Software Architecture

3rd Year – Semester 01

OSGi-Equinox-Exercise Lab
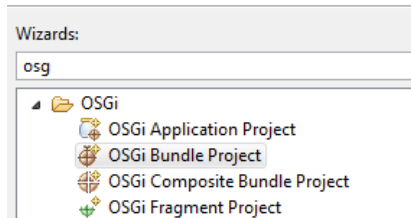
Lecturer - By Udara Samaratunge

Task 01 Create OSGi bundle project
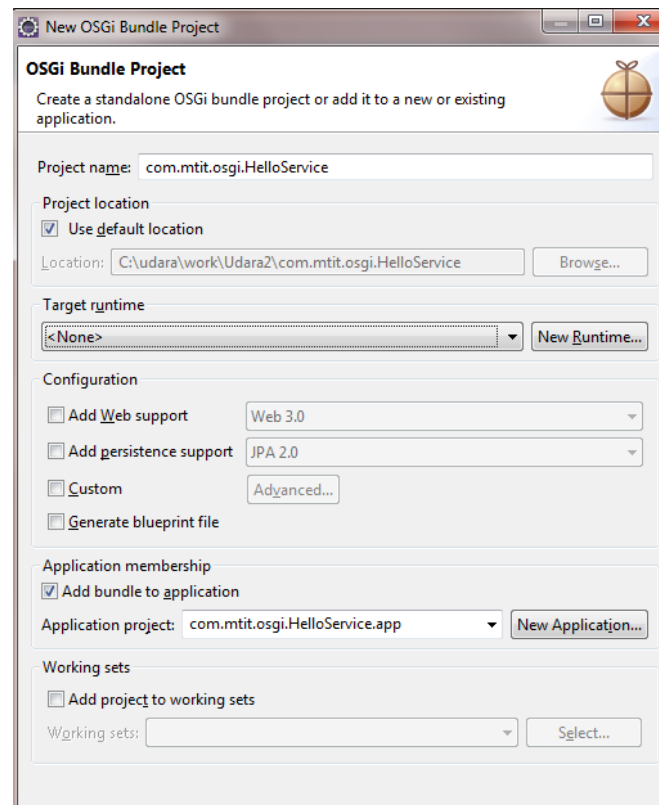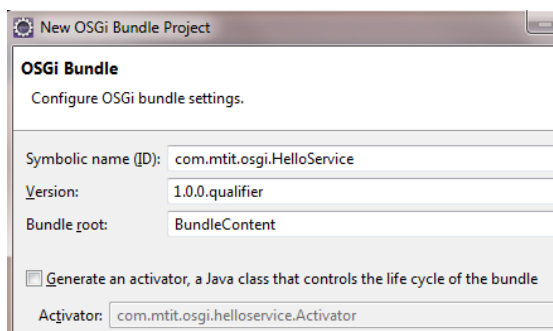
File => new => OSGi bundle project => next

Project name = **com.mtit.osgi.HelloService**

It creates the project as blow.

- com.mtit.osgi.HelloService
  - Manifest: com.mtit.osgi.HelloService
  - src
  - JRE System Library [JavaSE-1.7]
  - OSGi Dependencies
  - Plug-in Dependencies
  - BundleContent
- com.mtit.osgi.HelloService.app
  - Manifest: com.mtit.osgi.HelloService.app
  - META-INF

It create **OSGi Manifest** overview as follows

Activator.java	com.mtit.osgi.HelloService

Overview

General Information

This section describes general information about this plug-in.

ID:	com.mtit.osgi.HelloService
Version:	1.0.0.qualifier
Name:	com.mtit.osgi.HelloService
Vendor:	
Platform Filter:	
Activator:	Browse...

☐ Activate this plug-in when one of its classes is loaded
☐ This plug-in is a singleton

Execution Environments

Specify the minimum execution environments required to run this plug-in.

JavaSE-1.6

Add...
Remove
Up
Down

Configure JRE associations...

Update the classpath settings

Plug-in Content

The content of the plug-in is made up of two sections:

- Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.
- Runtime: lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

This plug-in may define extensions and extension points:

- Extensions: declares contributions this plug-in makes to the platform.
- Extension Points: declares new function points this plug-in adds to the platform.

Testing

Test this plug-in by launching an OSGi framework:

- Run on Server
- Debug on Server

Exporting

To package and export the plug-in:

1. Organize the plug-in using the Organize Manifests Wizard
2. Externalize the strings within the plug-in using the Externalize Strings Wizard
3. Specify what needs to be packaged in the deployable plug-in on the Build Configuration page

Create **Activator.java** and import **osgi framework** as dependencies of manifest.

Activator.java	com.mtit.osgi.HelloService

Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

Add...
Remove
Up
Down
Properties...

Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

org.osgi.framework

Add...
Remove
Properties...
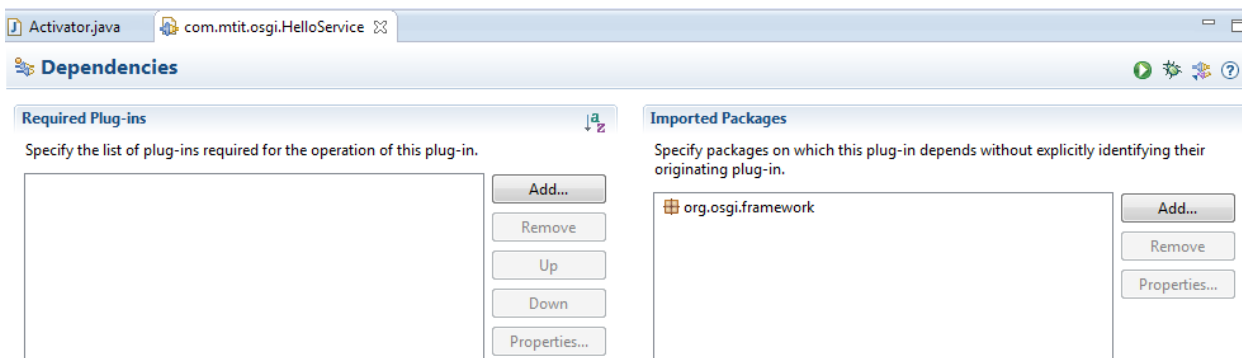
Override the life cycle methods in **Activator.java** is as follows   **Manifest.mf** should be as follows.

```java
  1⊖ import org.osgi.framework.BundleActivator;
  2  import org.osgi.framework.BundleContext;
  3
  4
  5  public class Activator implements BundleActivator{
  6
  7⊖     @Override
  8      public void start(BundleContext context) throws Exception {
  9          System.out.println("Start MTIT Osgi service");
 10      }
 11
 12⊖     @Override
 13      public void stop(BundleContext context) throws Exception {
 14          System.out.println("Stop MTIT Osgi service");
 15      }
 16  }
```

```
 1 Manifest-Version: 1.0
 2 Bundle-ManifestVersion: 2
 3 Bundle-Name: com.mtit.osgi.HelloService
 4 Bundle-SymbolicName: com.mtit.osgi.HelloService
 5 Bundle-Version: 1.0.0.qualifier
 6 Bundle-RequiredExecutionEnvironment: JavaSE-1.6
 7 Import-Package: org.osgi.framework
 8
```

Run Configuration

Select the **manifest** and **run as => run configuration**. Then modify the name **OSGi MTIT Service**
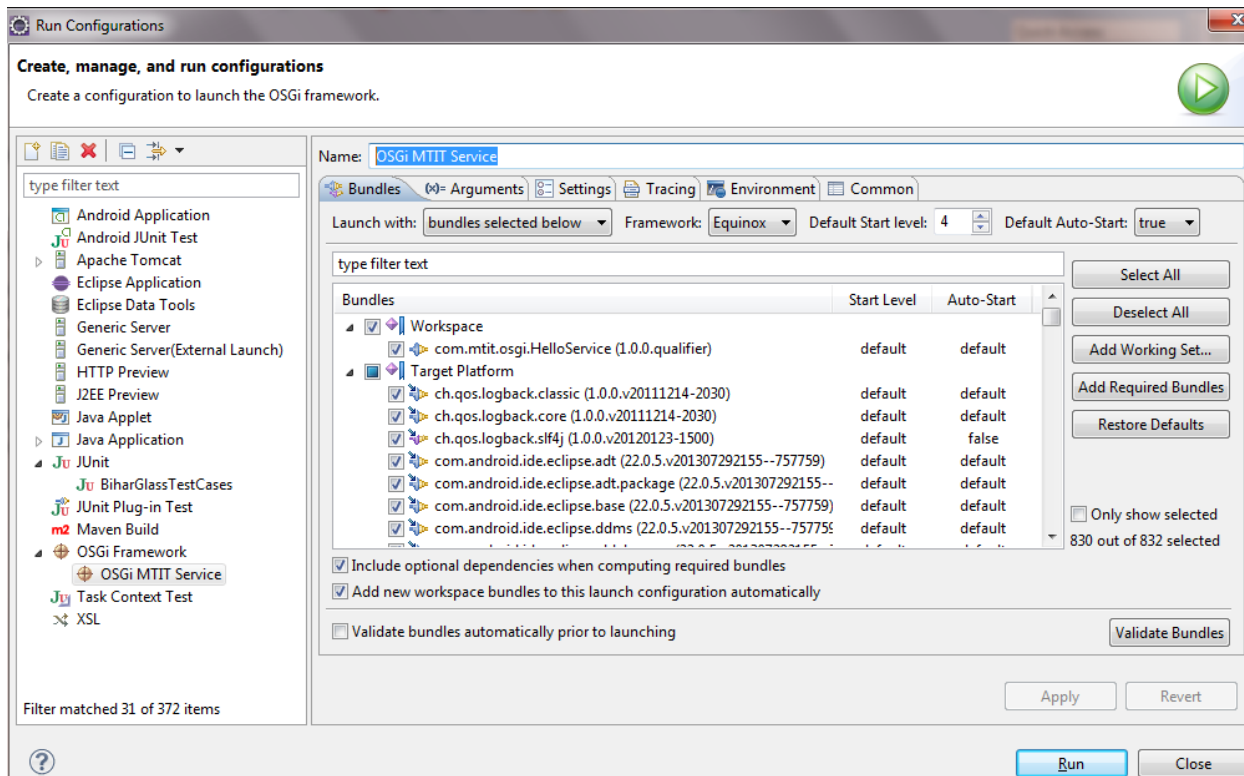
- Now using the **manifest run the project**
- In the console enter **lb** or **ss** and check all available bundles in equinox OSGi console. As below
- MTIT Hello service will be displayed as **activated state**.
- If you want you can stop the service **stop <bundle id>**

But it doesn't display bundle life cycle methods?.......... ☹

Troubleshoot the application

You should modify above program to display the below output. ☺


```
820|Active      |    4|ASM (3.3.1.v201105211655)
821|Active      |    4|SAT4J Core (2.3.0.v20110329)
822|Active      |    4|SAT4J Pseudo (2.3.0.v20110329)
823|Active      |    4|SLF4J API (1.6.4.v20120130-2120)
824|Active      |    4|m2e connector for the mavenarchiver and pom properties (0.
825|Active      |    4|UDDI4J (2.0.5.v200805270300)
826|Active      |    4|W3C CSS SAC (1.3.1.v200903091627)
827|Active      |    4|W3C SMIL DOM (1.0.0.v200806040011)
828|Active      |    4|W3C SVG DOM (1.1.0.v201011041433)
829|Active      |    4|ADT XML Overlay (22.0.5.v201307292155--757759)
830|Active      |    4|com.mtit.osgi.HelloService (1.0.0.qualifier)
osgi> stop 830
Stop MTIT Osgi service
osgi> start 830
Start MTIT Osgi service
```

Task 02 OSGi -Dependency Management (Publisher and Subscriber)
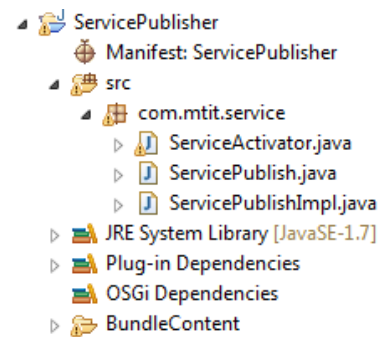
Service Publisher

1. Create a new **OSGi bundle project** => **ServicePublisher** and create interface **ServicePublish** then implement the interface for created java class **ServicePublishImpl**.

2. In **interface** declare the method called **publishService()** and implement it in the **ServicePublishImpl**.
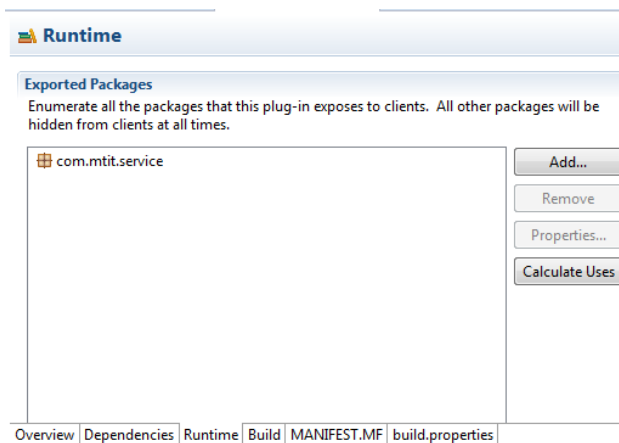
```java
public interface ServicePublish {

    public String publishService();

}
```

```java
public class ServicePublishImpl implements ServicePublish{

    @Override
    public String publishService() {
        return "Execute the publish service of ServicePublisher"
    }

}
```

3. Then implement the **ServiceActivator** and publish the service

```java
import org.osgi.framework.BundleActivator;

public class ServiceActivator implements BundleActivator {

    ServiceRegistration publishServiceRegistration;

    @Override
    public void start(BundleContext context) throws Exception {

        System.out.println("Publisher Start");
        ServicePublish publisherService = new ServicePublishImpl();
        publishServiceRegistration = context.registerService(
                ServicePublish.class.getName(), publisherService, null);
    }

    @Override
    public void stop(BundleContext context) throws Exception {

        System.out.println("Publisher Stop");
        publishServiceRegistration.unregister();
    }
}
```

```
▲ 🗁 ServicePublisher
    ⊕ Manifest: ServicePublisher
  ▲ 🗁 src
    ▲ 🗁 com.mtit.service
        ▷ 🗋 ServiceActivator.java
        ▷ 🗋 ServicePublish.java
        ▷ 🗋 ServicePublishImpl.java
  ▷ 🗁 JRE System Library [JavaSE-1.7]
  ▷ 🗁 Plug-in Dependencies
    🗁 OSGi Dependencies
  ▷ 🗁 BundleContent
```

4. Then in the **manifest.mf** file modify with adding service export details.

Runtime

Exported Packages

Enumerate all the packages that this plug-in exposes to clients. All other packages will be hidden from clients at all times.

⊞ com.mtit.service

[Add...]
[Remove]
[Properties...]
[Calculate Uses]

Overview | Dependencies | Runtime | Build | MANIFEST.MF | build.properties

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ServicePublisher
Bundle-SymbolicName: ServicePublisher
Bundle-Version: 1.0.0.qualifier
Bundle-Activator: com.mtit.service.ServiceActivato
Export-Package: com.mtit.service
Import-Package: org.osgi.framework
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
```

Service Subscriber

5. Now create new **OSGi Bunndle project => ServiceSubscriber** and implement below class. **ServiceActivator**

```java
public class ServiceActivator implements BundleActivator {

    ServiceReference serviceReference;

    @Override
    public void start(BundleContext context) throws Exception {

        System.out.println("Start Subscriber Service");
        serviceReference = context.getServiceReference(ServicePublish.class
                .getName());
        ServicePublish servicePublish = (ServicePublish) context
                .getService(serviceReference);
        System.out.println(servicePublish.publishService());
    }

    @Override
    public void stop(BundleContext context) throws Exception {

        System.out.println("Good Bye !!!");
        context.ungetService(serviceReference);
    }
}
```
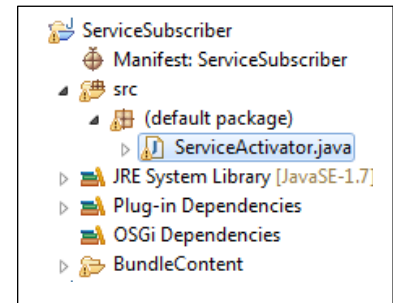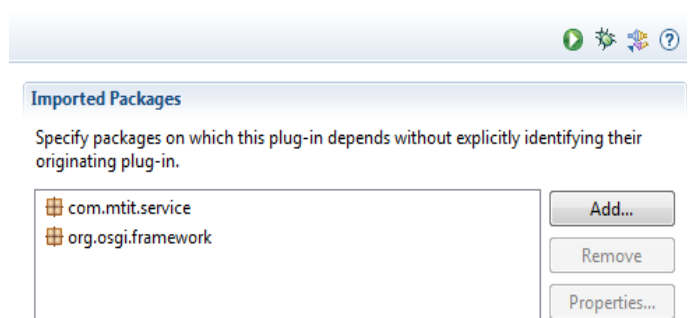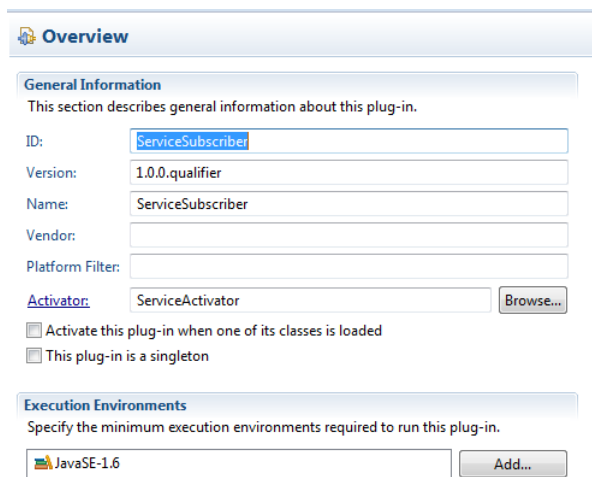
6. Subscriber is going to **subscribe** the **service published** by ServicePublisher. For that in the **Manifest** file you should modify with adding **publisher exported services**.

Subscriber Manifest

Publisher Manifest

7. Create separate Run configurations for Publisher module and Subscriber module. Then Run

```
Markers    Properties    Data Source Explorer    Snippets    Console 83    SQL Results              

MTIT Subscriber Service [OSGi Framework] C:\Program Files\Java\jdk1.7.0\bin\javaw.exe (Jan 9, 2016 5:10:41 PM)
    822|Active      |      4|ASM (3.3.1.v201105211655)
    823|Active      |      4|SAT4J Core (2.3.0.v20110329)
    824|Active      |      4|SAT4J Pseudo (2.3.0.v20110329)
    825|Active      |      4|SLF4J API (1.6.4.v20120130-2120)
    826|Active      |      4|m2e connector for the mavenarchiver and pom properties (0.16.0.201401211452)
    827|Active      |      4|UDDI4J (2.0.5.v200805270300)
    828|Active      |      4|W3C CSS SAC (1.3.1.v200903091627)
    829|Active      |      4|W3C SMIL DOM (1.0.0.v200806040011)
    830|Active      |      4|W3C SVG DOM (1.1.0.v201011041433)
    831|Active      |      4|ADT XML Overlay (22.0.5.v201307292155--757759)
    834|Active      |      4|ServicePublisher (1.0.0.qualifier)
    835|Active      |      4|ServiceSubscriber (1.0.0.qualifier)
osgi> stop 834
Publisher Stop
osgi> stop 835
Good Bye !!!
osgi> start 835
Start Subscriber Service
gogo: BundleException: Exception in ServiceActivator.start() of bundle ServiceSubscriber.
osgi> start 834
Publisher Start
osgi> start 835
Start Subscriber Service
Execute the publish service of ServicePublisher
osgi>
```

- Stop both and then start publisher first and then start subscriber
- Try to stop publisher and start subscriber

The End