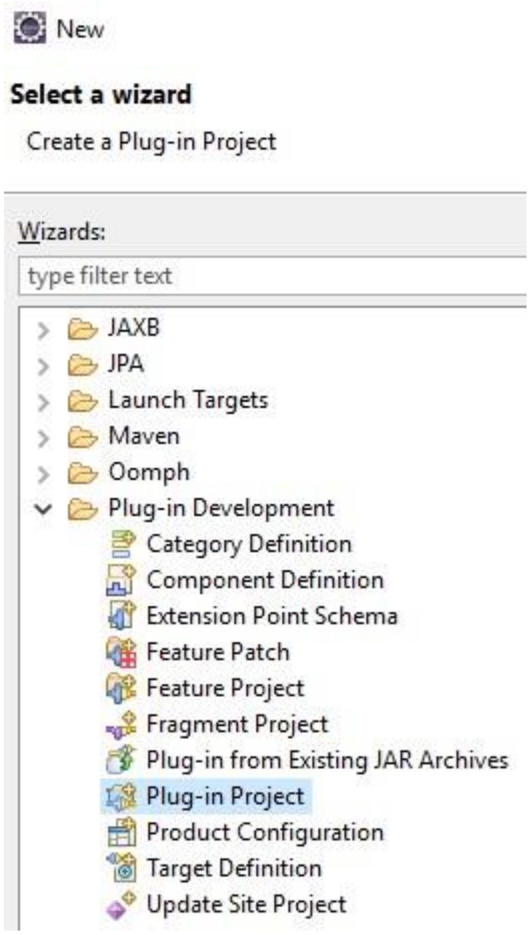
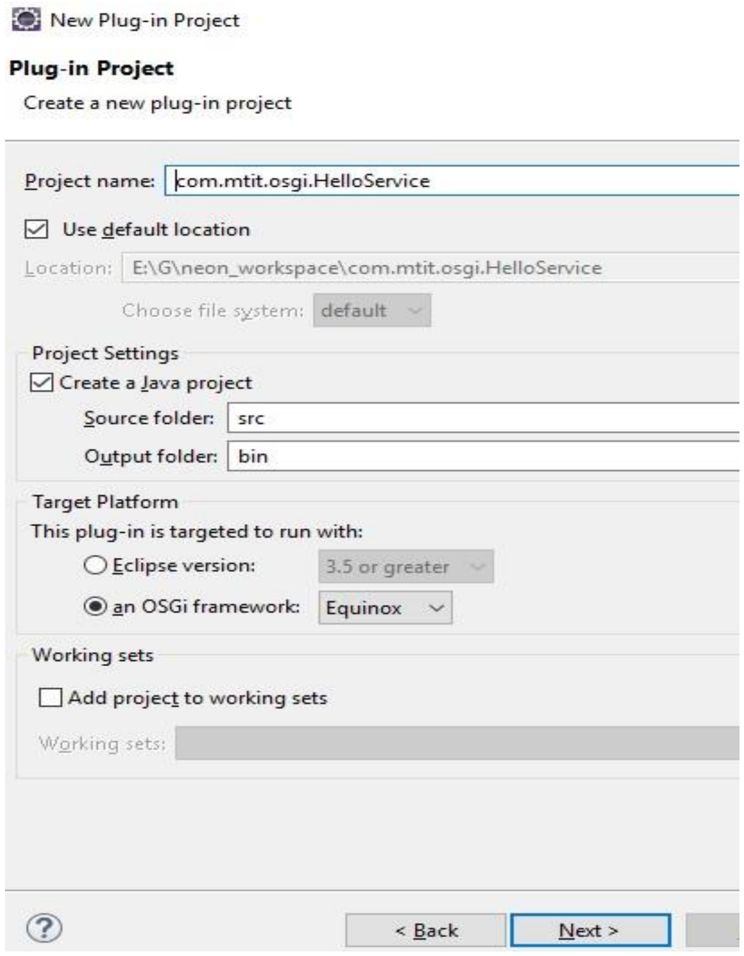
Lab 02 - OSGi-Eclipse Equinox

Task 01 Create OSGi bundle project

File => new => Plugin Development => Plug-in Project => next





Create a new plug-in project

Project name: com.mtit.osgi.HelloService

☑ Use default location

Location: E:\G\neon_workspace\com.mtit.osgi.HelloService

Choose file system: default

Project Settings

☑ Create a Java project

Source folder: src

Output folder: bin

Target Platform

This plug-in is targeted to run with:

○ Eclipse version: 3.5 or greater

● an OSGi framework: Equinox

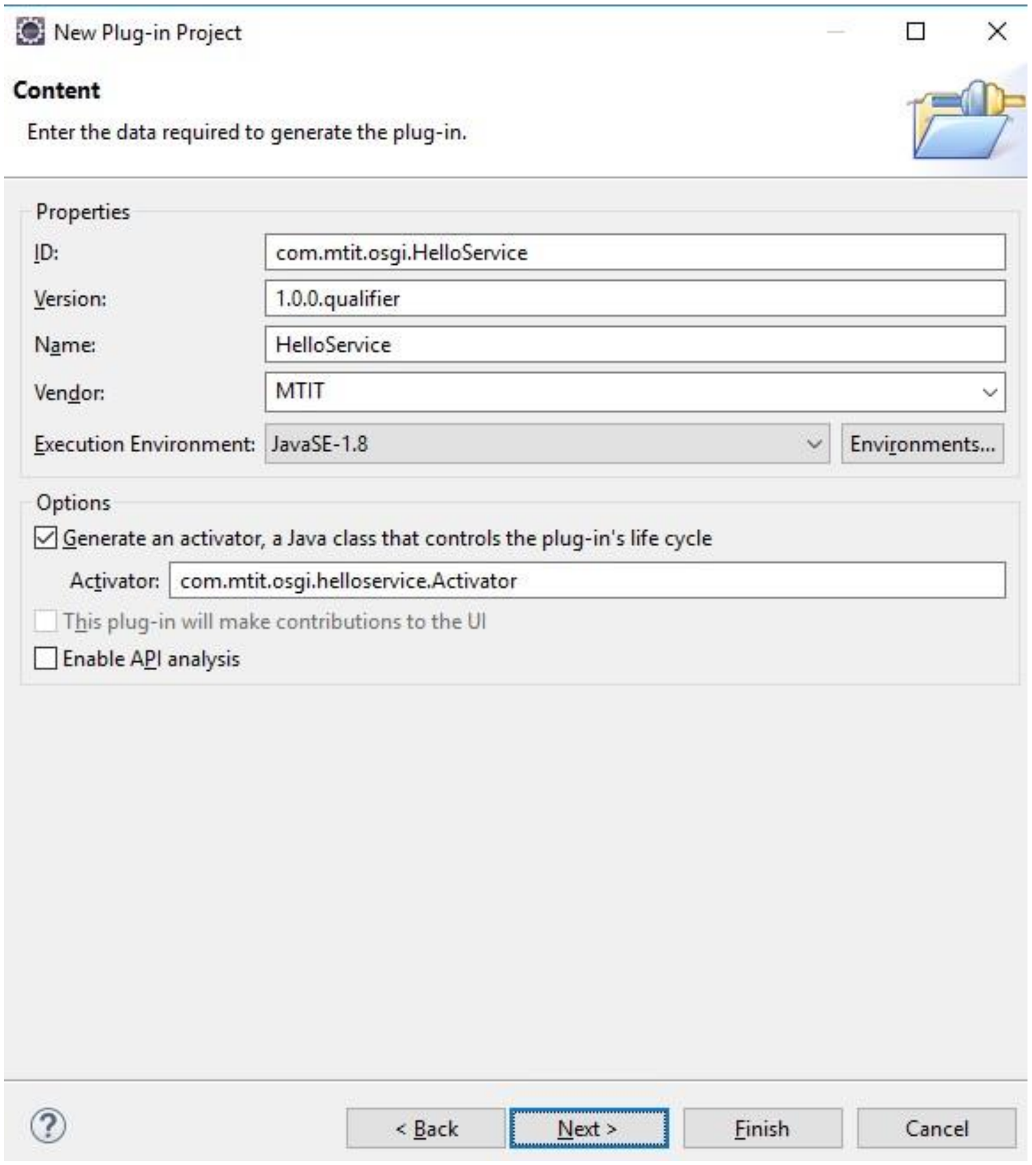
Working sets

☐ Add project to working sets

Working sets:

Create the project name = **com.mtit.osgi.HelloService**

Select generate **Activator** class as well and select **next** from following wizard



Content

Enter the data required to generate the plug-in.

Properties

ID: com.mtit.osgi.HelloService

Version: 1.0.0.qualifier

Name: HelloService

Vendor: MTIT

Execution Environment: JavaSE-1.8

Options

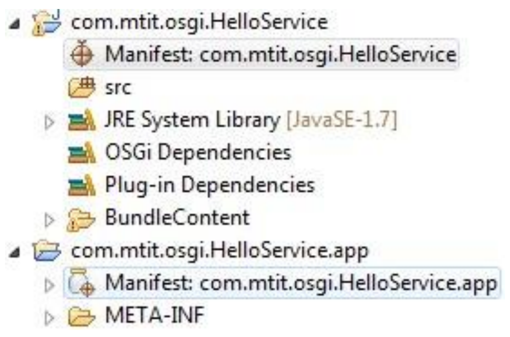
[✓] Generate an activator, a Java class that controls the plug-in's life cycle

Activator: com.mtit.osgi.helloservice.Activator

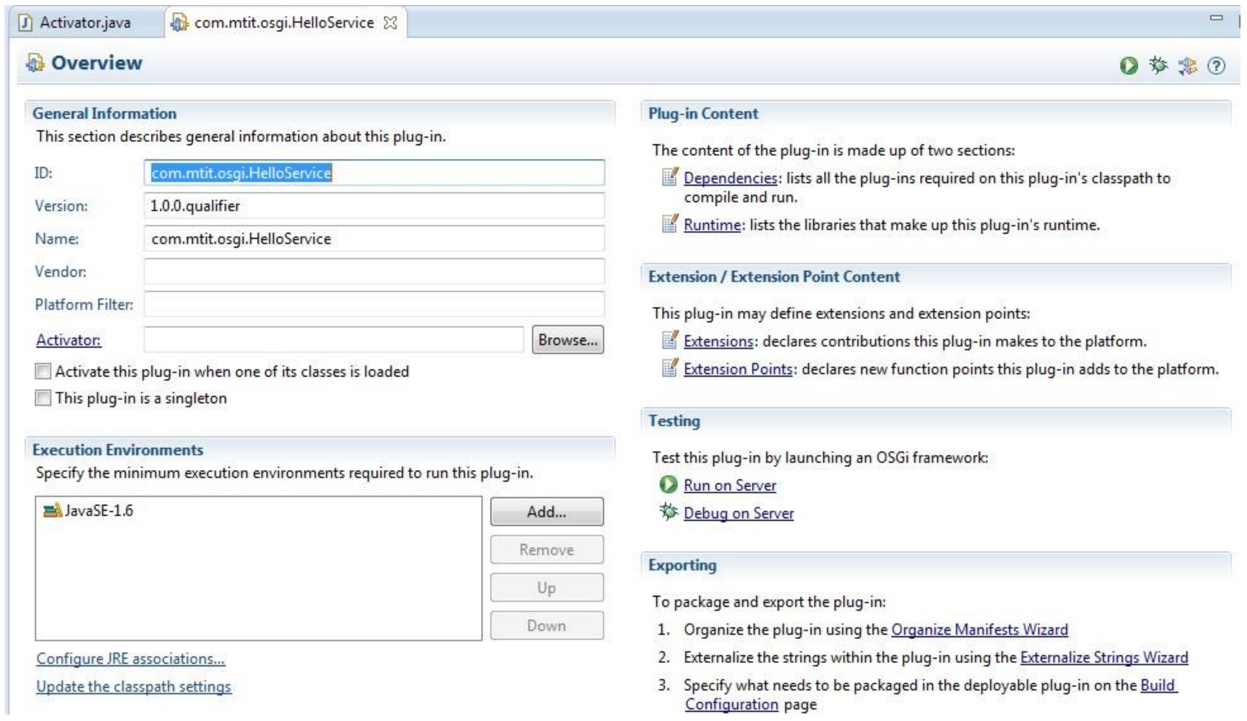
[ ] This plug-in will make contributions to the UI

[ ] Enable API analysis

Select next => Select your project => Finish


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

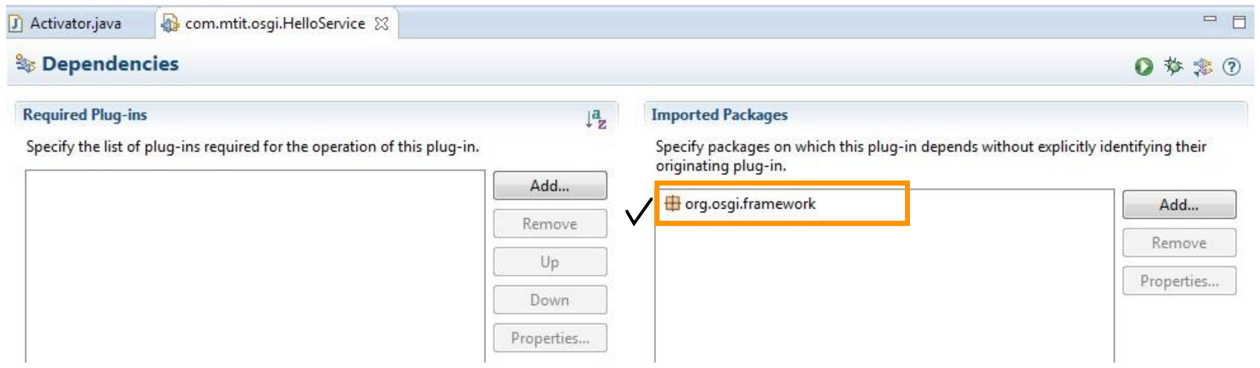
Plug-in Content
The content of the plug-in is made up of two sections:
Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.
Runtime: lists the libraries that make up this plug-in's runtime.

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.

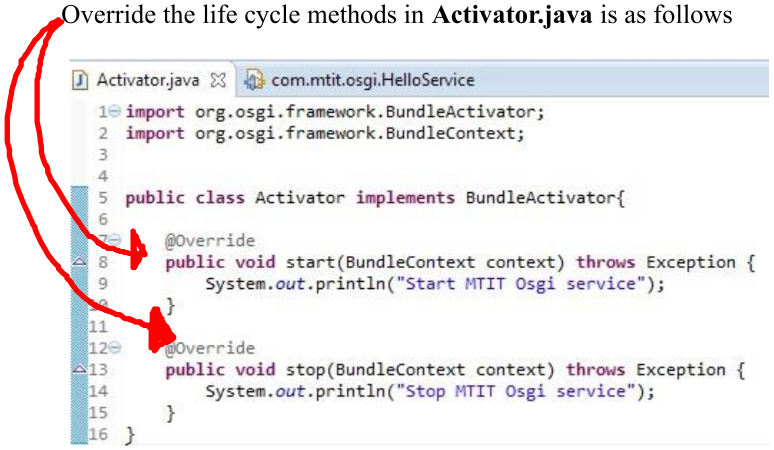

Activator.java | com.mtit.osgi.HelloService

Dependencies

Required Plug-ins
Specify the list of plug-ins required for the operation of this plug-in.
Add... Remove Up Down Properties...

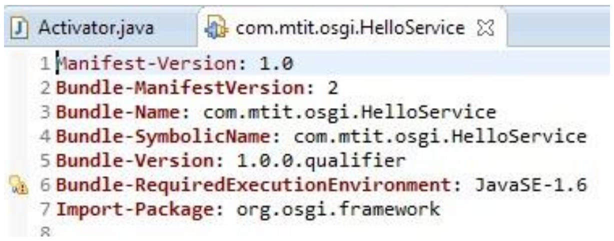
Imported Packages
Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.
✓ org.osgi.framework
Add... Remove Properties...

Override the life cycle methods in **Activator.java** is as follows



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


public class Activator implements BundleActivator{

    @Override
    public void start(BundleContext context) throws Exception {
        System.out.println("Start MTIT Osgi service");
    }

    @Override
    public void stop(BundleContext context) throws Exception {
        System.out.println("Stop MTIT Osgi service");
    }
}
```

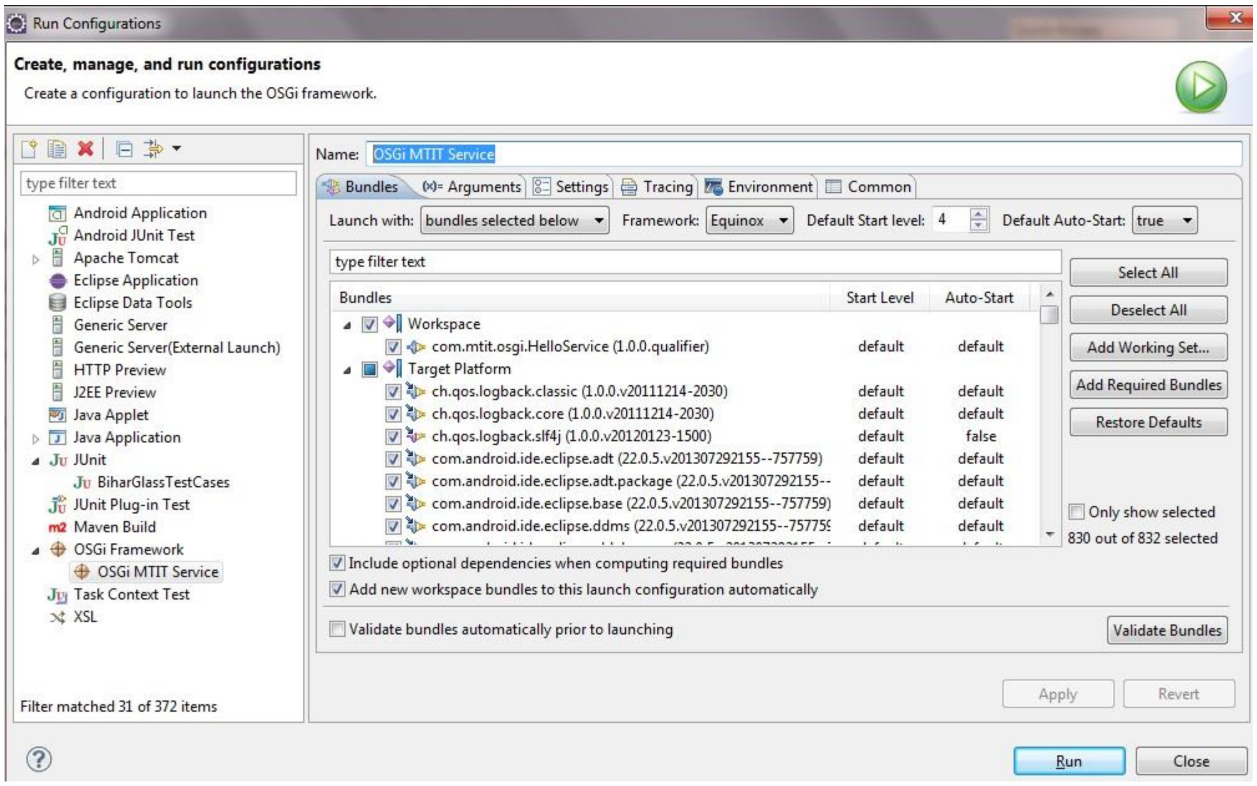
Manifest.mf should be as follows.



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

Run Configuration

Select the **manifest** and **run as => run configuration**. Then modify the name **OSGi MTIT Service**



Create, manage, and run configurations

Create a configuration to launch the OSGi framework.

Name: OSGi MTIT Service

Launch with: bundles selected below Framework: Equinox Default Start level: 4 Default Auto-Start: true

type filter text

```
Bundles                                                        Start Level   Auto-Start
Workspace
    com.mtit.osgi.HelloService (1.0.0.qualifier)                 default       default
Target Platform
    ch.qos.logback.classic (1.0.0.v20111214-2030)                default       default
    ch.qos.logback.core (1.0.0.v20111214-2030)                   default       default
    ch.qos.logback.slf4j (1.0.0.v20120123-1500)                  default       false
    com.android.ide.eclipse.adt (22.0.5.v201307292155--757759)   default       default
    com.android.ide.eclipse.adt.package (22.0.5.v201307292155--  default       default
    com.android.ide.eclipse.base (22.0.5.v201307292155--757759)  default       default
    com.android.ide.eclipse.ddms (22.0.5.v201307292155--75775    default       default
```

☐ Only show selected

830 out of 832 selected

☑ Include optional dependencies when computing required bundles

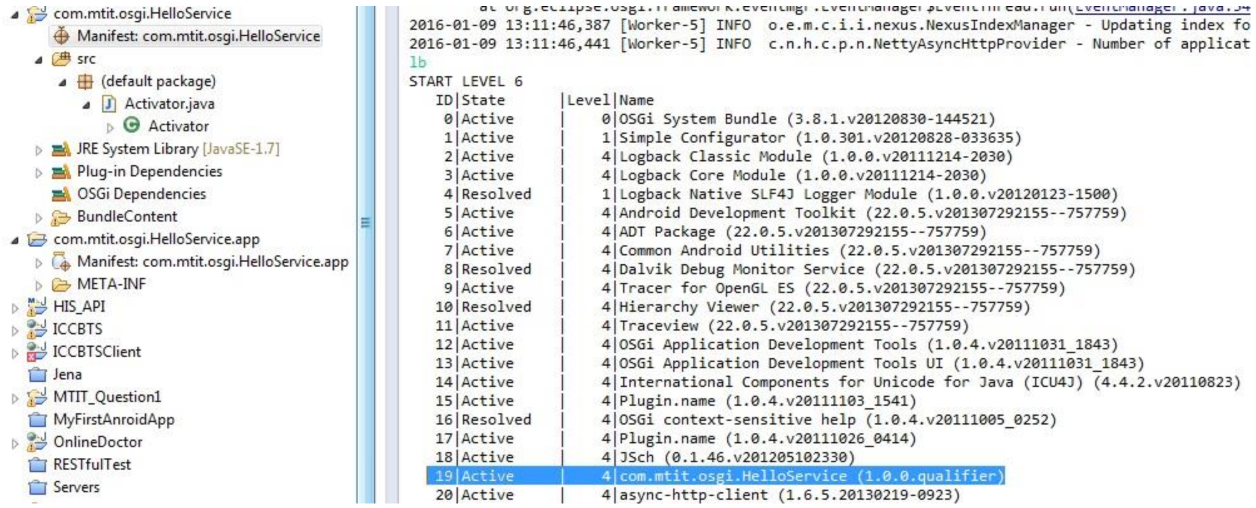
☑ Add new workspace bundles to this launch configuration automatically

☐ Validate bundles automatically prior to launching

type filter text

Filter matched 31 of 372 items

- 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>**



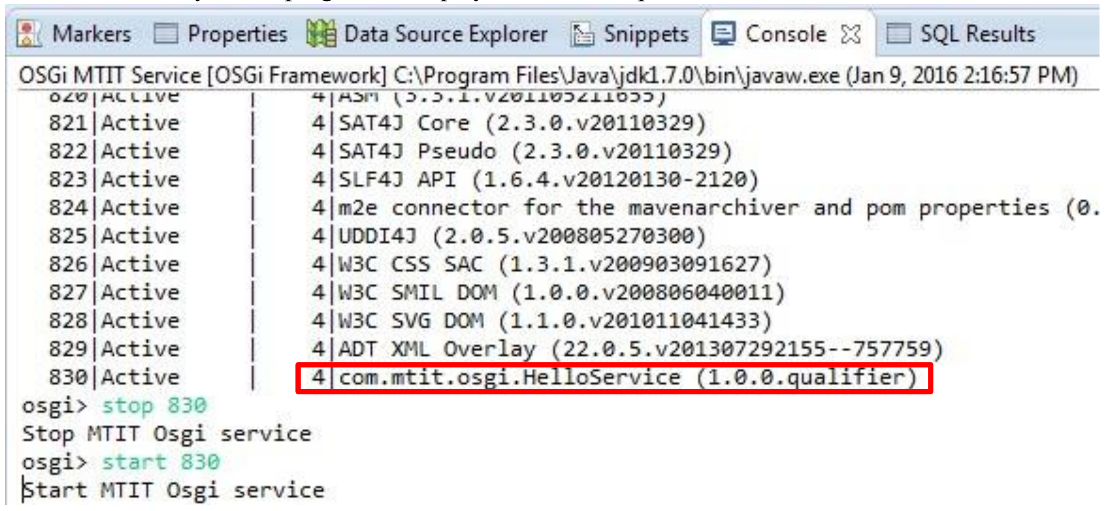
```
com.mtit.osgi.HelloService
  Manifest: com.mtit.osgi.HelloService
  src
    (default package)
      Activator.java
        Activator
  JRE System Library [JavaSE-1.7]
  Plug-in Dependencies
  OSGi Dependencies
  BundleContent
com.mtit.osgi.HelloService.app
  Manifest: com.mtit.osgi.HelloService.app
  META-INF
HIS_API
ICCBTS
ICCBTSClient
Jena
MTIT_Question1
MyFirstAnroidApp
OnlineDoctor
RESTfulTest
Servers
```

```
        at org.eclipse.osgi.framework.eventmgr.EventManager$EventThread.run(EventManager.java:34
2016-01-09 13:11:46,387 [Worker-5] INFO  o.e.m.c.i.i.nexus.NexusIndexManager - Updating index fo
2016-01-09 13:11:46,441 [Worker-5] INFO  c.n.h.c.p.n.NettyAsyncHttpProvider - Number of applicat
lb
START LEVEL 6
   ID|State      |Level|Name
    0|Active     |    0|OSGi System Bundle (3.8.1.v20120830-144521)
    1|Active     |    1|Simple Configurator (1.0.301.v20120828-033635)
    2|Active     |    4|Logback Classic Module (1.0.0.v20111214-2030)
    3|Active     |    4|Logback Core Module (1.0.0.v20111214-2030)
    4|Resolved   |    1|Logback Native SLF4J Logger Module (1.0.0.v20120123-1500)
    5|Active     |    4|Android Development Toolkit (22.0.5.v201307292155--757759)
    6|Active     |    4|ADT Package (22.0.5.v201307292155--757759)
    7|Active     |    4|Common Android Utilities (22.0.5.v201307292155--757759)
    8|Resolved   |    4|Dalvik Debug Monitor Service (22.0.5.v201307292155--757759)
    9|Active     |    4|Tracer for OpenGL ES (22.0.5.v201307292155--757759)
   10|Resolved   |    4|Hierarchy Viewer (22.0.5.v201307292155--757759)
   11|Active     |    4|Traceview (22.0.5.v201307292155--757759)
   12|Active     |    4|OSGi Application Development Tools (1.0.4.v20111031_1843)
   13|Active     |    4|OSGi Application Development Tools UI (1.0.4.v20111031_1843)
   14|Active     |    4|International Components for Unicode for Java (ICU4J) (4.4.2.v20110823)
   15|Active     |    4|Plugin.name (1.0.4.v20111103_1541)
   16|Resolved   |    4|OSGi context-sensitive help (1.0.4.v20111005_0252)
   17|Active     |    4|Plugin.name (1.0.4.v20111026_0414)
   18|Active     |    4|JSch (0.1.46.v201205102330)
   19|Active     |    4|com.mtit.osgi.HelloService (1.0.0.qualifier)
   20|Active     |    4|async-http-client (1.6.5.20130219-0923)
```

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

Troubleshoot the application

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



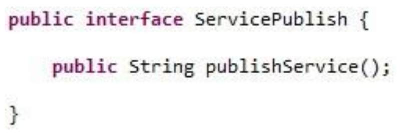
```
OSGi MTIT Service [OSGi Framework] C:\Program Files\Java\jdk1.7.0\bin\javaw.exe (Jan 9, 2016 2:16:57 PM)
  820|Active     |    4|ASM (3.3.1.v201105211655)
  821|Active     |    4|SAT4J Core (2.3.0.v20110329)
  822|Active     |    4|SAT4J Pseudo (2.3.0.v20110329)
  823|Active     |    4|SLF4J API (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
```

<u>**Task 02 OSGi -Dependency Management (Publisher and Subscriber)**</u>

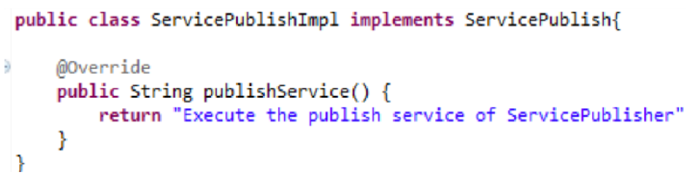
<u>**Service Publisher**</u>

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**.

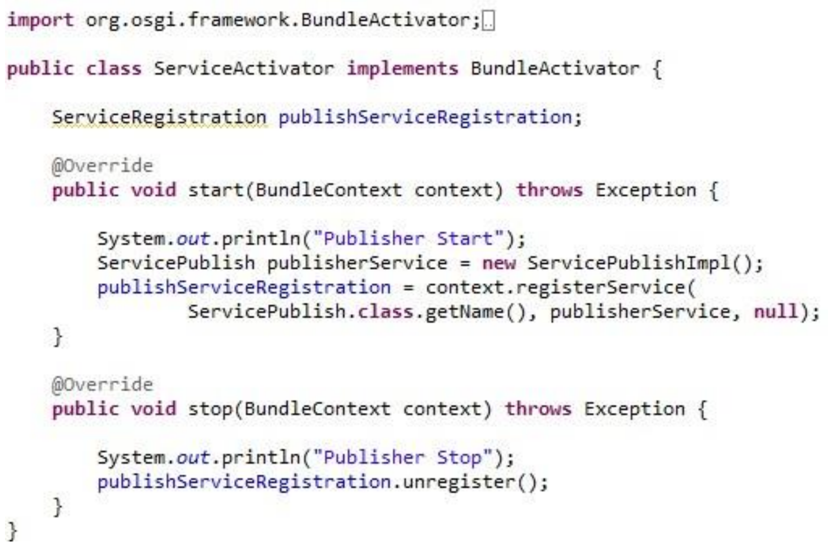


```
public interface ServicePublish {

    public String publishService();

}
```

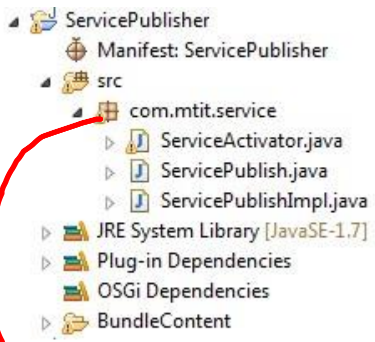


```
public class ServicePublishImpl implements ServicePublish{

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

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

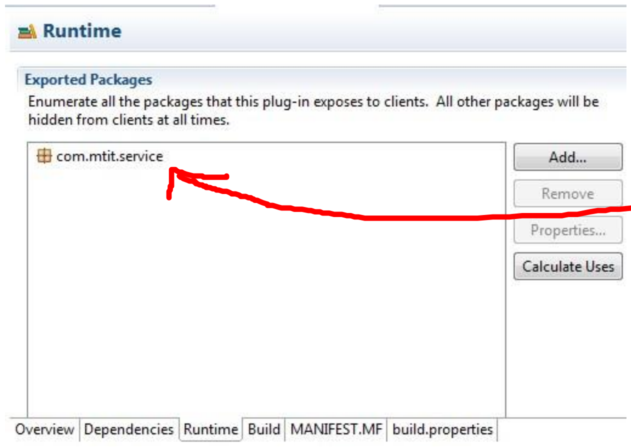


```
import org.osgi.framework.BundleActivator;

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.



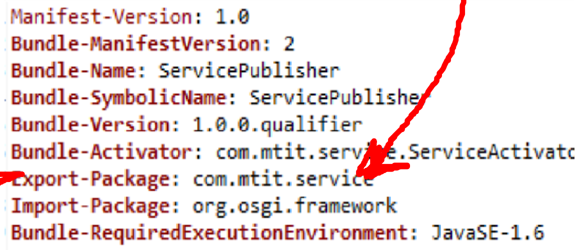
Runtime

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

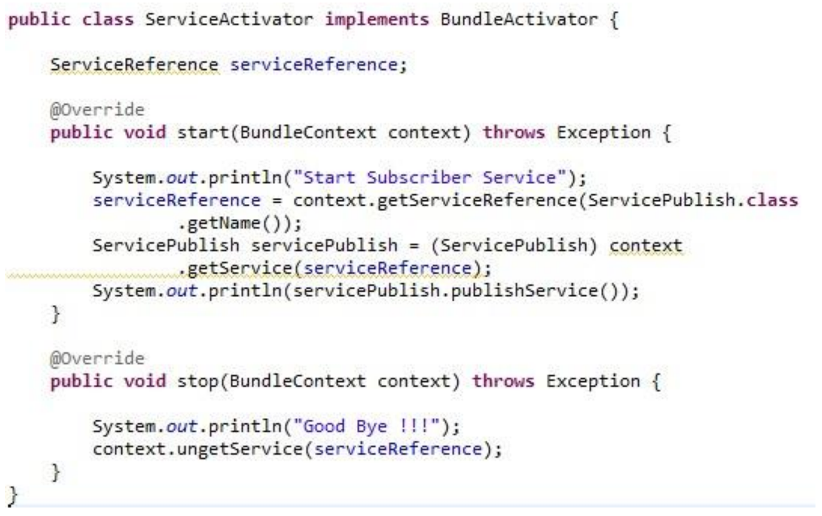
Add...
Remove
Properties...
Calculate Uses
Overview | Dependencies | Runtime | Build | MANIFEST.MF | build.properties



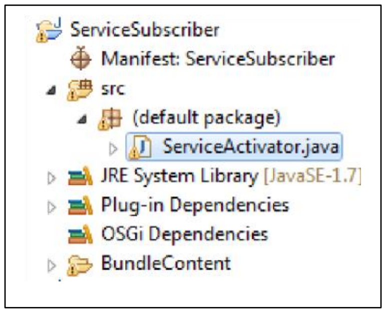
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ServicePublisher
Bundle-SymbolicName: ServicePublishe
Bundle-Version: 1.0.0.qualifier
Bundle-Activator: com.mtit.service.ServiceActivatc
Export-Package: com.mtit.service
Import-Package: org.osgi.framework
Bundle-RequiredExecutionEnvironment: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

Service Subscriber

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

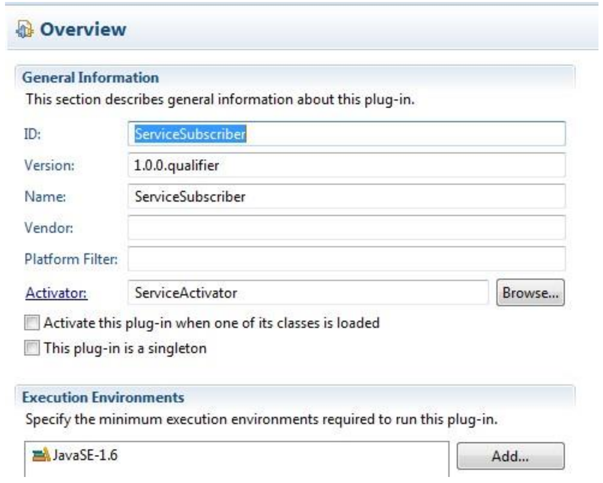


```
public class ServiceActivator implements BundleActivator {

    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);
    }
}
```



```
ServiceSubscriber
    Manifest: ServiceSubscriber
    src
        (default package)
            ServiceActivator.java
    JRE System Library [JavaSE-1.7]
    Plug-in Dependencies
    OSGi Dependencies
    BundleContent
```

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**.



Overview

General Information

This section describes general information about this plug-in.

ID:	ServiceSubscriber
Version:	1.0.0.qualifier
Name:	ServiceSubscriber
Vendor:	
Platform Filter:	
Activator:	ServiceActivator

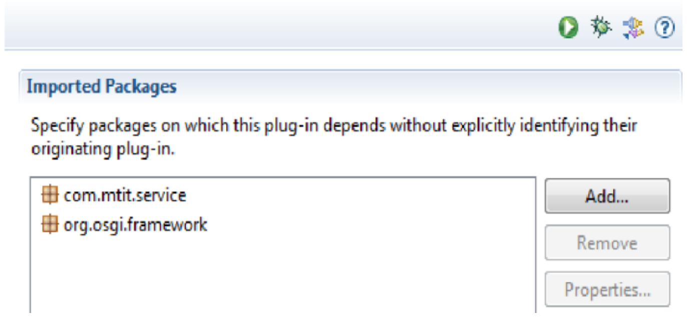
☐ Activate this plug-in when one of its classes is loaded

☐ This plug-in is a singleton

Execution Environments

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

JavaSE-1.6

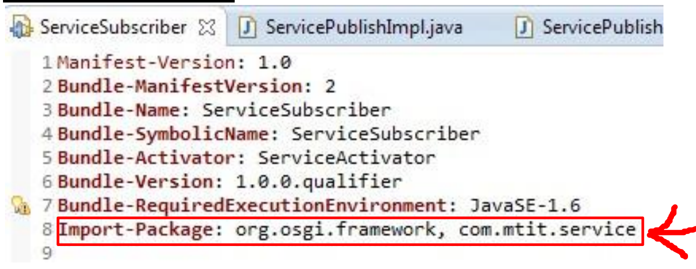


Imported Packages

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

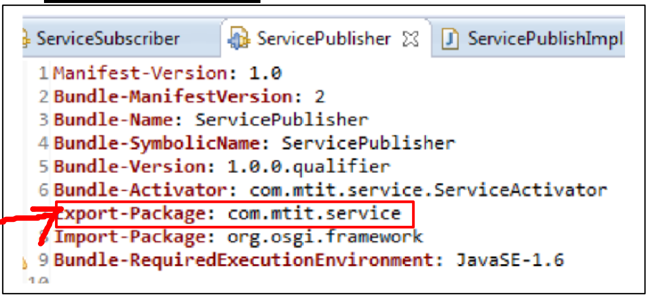
- com.mtit.service
- org.osgi.framework

Subscriber Manifest



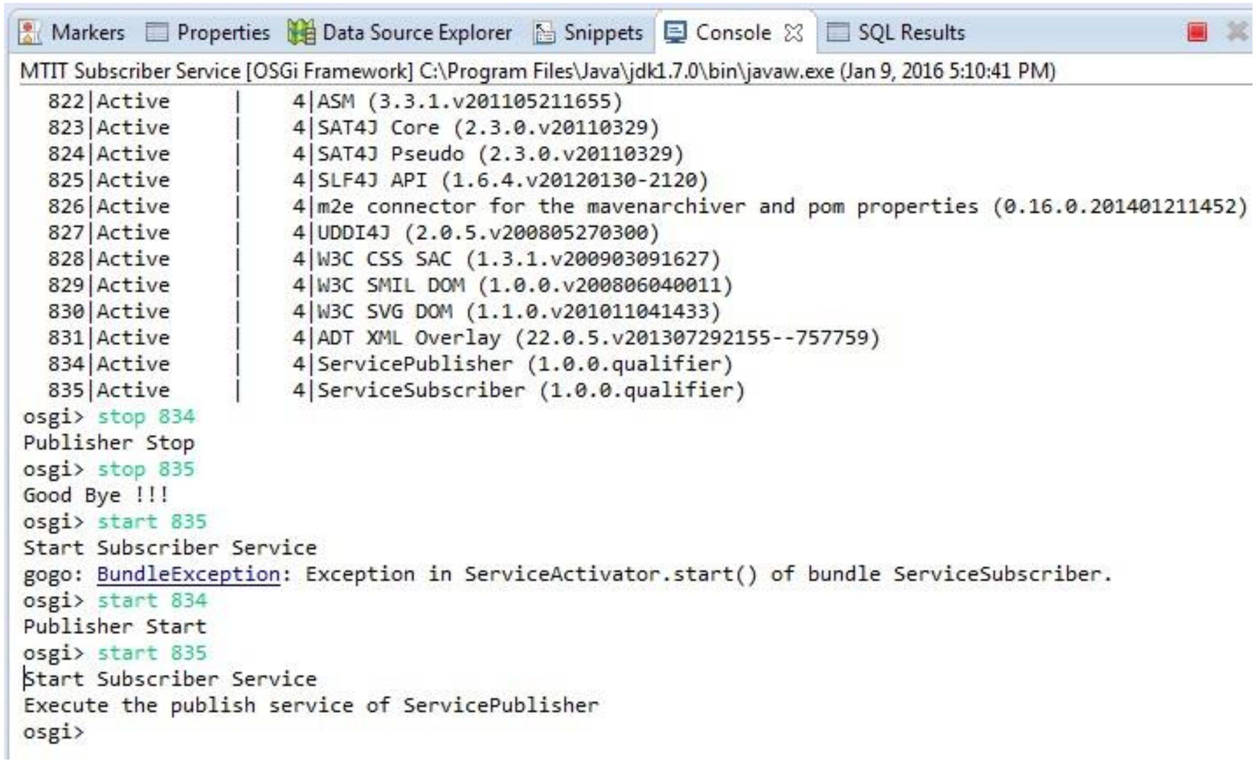
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ServiceSubscriber
Bundle-SymbolicName: ServiceSubscriber
Bundle-Activator: ServiceActivator
Bundle-Version: 1.0.0.qualifier
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Import-Package: org.osgi.framework, com.mtit.service
```

Publisher Manifest



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

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



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