

## Eclipse RCP Part II: Bundles and Plugins

Automotive Financial Services Insurance Life Science & Healthcare Public Sector  
Telecommunications & Media Travel & Logistics Utilities Automotive Financial  
Services Insurance Life Science & Healthcare Public Sector Telecommunications  
& Media Travel & Logistics Utilities Automotive Financial Services Insurance  
Life Science & Healthcare Public Sector Telecommunications & Media Travel  
& Logistics Utilities Automotive Financial Services Insurance Life Science  
Healthcare Public Sector Telecommunications & Media Travel & Logistics  
Utilities Automotive Financial Services Life Science & Healthcare Public  
Sector Telecommunications & Media Travel & Logistics Utilities Automotive  
Financial Services Insurance Life Science & Healthcare Public Sector  
Telecommunications & Media Travel & Logistics Utilities Automotive  
Financial Services Insurance Life Science & Healthcare Telecommunications  
& Media Travel & Logistics Utilities Automotive Financial Services Insurance  
Life Science & Healthcare Public Sector Telecommunications & Media Travel &  
Logistics Utilities Automotive Financial Services Insurance Life Science &  
Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities  
Automotive Financial Services Insurance Life Science & Healthcare Public Sector



.consulting .solutions .partnership

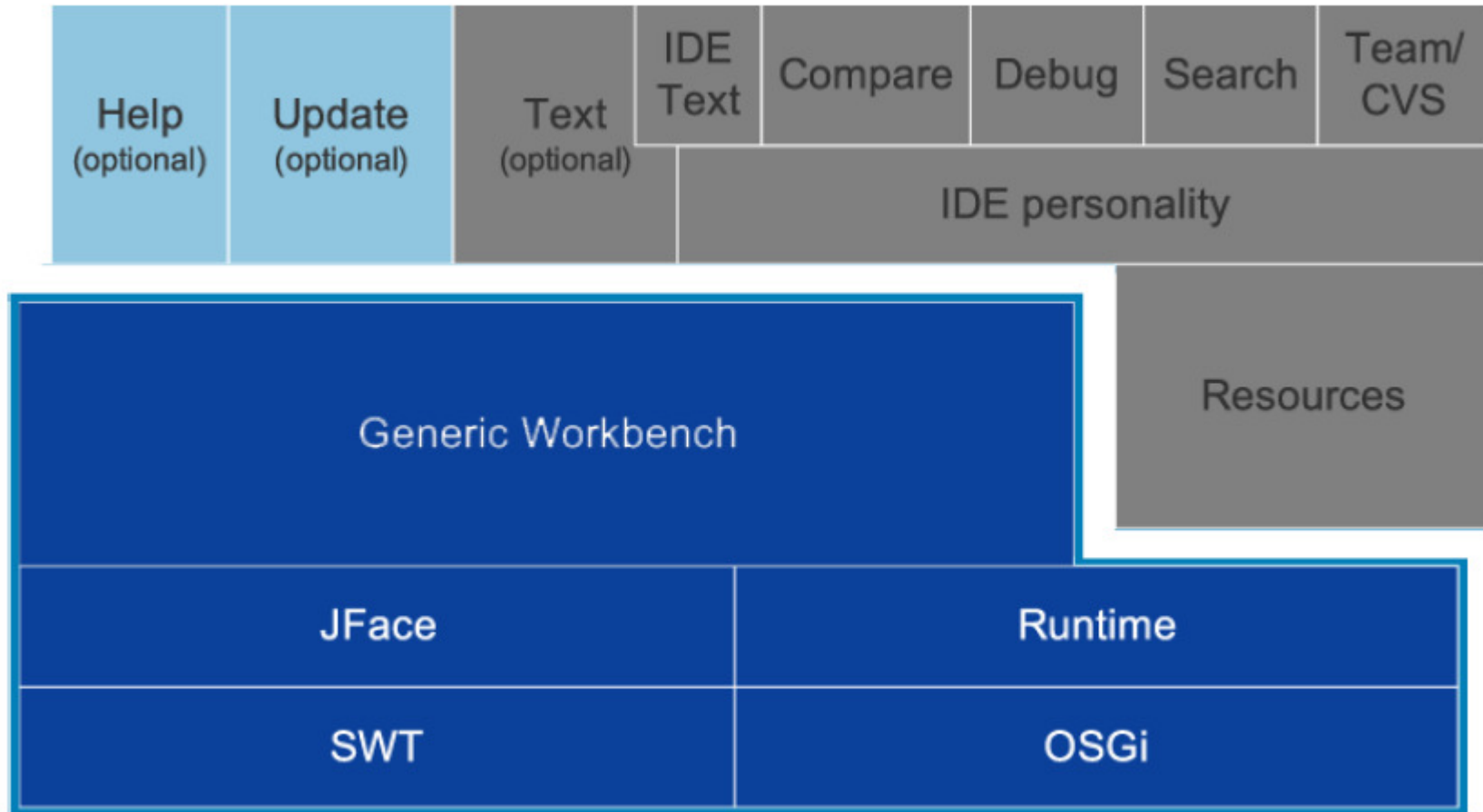


## Objective



- Understanding bundles
- Understanding plug-in dependencies
- How to use the run configuration
- How to use the target platform
- What's the language delta pack
- Understanding the content of the manifest.mf file
  - Lazy loading
  - The Activator class







































## The Architecture



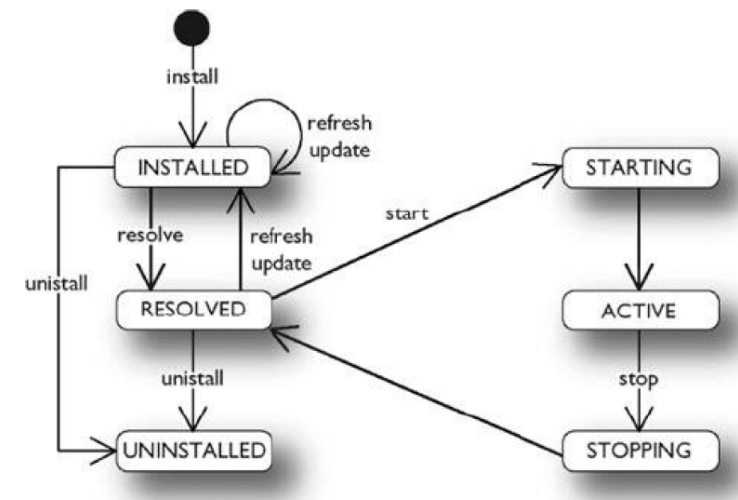
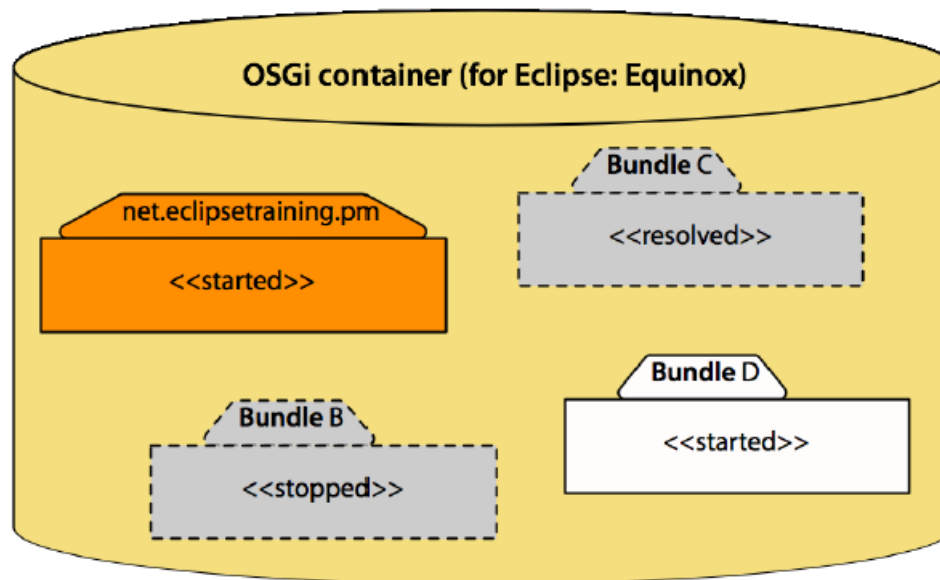
- OSGi is a specification for a service platform defined by the OSGi Alliance (formerly known as the Open Services Gateway initiative)
- Service platform specification
  - service architecture
  - dynamic install/uninstall/update
  - security
  - logging
  - remote configuration API
  - declarative services
  - and many more...



- There are several Open-Source OSGi implementations:
  - Equinox
  - Apache Felix
  - Knoplerfish OSGi
- Equinox is an implementation of OSGi used by Eclipse

 org.eclipse.equinox.p2.exemplarysetup.source\_1.0.0.v20080427-2136.jar  
 org.eclipse.equinox.p2.exemplarysetup\_1.0.0.v20080427-2136.jar  
 org.eclipse.equinox.p2.extensionlocation.source\_1.0.2.R34x\_v20080825.jar  
 org.eclipse.equinox.p2.extensionlocation\_1.0.2.R34x\_v20080825.jar  
 org.eclipse.equinox.p2.garbagecollector.source\_1.0.1.R34x\_v20080818.jar  
 org.eclipse.equinox.p2.garbagecollector\_1.0.1.R34x\_v20080818.jar  
 org.eclipse.equinox.p2.jarprocessor.source\_1.0.0.v20080514-1900.jar  
 org.eclipse.equinox.p2.jarprocessor\_1.0.0.v20080514-1900.jar  
 org.eclipse.equinox.p2.metadata.generator.source\_1.0.1.R34x\_v20080819.jar  
 org.eclipse.equinox.p2.metadata.generator\_1.0.1.R34x\_v20080819.jar  
 org.eclipse.equinox.p2.metadata.repository.source\_1.0.0.v20080604.jar  
 org.eclipse.equinox.p2.metadata.repository\_1.0.0.v20080604.jar  
 org.eclipse.equinox.p2.metadata.source\_1.0.0.v20080514-1900.jar  
 org.eclipse.equinox.p2.metadata\_1.0.0.v20080514-1900.jar  
 org.eclipse.equinox.p2.reconciler.dropins.source\_1.0.2.R34x\_v20080909.jar  
 org.eclipse.equinox.p2.reconciler.dropins\_1.0.2.R34x\_v20080909.jar  
 org.eclipse.equinox.p2.touchpoint.eclipse.source\_1.0.2.R34x\_v20080910.jar  
 org.eclipse.equinox.p2.touchpoint.eclipse\_1.0.2.R34x\_v20080910.jar  
 org.eclipse.equinox.p2.touchpoint.natives.source\_1.0.0.v20080505-1850.jar  
 org.eclipse.equinox.p2.touchpoint.natives\_1.0.0.v20080505-1850.jar  
 org.eclipse.equinox.p2.ui.sdk.source\_1.0.1.R34x\_v20080818.jar  
 org.eclipse.equinox.p2.ui.sdk\_1.0.1.R34x\_v20080818.jar  
 org.eclipse.equinox.p2.ui.source\_1.0.1.R34x\_v20080909.jar  
 org.eclipse.equinox.p2.ui\_1.0.1.R34x\_v20080909.jar  
 org.eclipse.equinox.p2.updatechecker.source\_1.0.0.v20080427-2136.jar  
 org.eclipse.equinox.p2.updatechecker\_1.0.0.v20080427-2136.jar  
 org.eclipse.equinox.p2.updatechecker\_1.0.1.R34x\_v20080808-1156.jar  
 org.eclipse.equinox.p2.updatechecker\_1.0.1.R34x\_v20080808-1156.jar  
 org.eclipse.equinox.preferences.source\_3.2.201.R34x\_v20080709.jar  
 org.eclipse.equinox.preferences\_3.2.201.R34x\_v20080709.jar  
 org.eclipse.equinox.registry.source\_3.4.0.v20080516-0950.jar  
 org.eclipse.equinox.registry\_3.4.0.v20080516-0950.jar  
 org.eclipse.equinox.security.source\_1.0.1.R34x\_v20080721.jar  
 org.eclipse.equinox.security.ui.source\_1.0.0.v20080603-1810.jar  
 org.eclipse.equinox.security.ui\_1.0.0.v20080603-1810.jar  
 org.eclipse.equinox.security.win32.x86.source\_1.0.0.v20080529-1600.jar  
 org.eclipse.equinox.security.win32.x86\_1.0.0.v20080529-1600.jar  
 org.eclipse.equinox.security\_1.0.1.R34x\_v20080721.jar

- The Eclipse runtime uses Equinox as a framework for its plug-in concept
- Plug-ins are loaded and started in an OSGi container:

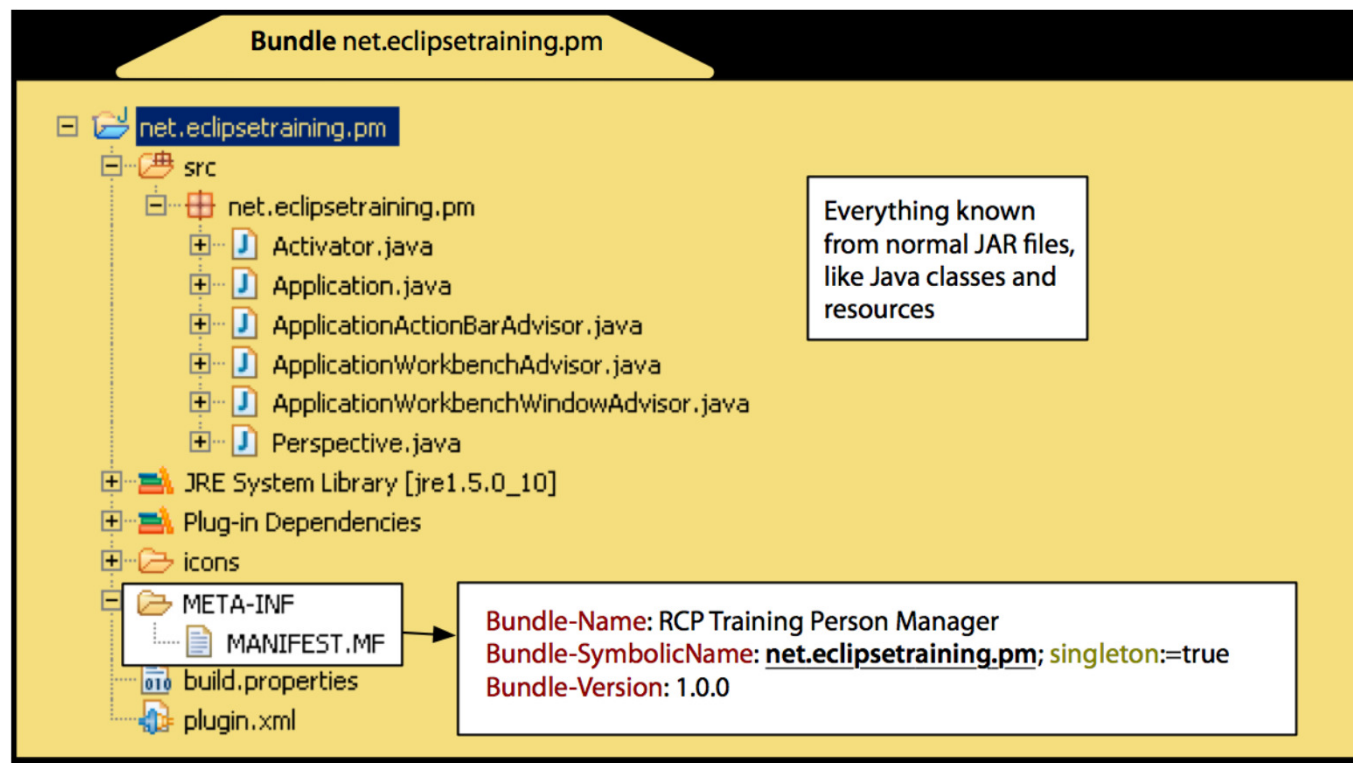


- Plug-ins are the components of an Eclipse application
- Every plug-in is a bundle in terms of OSGi
- Plug-in (Eclipse)  $\equiv$  bundle (OSGi)
- An RCP application is a set of interdependent plug-ins (RCP target plug-ins + your plug-ins)



## What is a plug-in

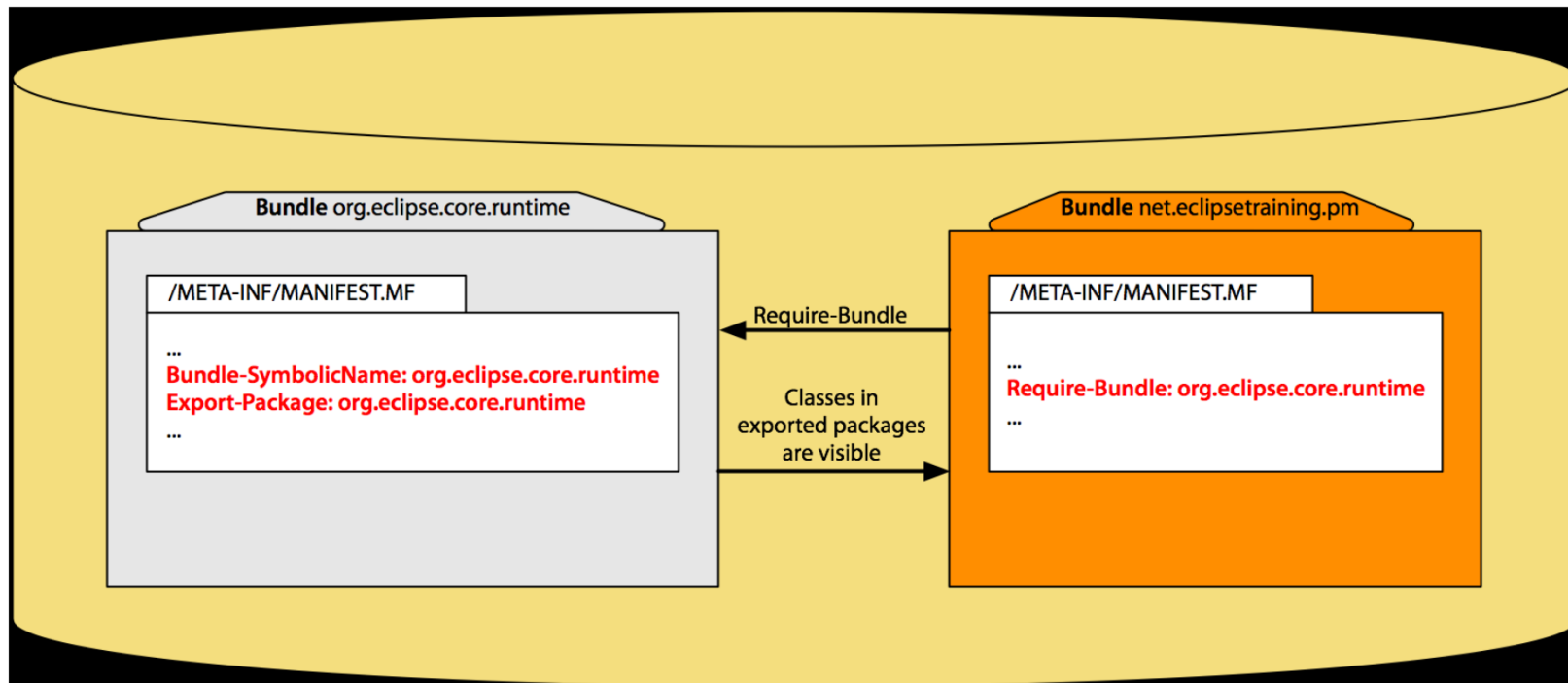
- We have created a plug-in that runs as a bundle in the OSGi Container
- Ok, so what's an Eclipse plug-in?





- The plug-in ID is the unique identification of a plug-in
- There is a convention to use the same name for:
  - Plug-in ID
  - Eclipse Project where your plug-in is hosted
  - Top-level package
- The Eclipse "New Plug-in Project" wizard is aware of this convention

- Plug-ins may declare dependencies on other plug-ins
- The dependencies are resolved by OSGi:



- Every plug-in may export some of its Java packages to other plugins
- These packages must be enumerated with Export-Package in the bundle manifest
- A plug-in requiring any of those packages must mention the exporting plug-in's bundle ID in its Require-Bundle list

### plugin com.msg.client

*import com.msg.server.core.ISample*

MANIFEST.MF

....

Required Bundle: com.msg.server

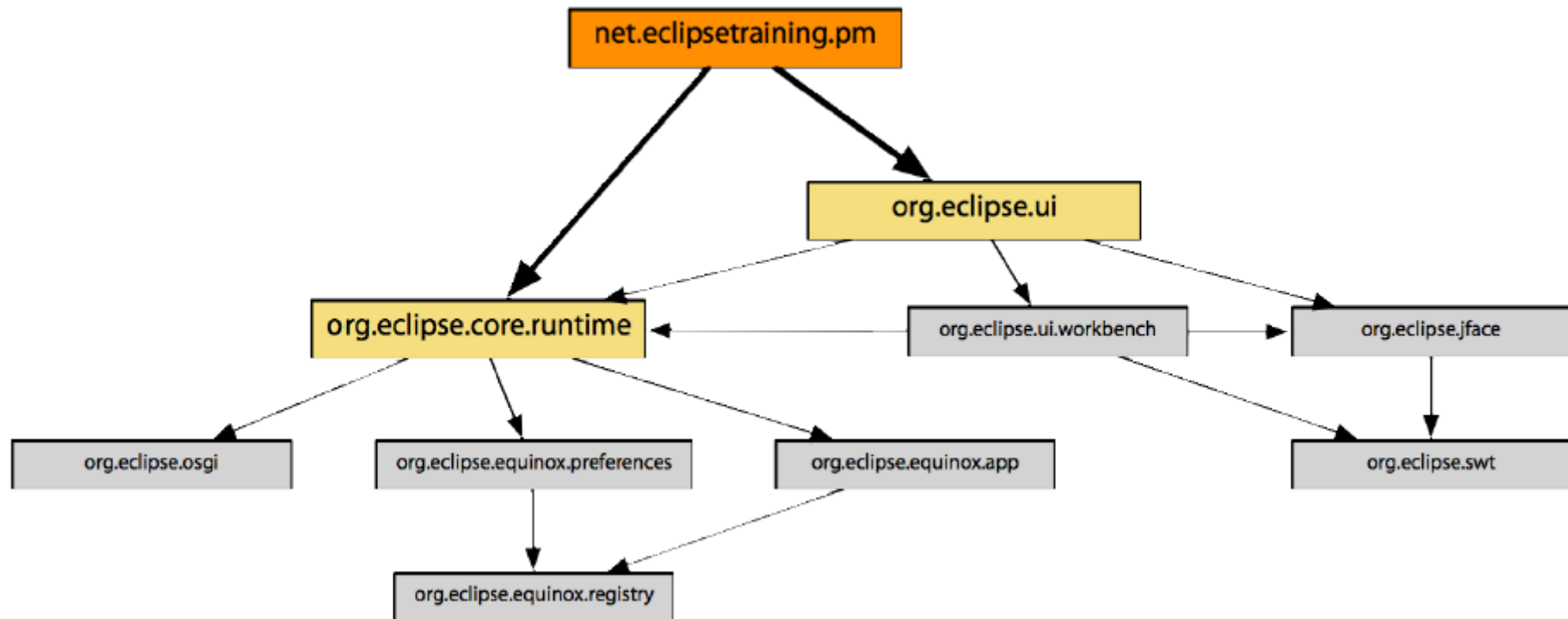
### plugin com.msg.server

MANIFEST.MF

...

Export-Package: com.msg.server.core

## Dependencies



## Dependencies

### Required Plug-ins

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

org.eclipse.ui  
 org.eclipse.core.runtime

Add...  
Remove  
Up  
Down  
Properties...

Total: 2

### Imported Package

Specify packages or

### Automated Management of Dependencies

Overview

Dependencies

Runtime

Extensions

Extension Points

Build

MANIFEST.MF

plugin.xml

bu

```

Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Ttt Plug-in
Bundle-SymbolicName: ttt; singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator: ttt.Activator
Require-Bundle: org.eclipse.ui,
org.eclipse.core.runtime
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment: JavaSE-1.6

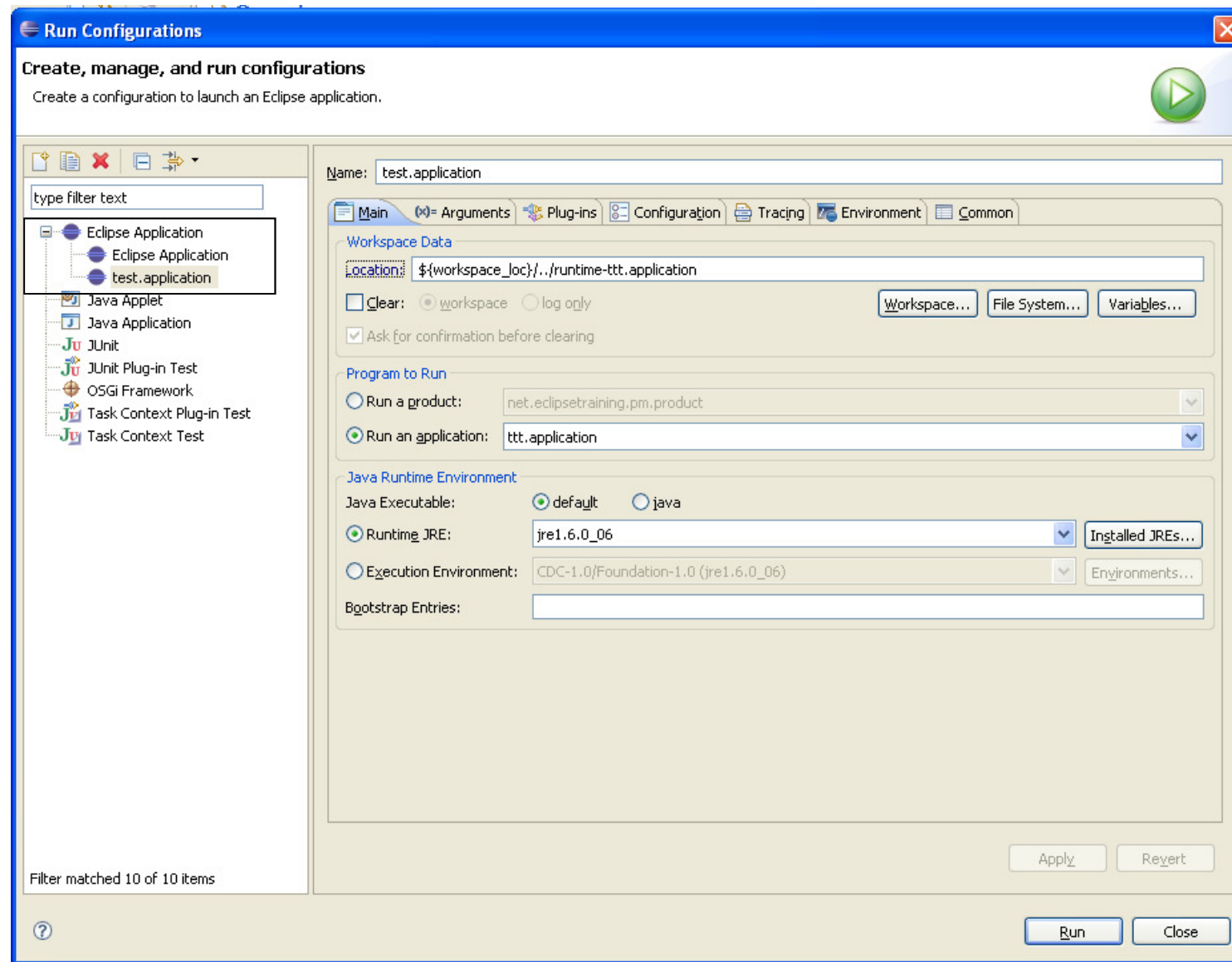
```

## what happens with the dependencies



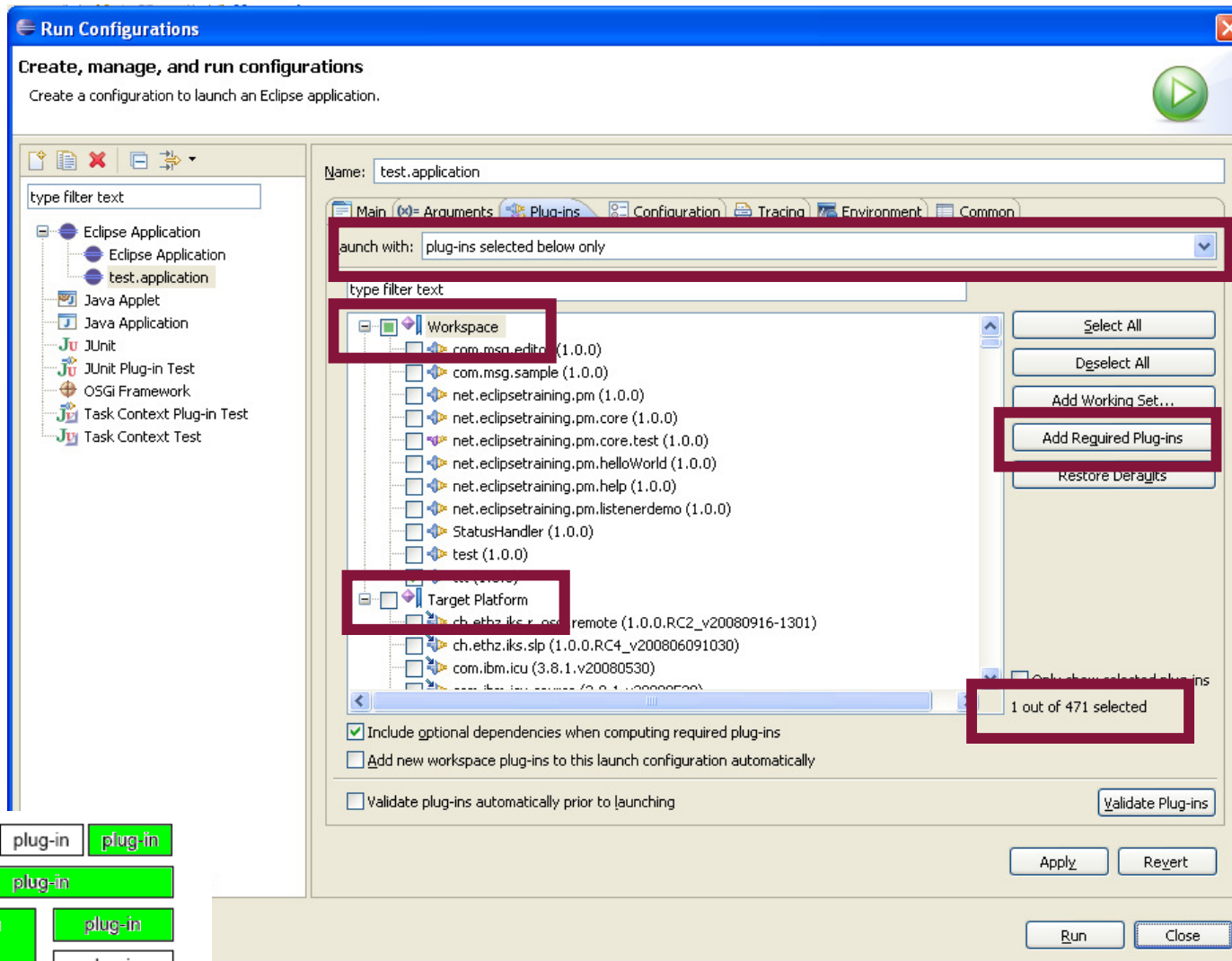
- All these dependencies are resolved at runtime
- you can configure which bundles are available in the run configuration of your application
- plug-ins are loaded on demand (lazy loading)
  - Plug-ins contribute to a menu or toolbar are loaded?

## Run configuration



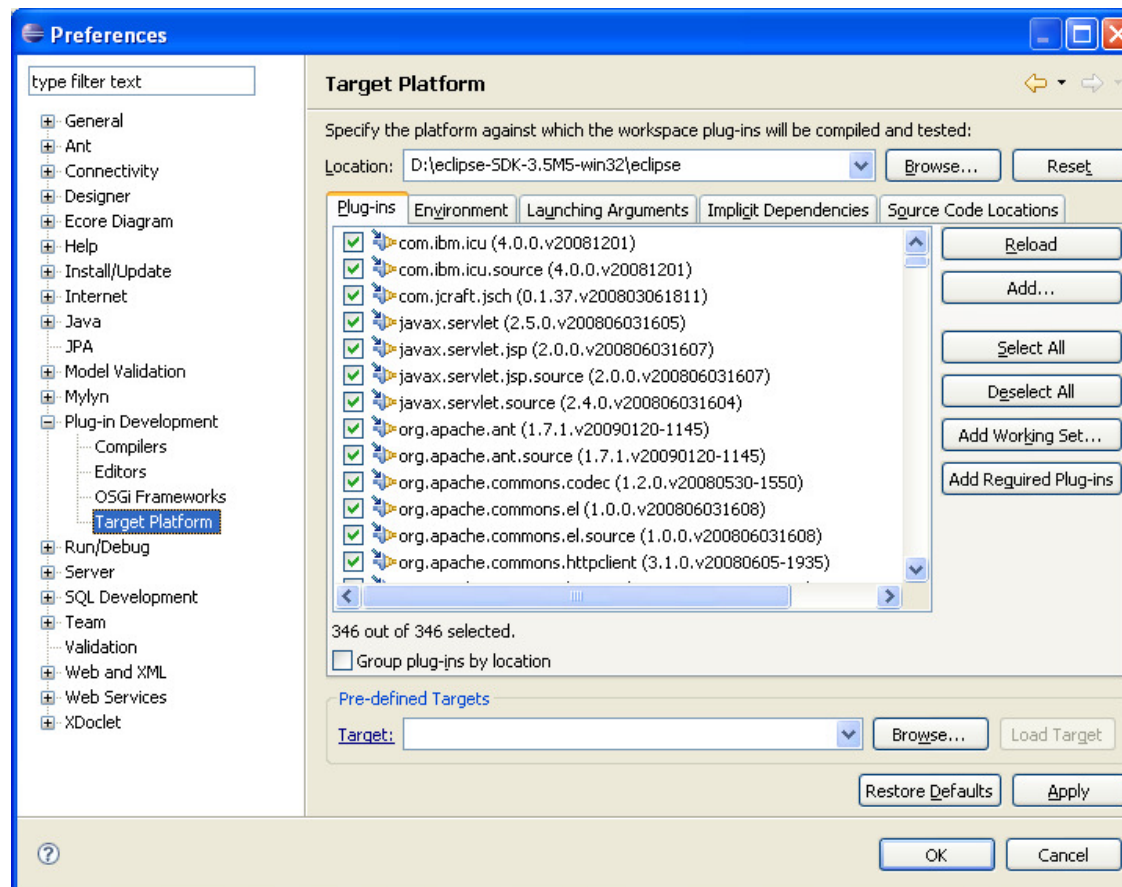


# Run Configuration



## Setup the target platform

- see language delta pack on <http://download.eclipse.org/eclipse/downloads/>



```
public void start(BundleContext context) throws Exception {  
    super.start(context);  
    plugin = this;  
}  
  
/*  
 * (non-Javadoc)  
 * @see  
 *   org.eclipse.ui.plugin.AbstractUIPlugin#stop(org.osgi.framework.BundleContext)  
 */  
public void stop(BundleContext context) throws Exception {  
    plugin = null;  
    super.stop(context);  
}
```

- Build two plug-ins **com.msg.client** and **com.msg.server** with a simple view (use the wizard)
- Define a package **com.msg.server.test** and implement a simple test class with a method  
**public String testme()** ... returning a test string
- Call this Method from the **com.msg.client** plug-in when it is loaded and print the string to **System.out**
- Don't forget to export your package and to set the dependencies correctly
- Run your application and make some tests at your own

**Vielen Dank für Ihre Aufmerksamkeit**



.consulting .solutions .partnership

