tutorial \tutorial\example2

1. Once the JAR file is created, we are ready to install and start the bundle.

**Service installation and subscribe.**

1. When we start Felix, it asks for a profile name, we will put all of our bundles in a **profile** named **tutorial**.
2. After running Felix, we should make sure that the bundle from Example 1 is active.
3. We can use the Felix lb shell command to get a list of all bundles, their state, and their bundle identifier number.



1. Use below command to Login Apachi Felix shell

/felix-framework-4.0.3/> java -jar bin/felix.jar

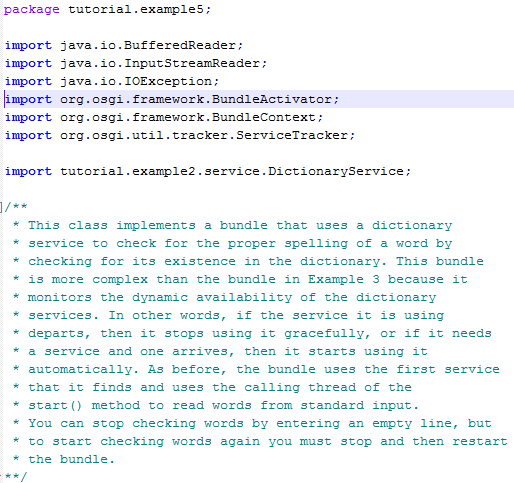
1. If the Example 1 bundle is not active, we should start the bundle using the **start** command
   1. Bundle's identifier number that is displayed by the lb command.
   2. Now we can install and start our dictionary service bundle using bundle number.
2. Assuming that we created our bundle in the directory c:\tutorial, we can install and start it in Felix' shell using the following command:

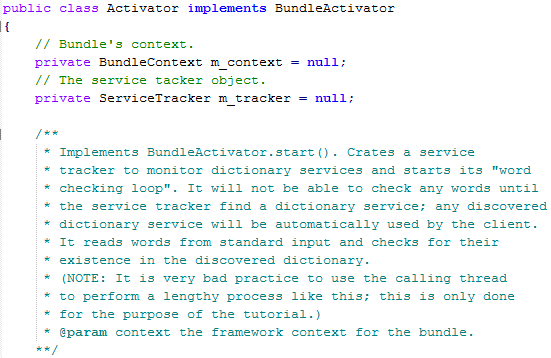
install file:/C:/Udara/tutorial/example2.jar

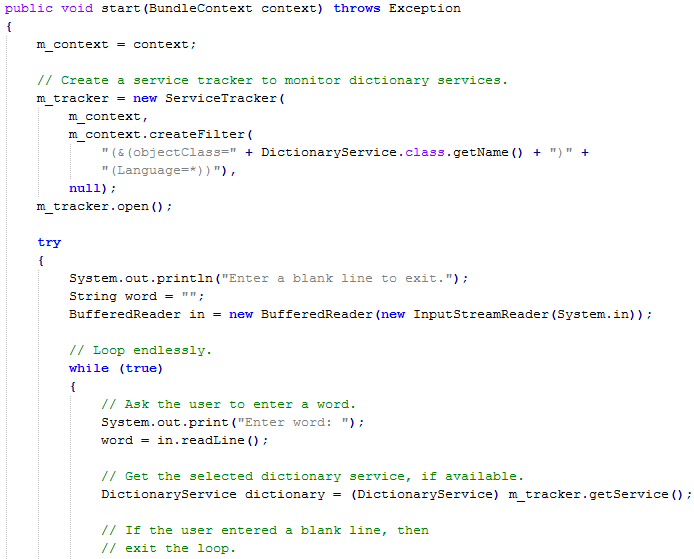
start file:/C:/Udara/tutorial/example2.jar

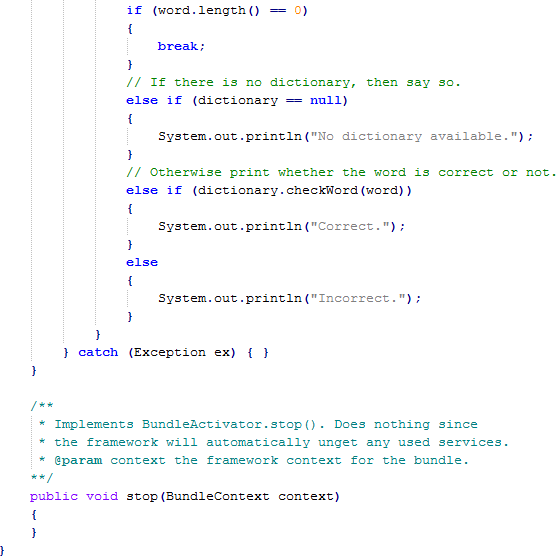
1. It is also possible to install and start the bundle in two steps by using the Felix install and start shell commands. To stop the bundle, use the Felix stop shell command.
2. To subscribe the service create **Service Tracker Dictionary client bundle** and use the installed service as below.

**Service Tracker Dictionary client bundle**

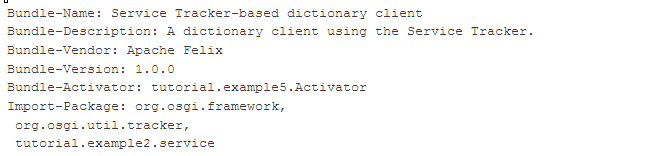








1. Create a separate manifest file as below for this service.

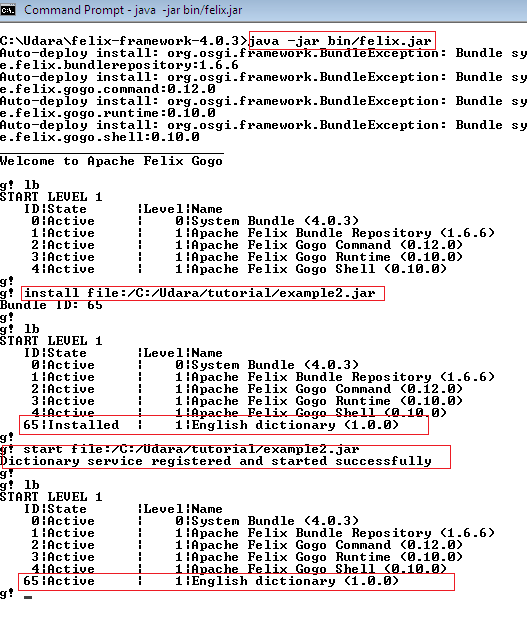


1. You have to import the Dictionary service package you created the in previous example 2 for the manifest.mf file as above.

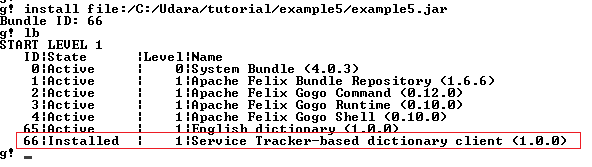
1. Create a separate bundle with compiling above source file and bundle it as example5.jar.

**Some screen shots of final out put**

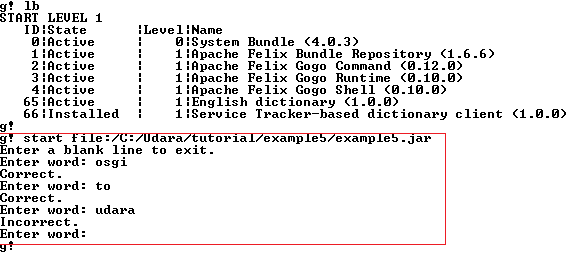
1. **Start to install services.**



1. **Install client service to subscribe dictionary service.**



1. **Start the client service and subscribe the dictionary service.**



1. **Unregister the dictionary service and client attempt to access the service.**

