SHOW112

Domino® OSGi Development

David Taieb | Senior Software Engineer | IBM
Paul Fiore | Advisory Software Engineer | IBM
Elizabeth Sawyer | Development Manager | IBM

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Opening comments

A Social Business is...

- Engaged
- Transparent
- Nimble

This dynamic, open framework enables you to be nimble and develop applications faster and easier.

What is Domino OSGi?

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Introduction to OSGi™

- OSGi used to stand for Open Service Gateway Initiative but is now just a trademark used to define the specifications for a module system and service platform
- Component model: OSGi bundle
  - Private classpath: each bundle has its own classloader
  - Static resources
  - Fine grained dependencies control via package import/export
- Many other capabilities available
  - Easy deployment
  - Hot installation and dynamic configuration
  - Provisioning
  - Lifecycle via Bundle activator
- Multiple open source implementations
  - Eclipse equinox
  - Apache felix, Knopflerfish, Karaf, Concierge…

Where is it used in IBM Lotus® Notes®/Domino®?

Notes Client UI rewrite
using Expeditor/RCP platform
Tight Mail integration
Extension points
Preferences, ISA

Notes Client 8.0

Notes Designer UI rewrite
as plugins running within
same process as Notes Client
New Eclipse editor for Xpages
Java, CSS, LotusScript

Domino Designer 8.5

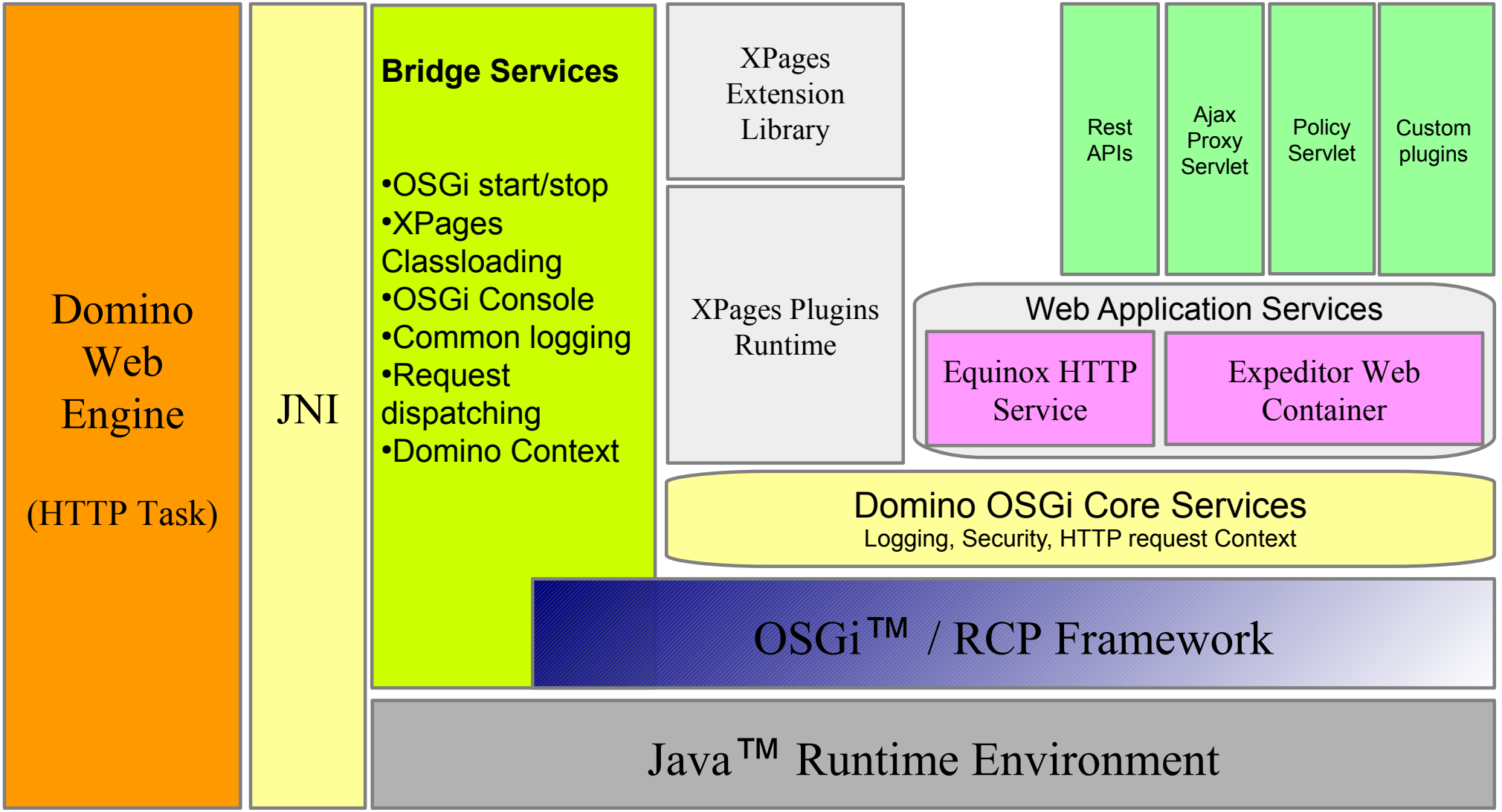
OSGi integration within the
HTTP task
Xpages runtime enhanced to
run as OSGi plugins
Security, OSGi console, …
Extension points

Domino Server 8.5.2

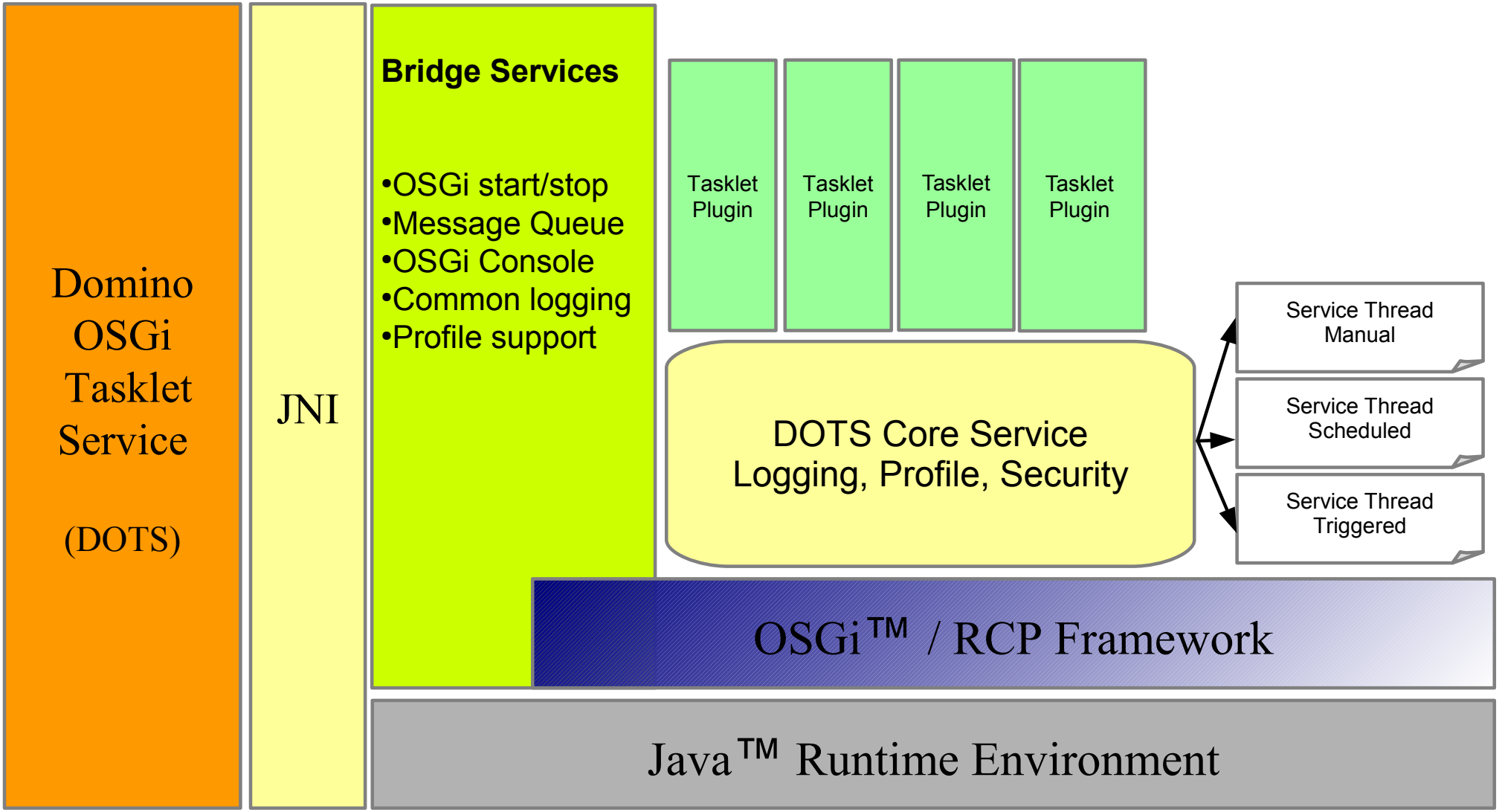
Domino Server 8.5.3+

Expand OSGi support to other
Domino services
•Rest APIs
•Open Social Container
•XPages Extension Library
•Domino OSGi Tasklet Service
•Domino Servlet Container

IBM Lotus Domino OSGi in HTTP architecture



IBM Lotus Domino OSGi Tasklet Service architecture



Multiple deployments

- File based
  - Bundles are packaged as jars or directories and dropped into the server file system
  - Recommended location is {DominoData}/domino/workspace/applications
  - Bundles execute with maximum security privileges
- NSF Based
  - Bundles are imported into a database using the UpdateSite.ntf template
  - Server is configured to use the updateSite database (notes.ini variable for HTTP, Profile for DOTS)
  - Documents containing the bundles must be signed with privileged user id to load, after which bundles execute with maximum security privileges
- PDE tool: Developer configuration
  - Server is instrumented to run bundles directly from eclipse workspace: deployment-less
  - No need to export bundles every time a change is made
  - Bundle are not physically installed making it easy to switch between configurations

Why does it matter to you?

- OSGi is now a very important part of the Notes/Domino programming model
  - Notes Client
  - Domino Designer
  - Domino Server: XPages, DOTS, Domino Web Container, Open Social Container, etc...
- OSGi skills are required for writing any next generation Notes/Domino application.
- In this session, we'll show you how:
  - To set up your development environment for Domino OSGi programming
  - Create a plugin for the HTTP Equinox Service, the XPD Web Container, XPages Extension Library and DOTS
  - How to run and debug these plugins using all 3 deployment methods: File Based, NSF Based and PDE Tool

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Prerequisites

- IBM Lotus Domino Server, 8.5.2+
  - http://www.ibm.com/developerworks/downloads/ls/lsds
- Eclipse, 3.6.2+
  - http://www.eclipse.org/downloads
- IBM Lotus Domino Debug Plugin (PDE tool)
  - http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=IBM%20Lotus%20Domino%20Debug%20Plugin
- OSGi Tasklet Service for IBM Lotus Domino, 2.0.3+
  - http://www.openntf.org/internal/home.nsf/release.xsp?databaseName=CN=NotesOSS2/O=NotesOSS!!Projects/pmt.nsf&documentId=84018B5D35E44E2F8625795E00140D3A&action=openDocument
- XPages Extension Library
  - Domino 8.5.3 Upgrade Pack 1
  - or http://www.openntf.org/internal/home.nsf/release.xsp?databaseName=CN=NotesOSS2/O=NotesOSS!!Projects/pmt.nsf&documentId=2E25B5DA1BDFB9B48625794D005B0821&action=openDocument

Agenda

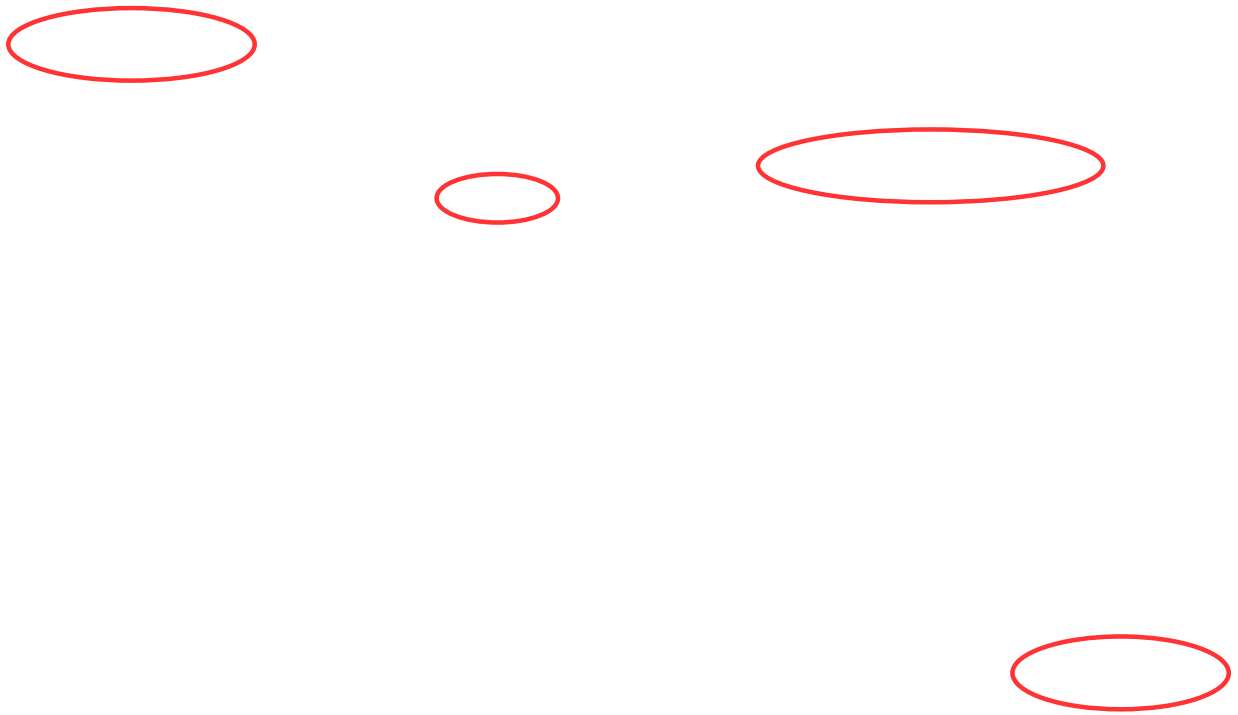
- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Domino OSGi for HTTP

- In the next few sections, we'll demonstrate capabilities related to Domino OSGi for the HTTP task
  - Create, deploy & debug an OSGi plugin that contributes a simple Hello World J2EE servlet using the Equinox HTTP Service
  - Create & deploy an OSGi plugin that contributes a complete J2EE web app using to the IBM Lotus eXpeditor™ Web Container.
  - Create & deploy an OSGi plugin that contributes an XPage custom control to the XPages Extension Library
- As a prerequisite, the next few slides will show you how to set up your Eclipse development environment for creating OSGi plugins for the Domino OSGi framework
- As part of this preparation, we'll also show you how to install the IBM Lotus Domino Debug Plugin (a.k.a. PDE tool). This tool helps you deploy and run your plugins directly from your eclipse workspace, thus tremendously improving your productivity.

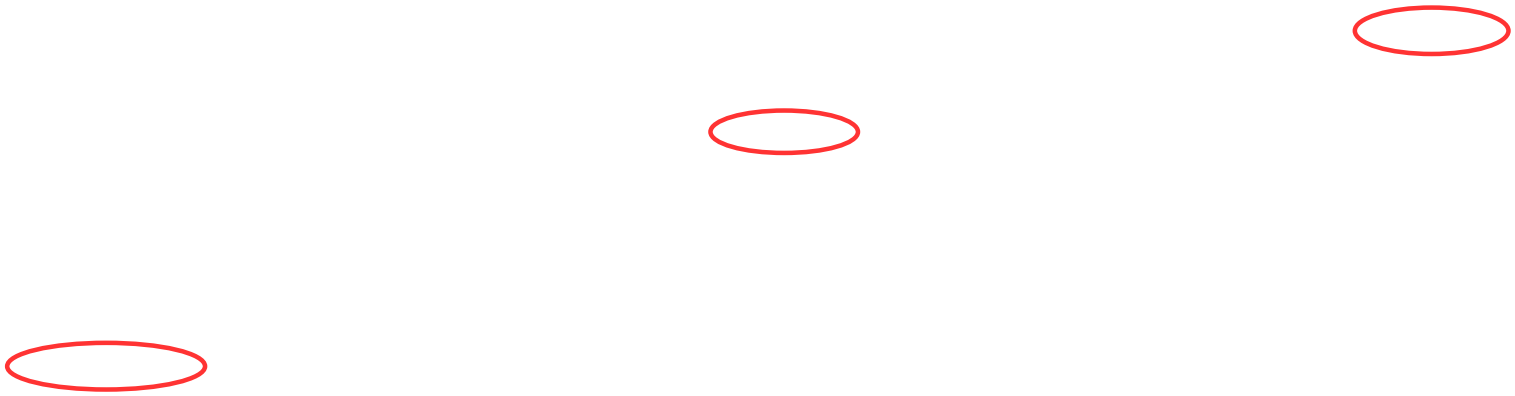
Environment setup for HTTP (Perspective)

- Switch into the Plug-in Perspective
  - Window->Open Perspective->Other...
  - Select Plug-In Development and Click OK



Environment setup for HTTP (Target Platform)

- Set up the Target Platform
  - Window->Preferences
  - Select Plug-in Development->Target Platform
  - Click on the Add... button

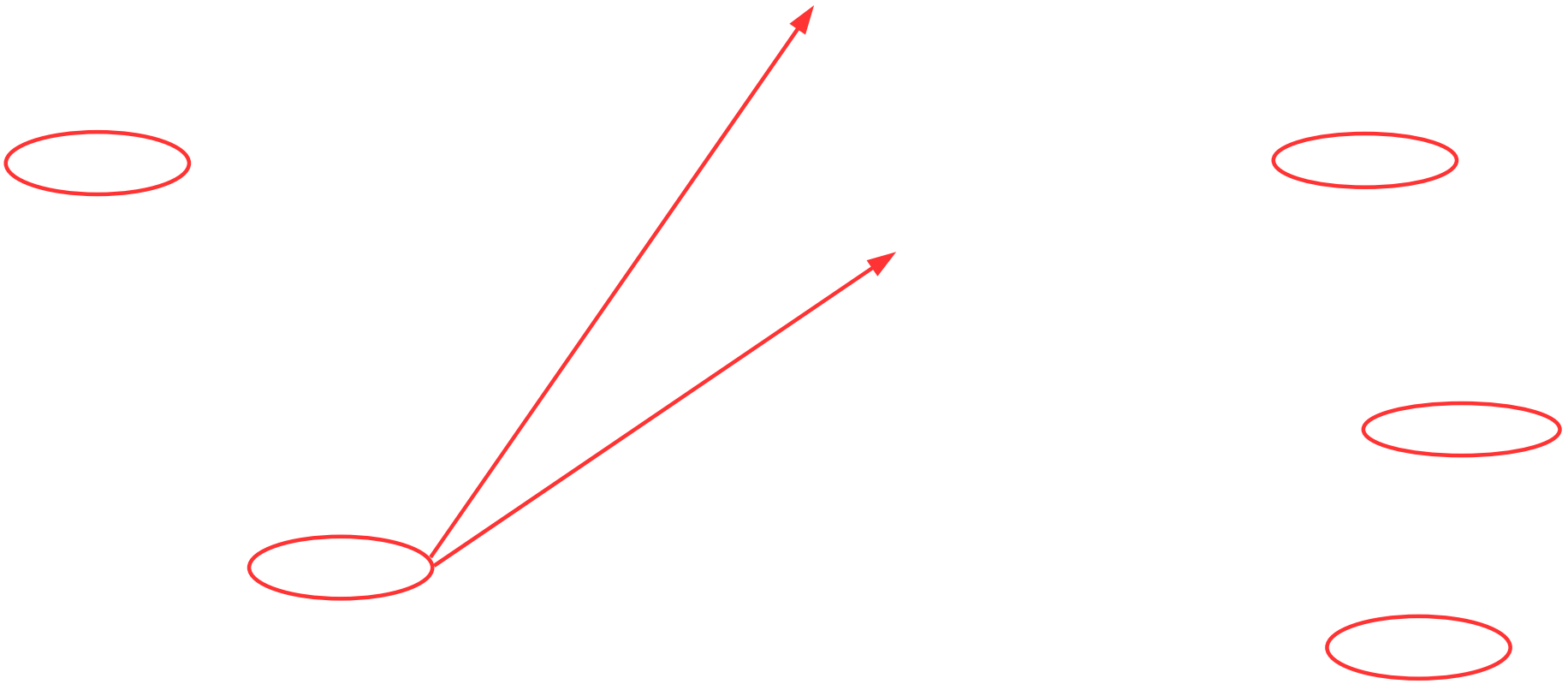


Environment setup for HTTP (Target Platform)

- On the Target Definition screen
  - Select: “Nothing: Start with an empty target definition”
  - Click on the Next button
- On the Target Content screen click the Add button

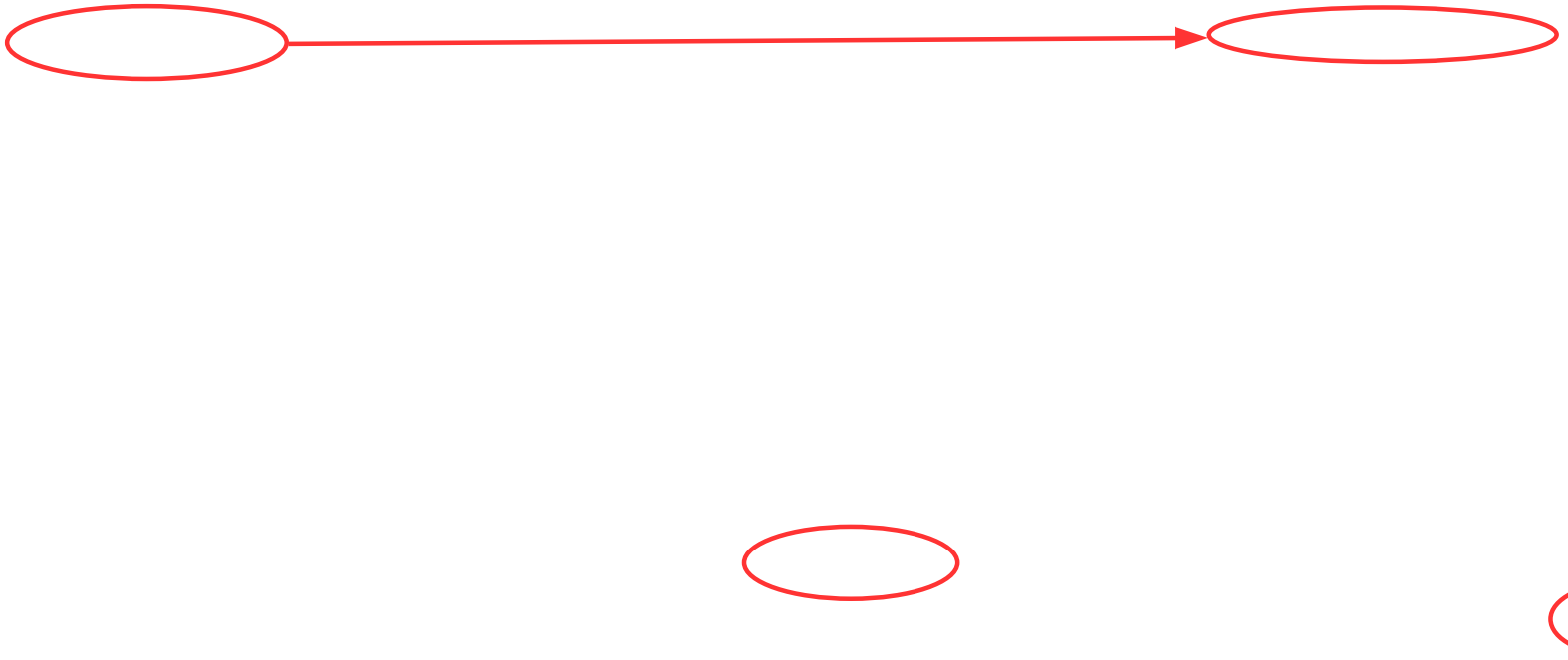
Environment setup for HTTP (Target Platform)

- On the Add Content screen, select Directory and click the Next button
- Add the {DominoBin}\osgi\rcp\eclipse directory and click the Finish button
- Repeat the process to add the {DominoBin}\osgi\shared\eclipse directory



Environment setup for HTTP (Target Platform)

- Give your target platform a Name, this can be whatever you want
  - Click on the Finish button
- Select the new Target definition you just created and click the OK button



Environment setup for HTTP (Notes.jar)

- Create a workspace plug-in to expose Notes.jar during development
  - File->New->Plug-in-Project

Environment setup for HTTP (Notes.jar)

- Project Name: com.ibm.notes.java.api
  - Select “an OSGi framework” radio button under the Target Platform section
  - Click on the Next button
- On the Content screen click on the Finish button

Environment setup for HTTP (Notes.jar)

- From the Package Explorer view, right click on the com.ibm.notes.java.api project and select Import... from the context menu

Environment setup for HTTP (Notes.jar)

- Select General->File System and click the Next button
- Browse to the {DominoBin}\jvm\lib\ext directory
  - Select Notes.jar from the file list
  - Click on the Finish button

Environment setup for HTTP (Notes.jar)

- Edit the META-INF\MANIFEST.MF file and go to the Runtime tab
- Click on the Add... button under the Classpath section

Environment setup for HTTP (Notes.jar)

- Select the Notes.jar file and click on the OK button
- Make sure the check box “Update the build path” is selected
- Click on the OK button

Environment setup for HTTP (Notes.jar)

- Back on the Runtime tab click on the Add... button under the Exported section
- Select all the packages from the Exported Packages screen and click on the OK button

Environment setup for HTTP (Notes.jar)

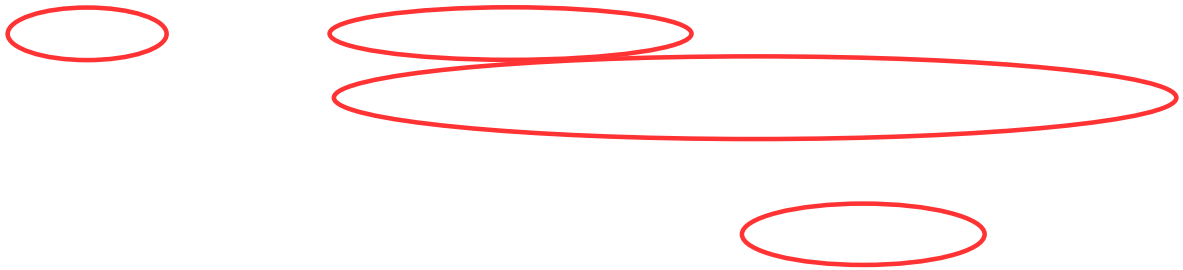
- Save the MANIFEST.MF file (File->Save or Ctrl-S) and close the editor window

Environment setup for HTTP (PDE tool)

- Install the IBM Lotus Domino Debug Plug-in
  - Help->Install New Software...

Environment setup for HTTP (PDE tool)

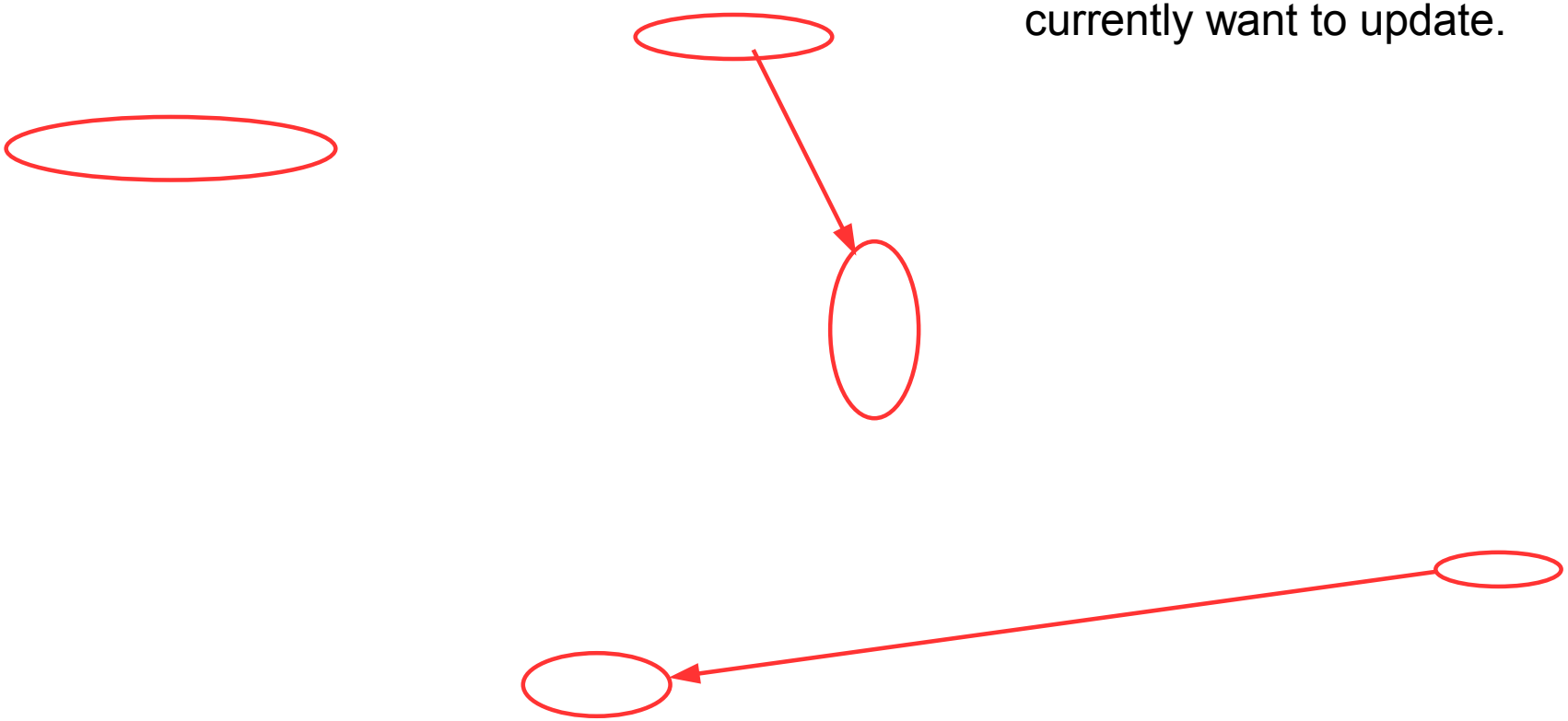
- On the Install screen click on the Add... button
- Give the Plugin any name you want
  - Click on the Archive... button and select the zip file you downloaded for the PDE Tool from OpenNTF
  - Click on the OK button



Environment setup for HTTP (PDE tool)

- Select the Domino Debug Plugin and the Domino Debug Feature
- Click on the Next button

Make sure the only software sites enabled are ones you currently want to update.



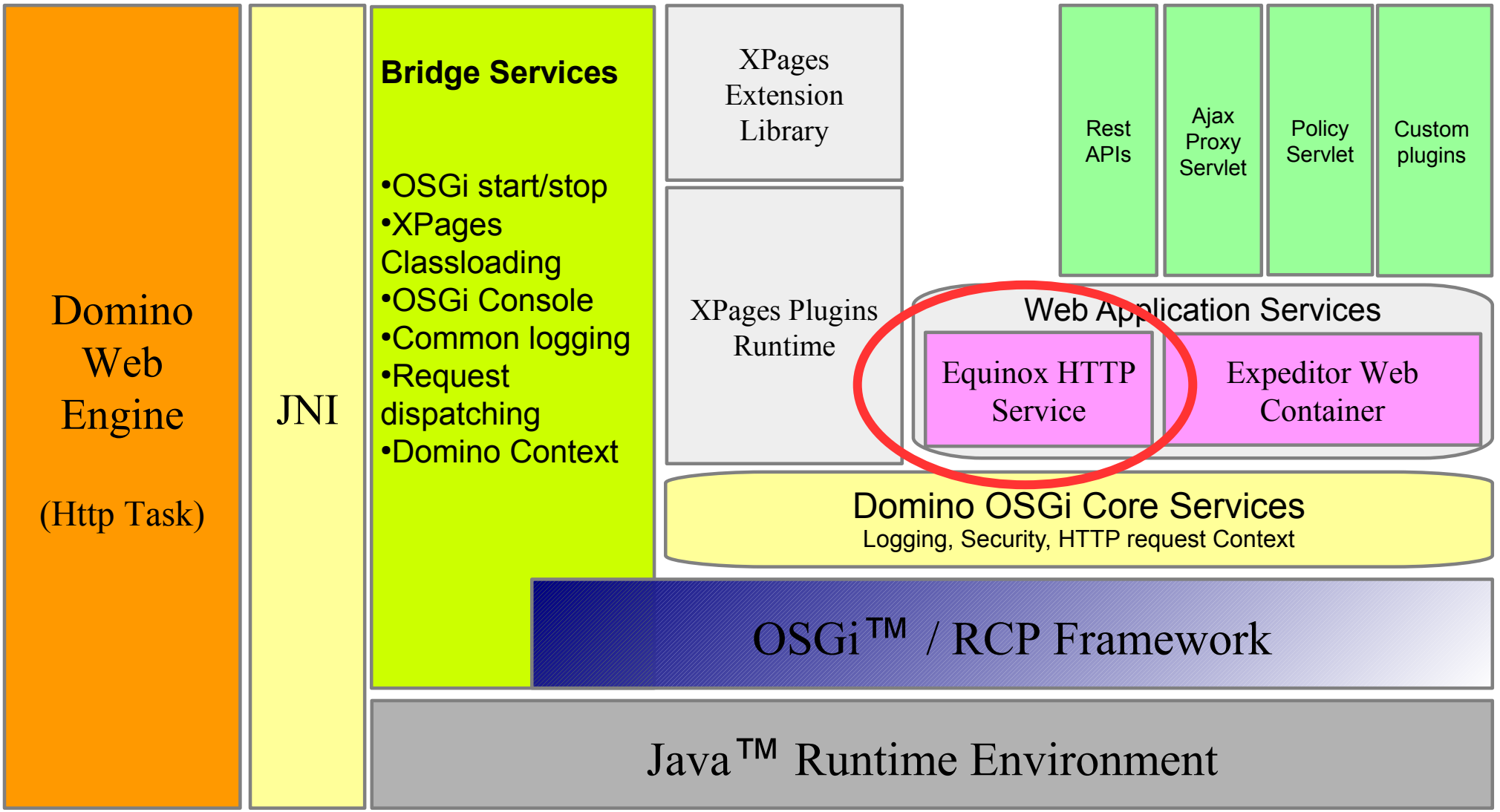
Environment setup for HTTP (PDE tool)

- On the Install screen click on the Next button
- On the License screen, select the “I accept the terms of the license agreement” and click on the Finish button
  - Click the OK button on any Security Warning dialogs
- When prompted click on the Restart Now button on the Software Updates dialog

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Creating a simple servlet running on Equinox



Section goals

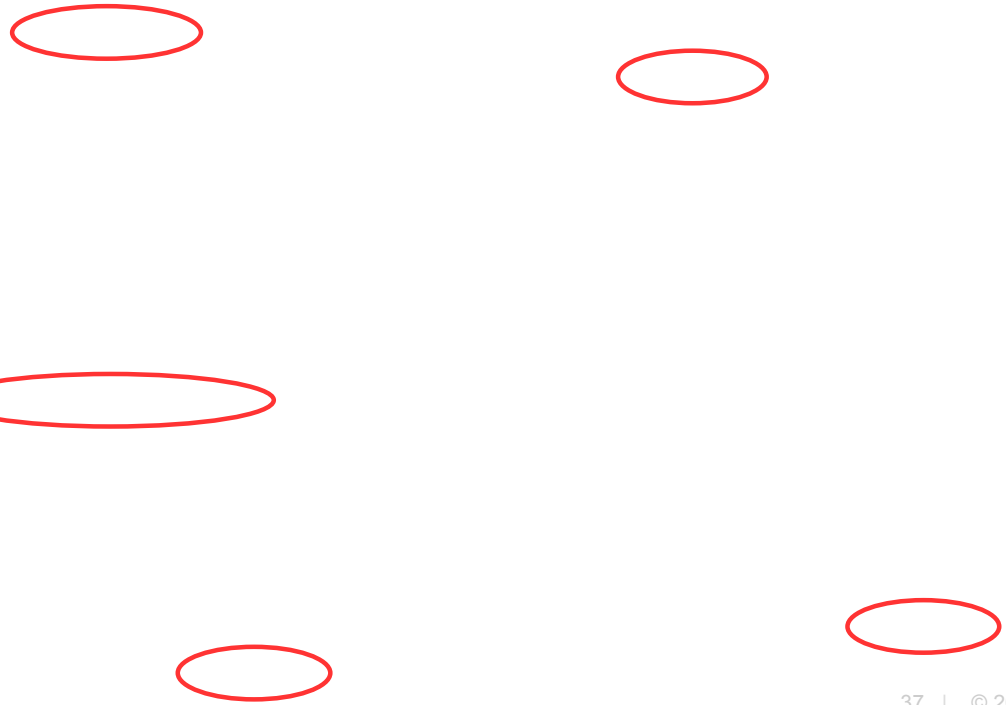
- Create an OSGi plugin that contributes a simple servlet using the Equinox HTTP Service
- The Equinox HTTP Service provides a way to create lightweight servlets via Eclipse Equinox extension points without the need to produce a full J2EE web app (as specified by the J2EE standard specification)
- It is recommended to use this method when you need to create stand-alone servlets that do not require the use of more sophisticated J2EE APIs like filters.
- As an example, the Domino Rest APIs are implemented using this framework.
- In this section, we'll also show you how to deploy and debug your plugin using the PDE tool

Creating a simple servlet running on Equinox

- Create a new workspace plug-in
  - File->New->Plug-in-Project

Creating a simple servlet running on Equinox

- Project Name: com.ibm.ls2012
  - Select “an OSGi framework” radio button under the Target Platform section
  - Click on the Next button
- On the Content screen change the name to SimpleServlet
  - Click on the Finish button

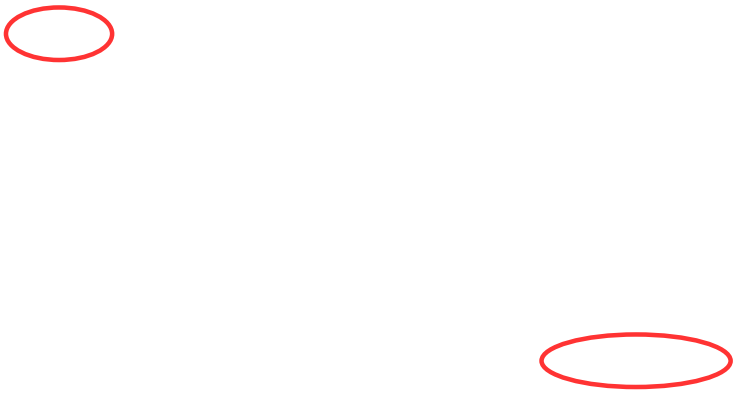


Creating a simple servlet running on Equinox

- Edit the META-INF\MANIFEST.MF file and go to the Dependencies tab
- Click on the Add... button under Imported Packages add the plug-ins listed below

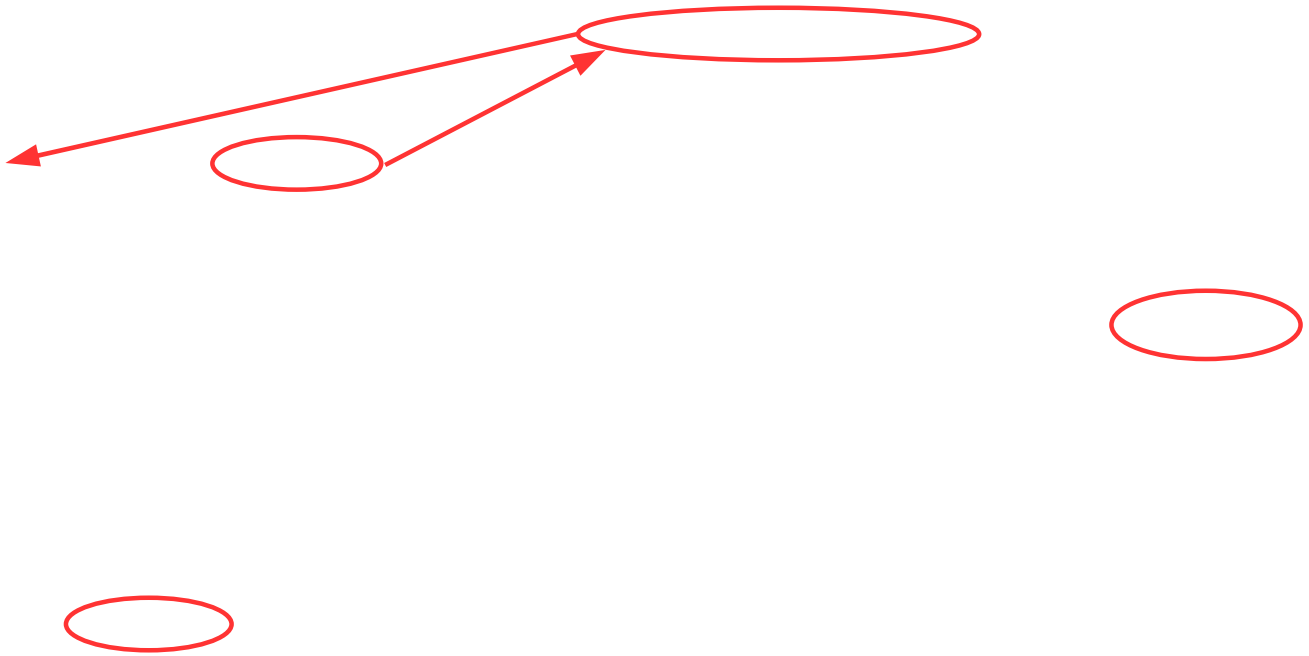
Creating a simple servlet running on Equinox

- Right click on the com.ibm.ls2012 package and click New->Class



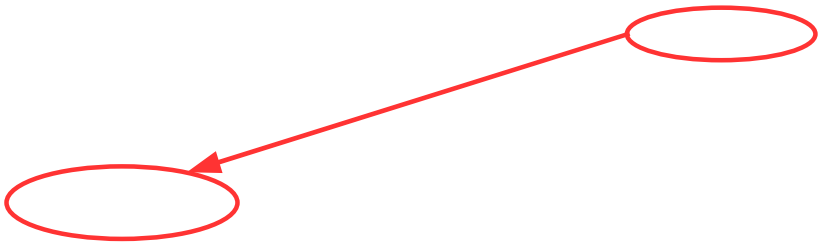
Creating a simple servlet running on Equinox

- Create a new Java class named SimpleServlet extending javax.servlet.http.HttpServlet



Creating a simple servlet running on Equinox

- Go to the Overview tab, under Extension / Extension Point Content and click on Extensions
- Click Yes on the Extensions pages hidden screen

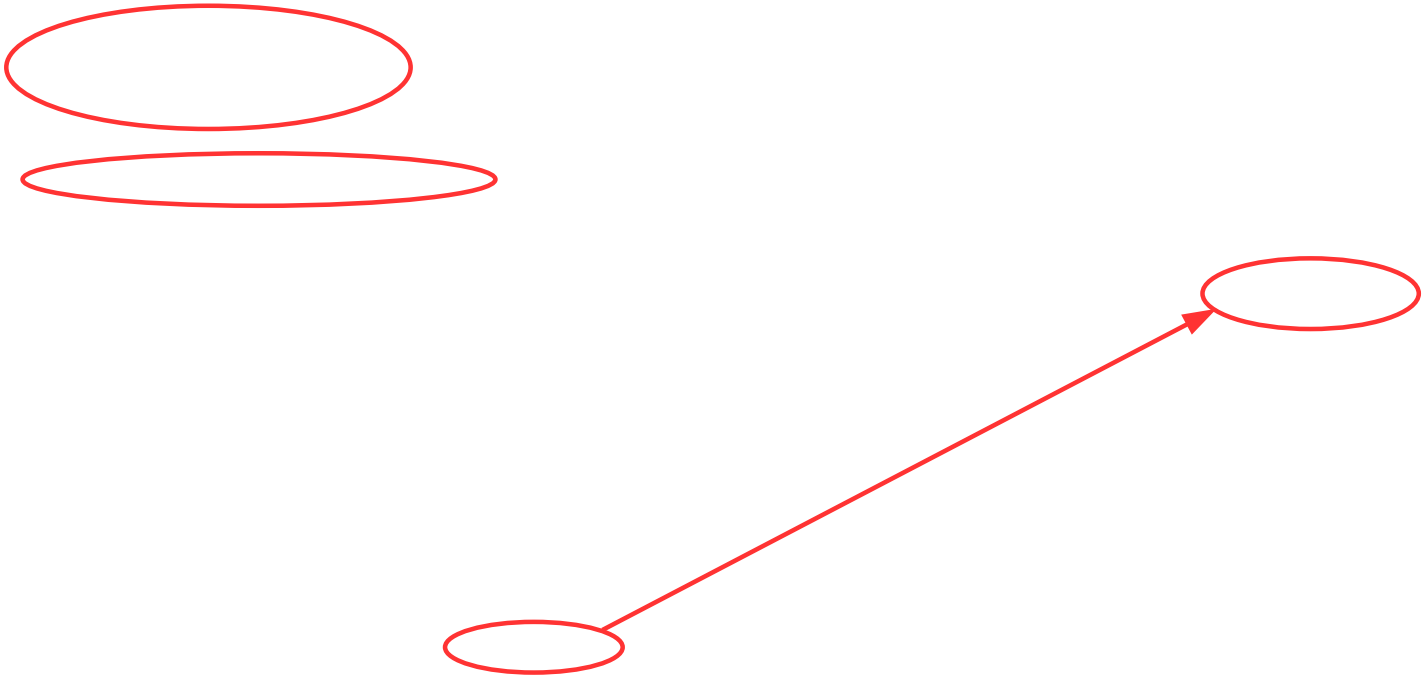


Creating a simple servlet running on Equinox

- Go to the Extensions tab and click the Add button...

Creating a simple servlet running on Equinox

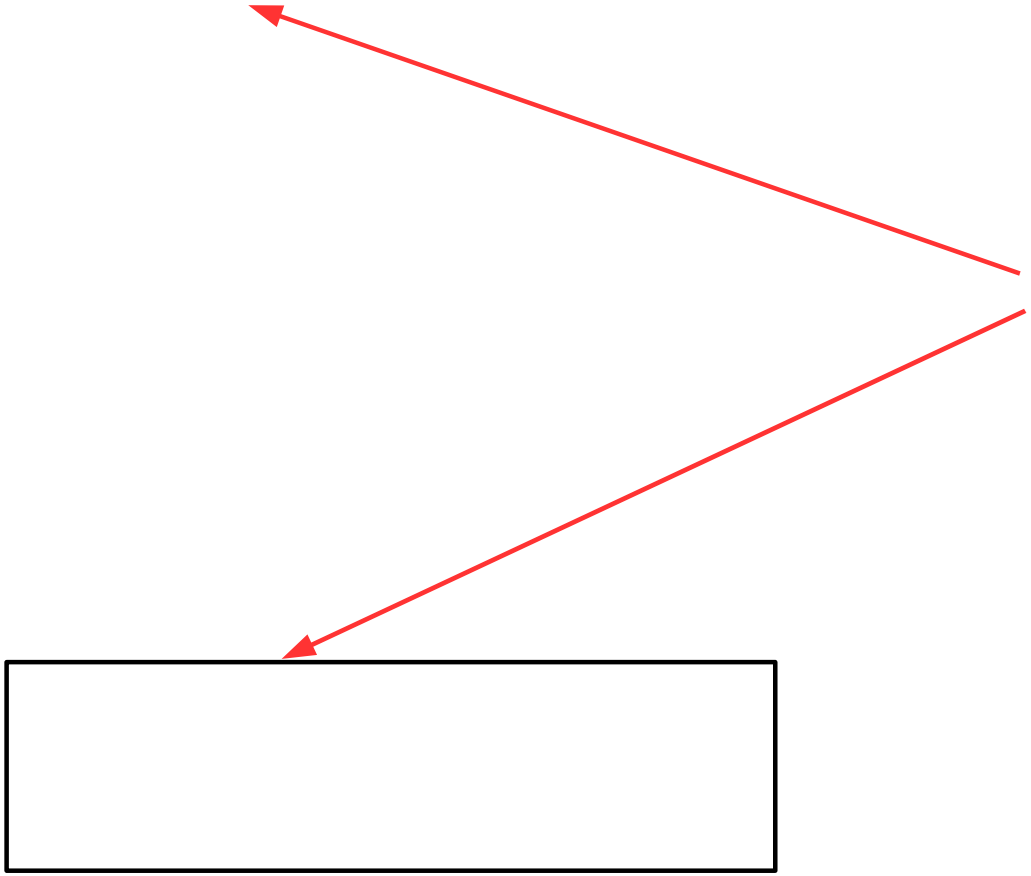
- Deselect the “Show only extension points from the required plug-ins”
  - Select the org.eclipse.equinox.http.registry.servlets Extension Point
  - Click on the Finish button
- On the New plug-in dependency screen, click the Yes button



Creating a simple servlet running on Equinox

- Go to the plugin.xml tab and add the following content

Matching urls:
•http://{server}/simpledemo
•http://{server}/{db.nsf}/simpledemo



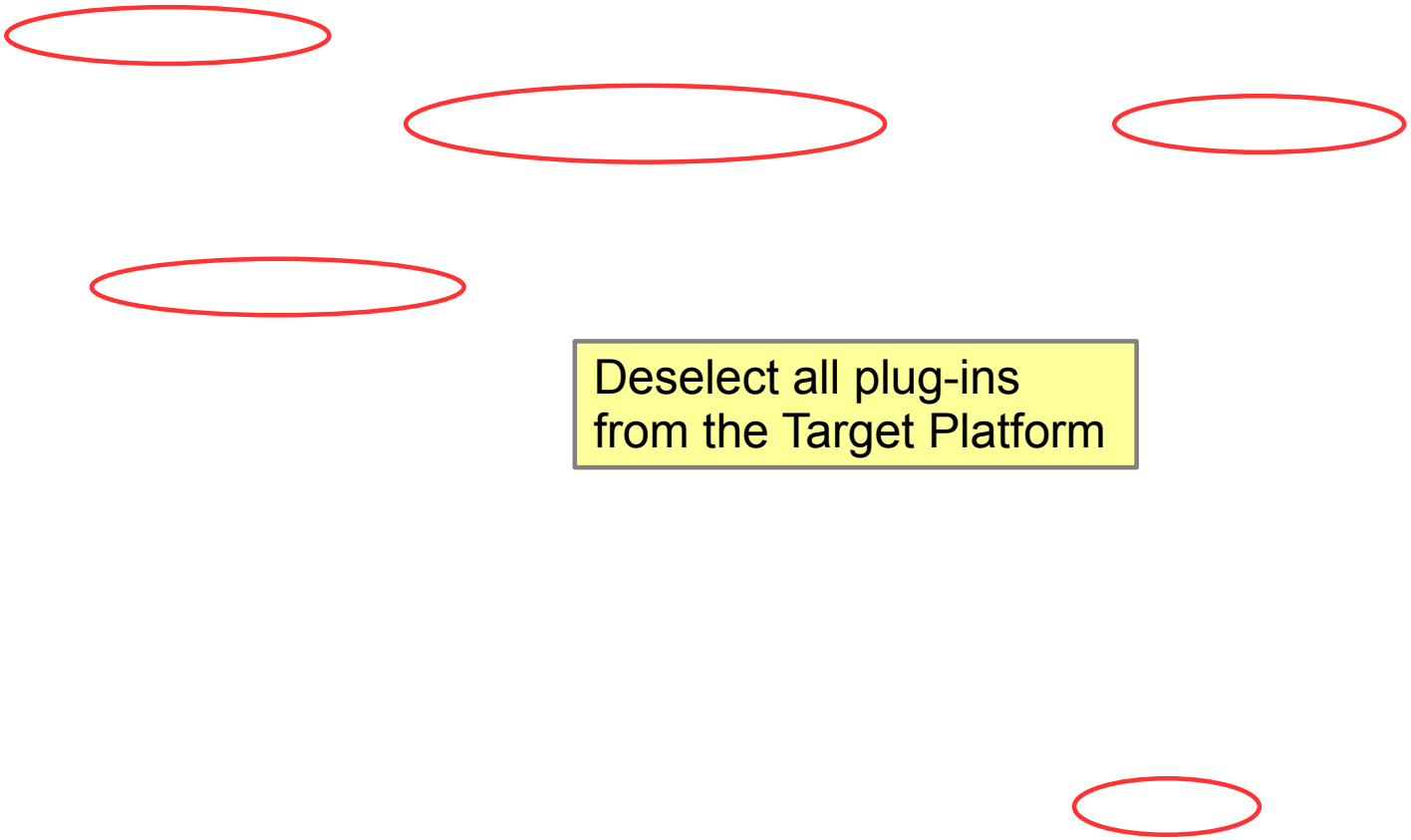
Deploying & debugging a simple servlet

- Click on Run->Debug Configurations...

Deploying & debugging a simple servlet

- Create a new OSGi Framework

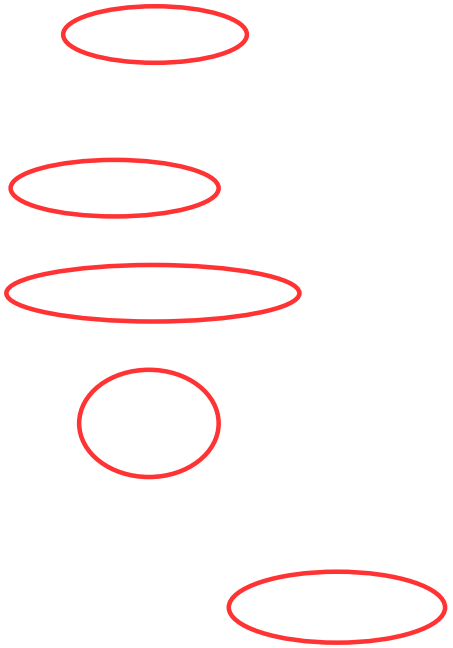
Deploying & debugging a simple servlet



Deploying & debugging a simple servlet

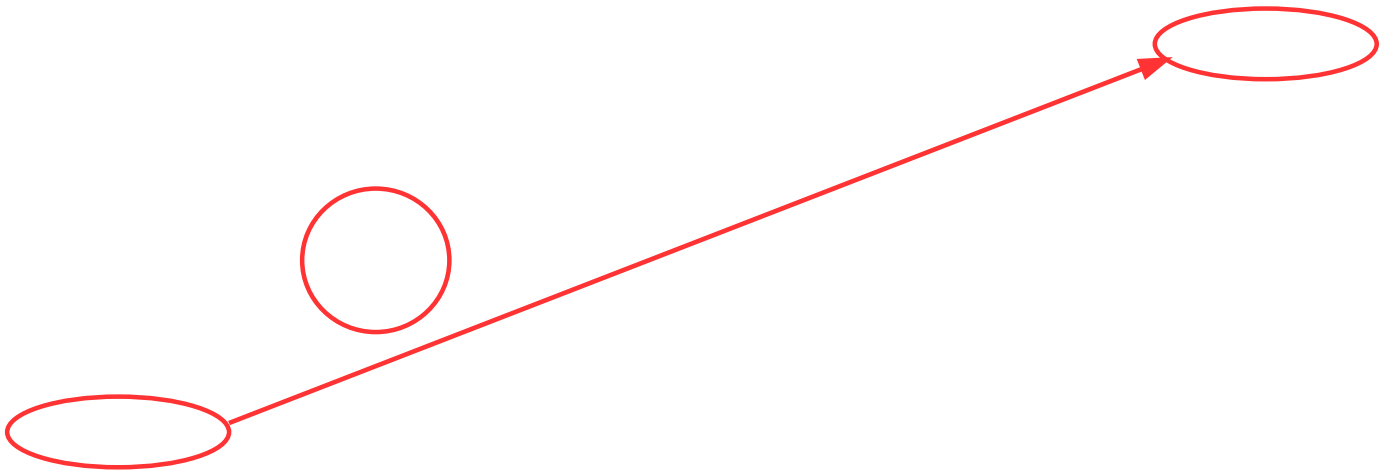
- Create a new Remote Java Application

Deploying & debugging a simple servlet



Deploying & debugging a simple servlet

- Click on the HTTP OSGi Framework and then click the Debug button
  - This launches the PDE Tool
  - Select the Domino Binary Directory, Data will pre-populate to default
  - Click on the OK Button
- If everything worked OK then you'll see a Success dialog
  - Click on the OK button



Deploying & debugging a simple servlet

- Set a break point in the code

- Enable debugging via notes.ini
  - JavaEnableDebug=1
  - JavaDebugOptions=transport=dt_socket,server=y,suspend=n,address=8000
- Restart HTTP on the server
- Go back to Run->Debug Configurations...
  - Select SimpleServlet under Remote Java Applications
  - Click on the Debug button

Deploying & debugging a simple servlet

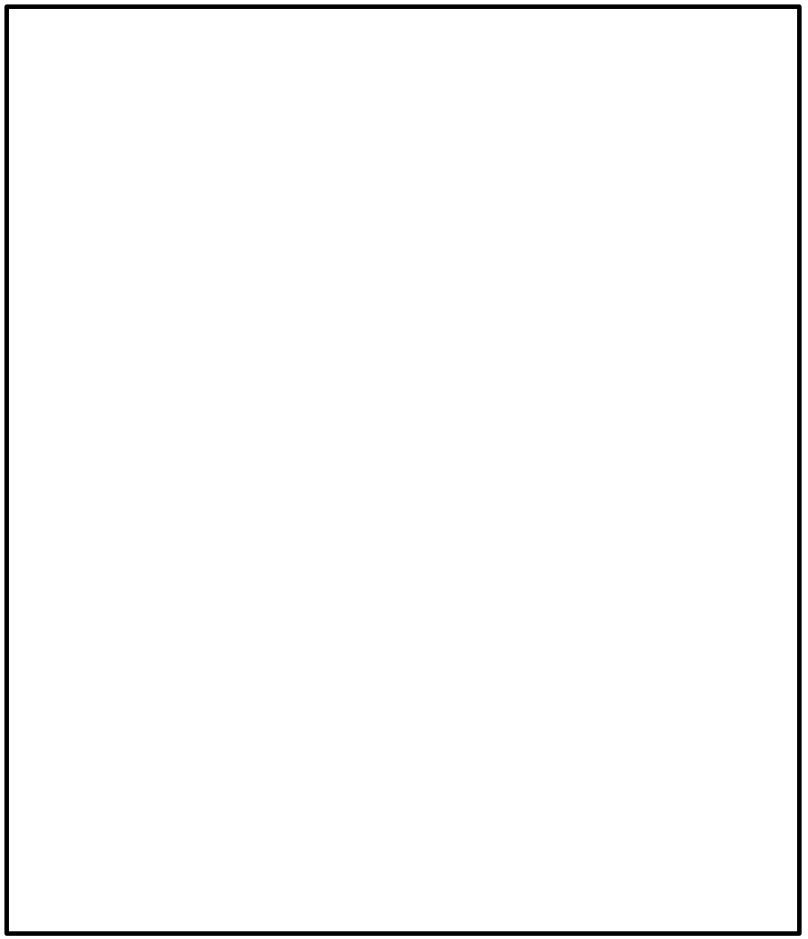
- Browse to http://{server}/simpledemo
- Confirm the switch to the Debug Perspective by clicking the Yes button

Deploying & debugging a simple servlet

- Debug as required

Deploying & debugging a simple servlet

- Output



Deploying & debugging a simple servlet

- Browse to http://{server}/{db.nsf}/simpledemo
- Authentication / Authorization can be done via HttpContext interface

Deploying & debugging a simple servlet

- On the server console you can manage this servlet like any other OSGi plug-in

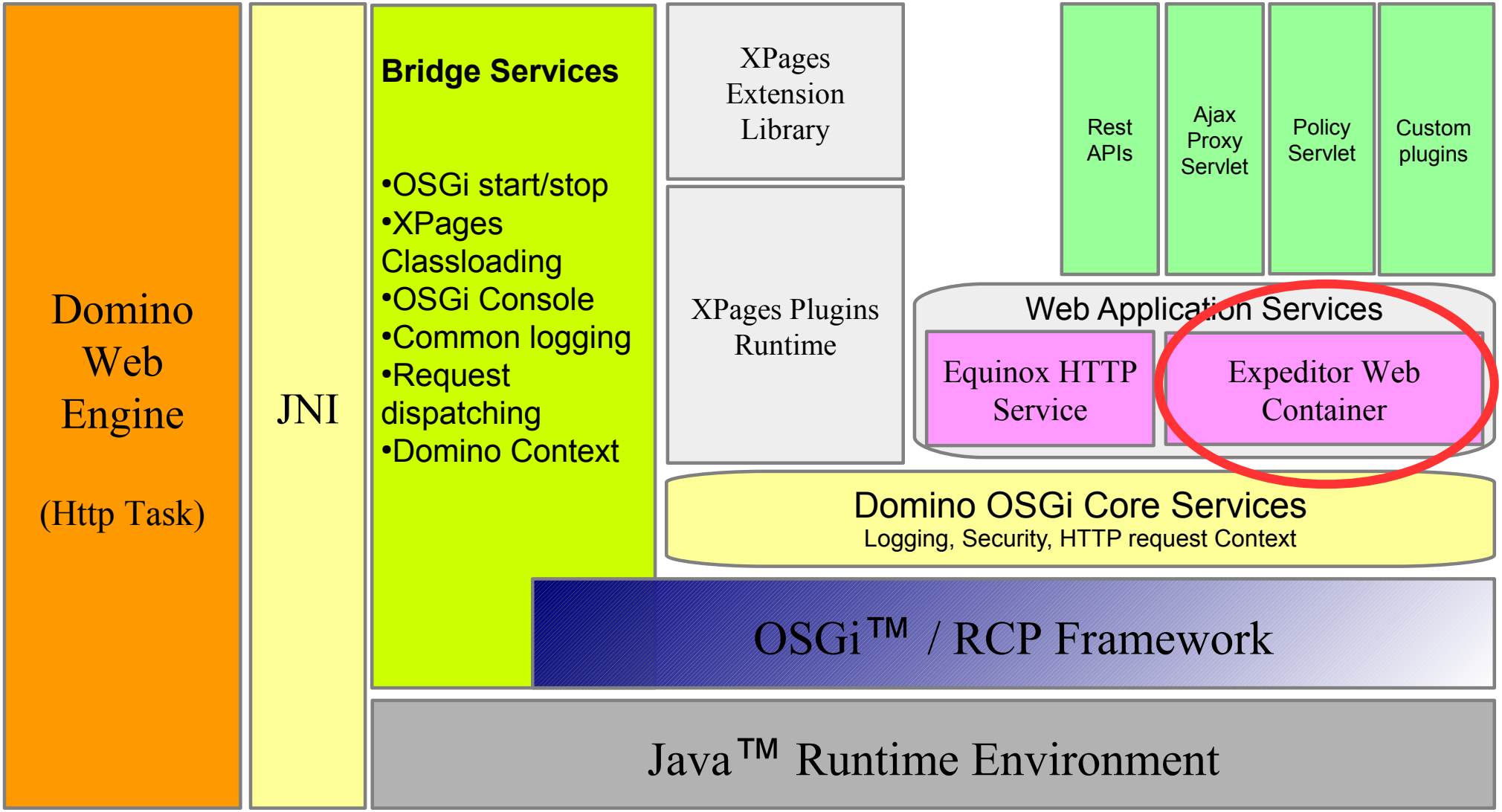
Summary of what we accomplished in this section

- We learned how to create an OSGi plugin that contribute a simple servlet using the Equinox HTTP Service
- We deployed the plugin using the PDE Tool
- We configured the HTTP Task to run in debug mode and attached a debugging session from Eclipse.
- We learned how to debug through the servlet code.
- We also showed a few OSGi console commands to verify the status of our plugins.

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Creating a plugin for the Domino XPD Web Container

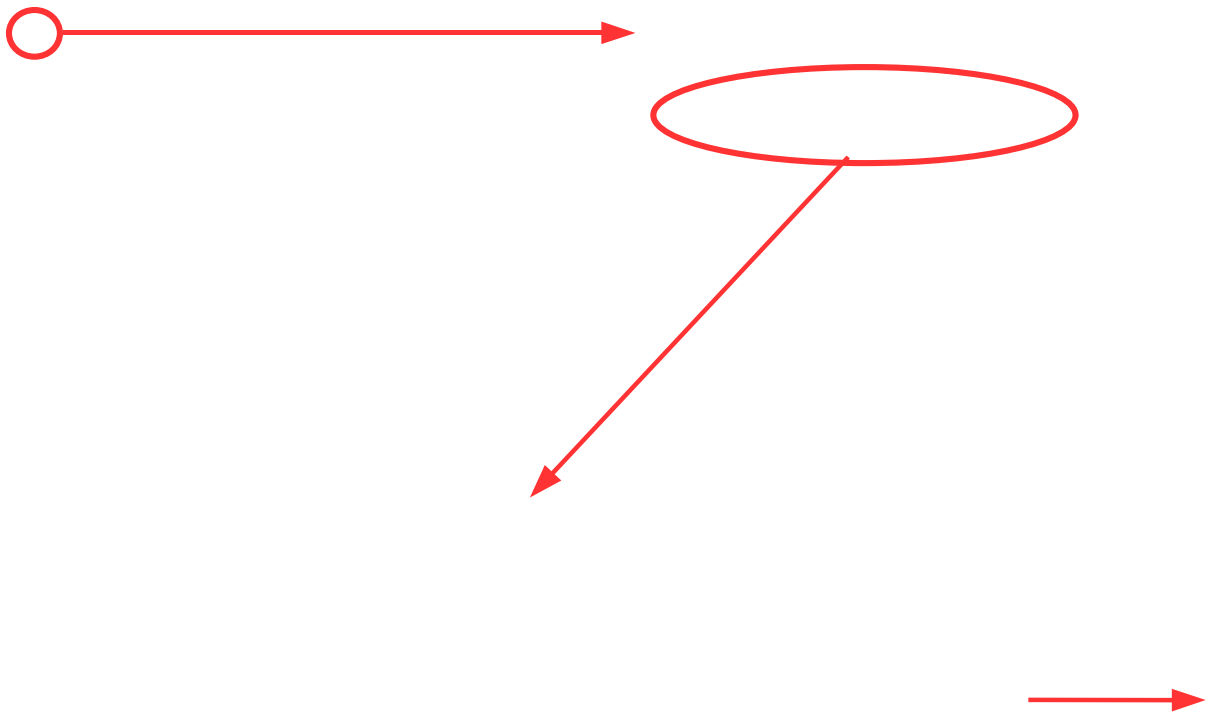


Section goals

- Create an OSGi plugin that contributes a J2EE web app to the eXpeditor™ Web Container
  - The J2EE web app contains a fully functional web.xml descriptor with simple Hello World Servlet that output a static message
  - Feel free to augment the sample web app by using other APIs of the J2EE specs like Filter, security-constraints, etc...
- Deploy the plugin to the Domino Server using an NSF-based repository
  - This method of deployment is interesting because it leverages the replication capabilities of NSF allowing allowing you to automatically deploy the plugin to all the Domino servers in your cluster.
- Verify that the plugin is correctly installed using OSGi console commands
- Invoke the web app from a browser

Reminder of what the target platform should be

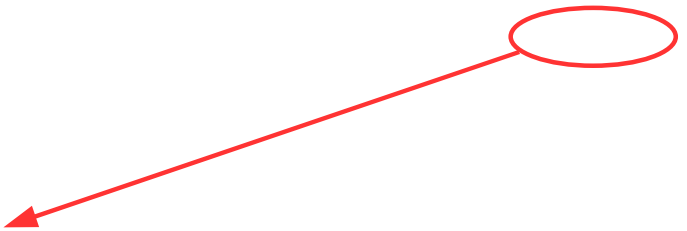
Plugin for the Domino XPD Web Container
Use PDE Tool to configure the com.ibm.notes.java.api plugin



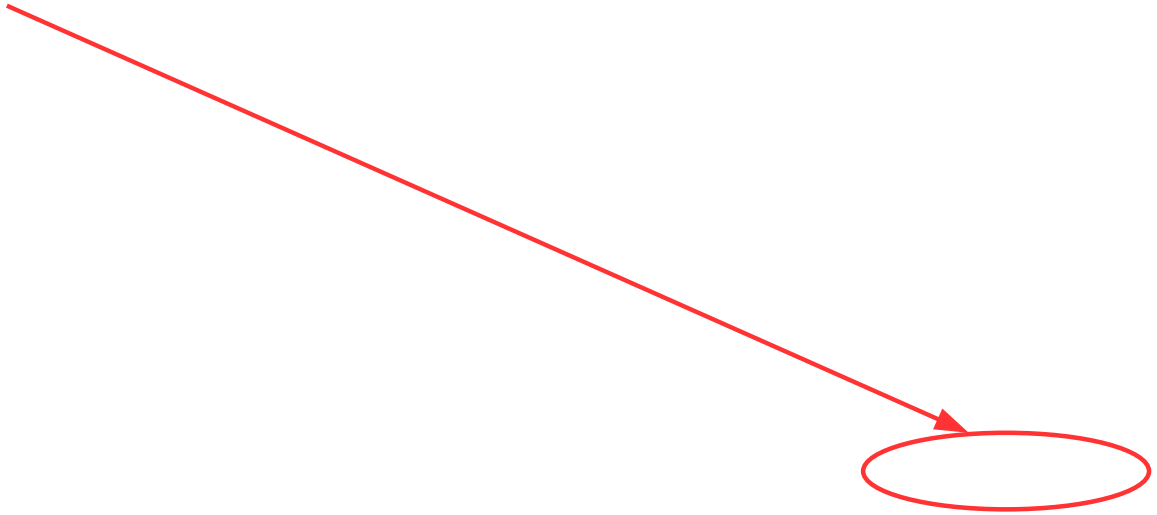
Standard New Plugin Project Wizard

Standard New Plugin Project Wizard (2)

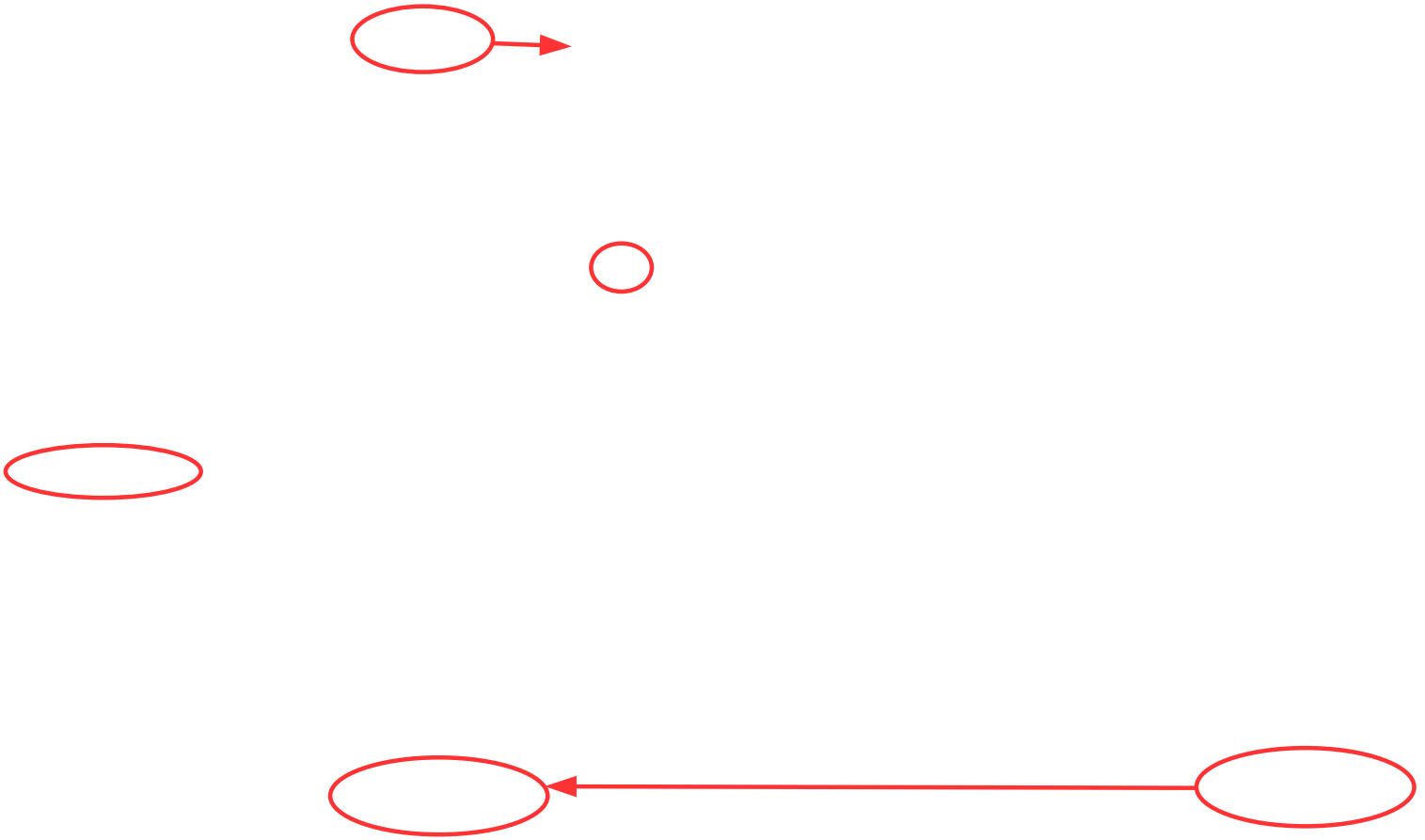
Plugin configuration



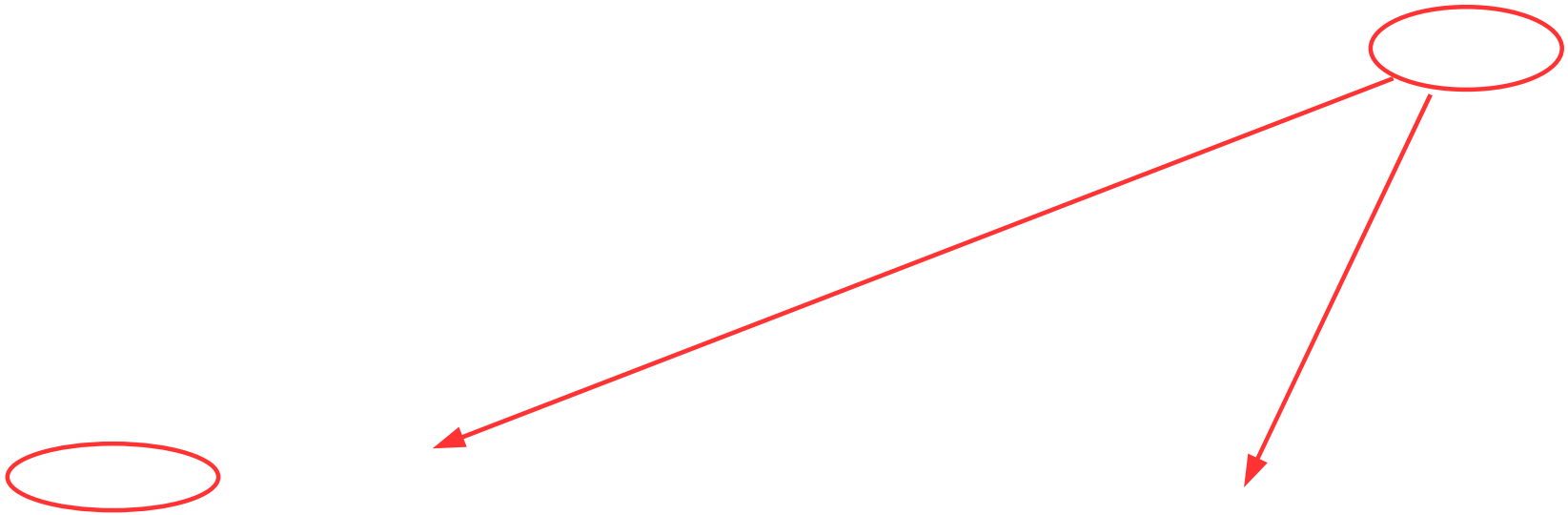
Fix the manifest



Create the extension point

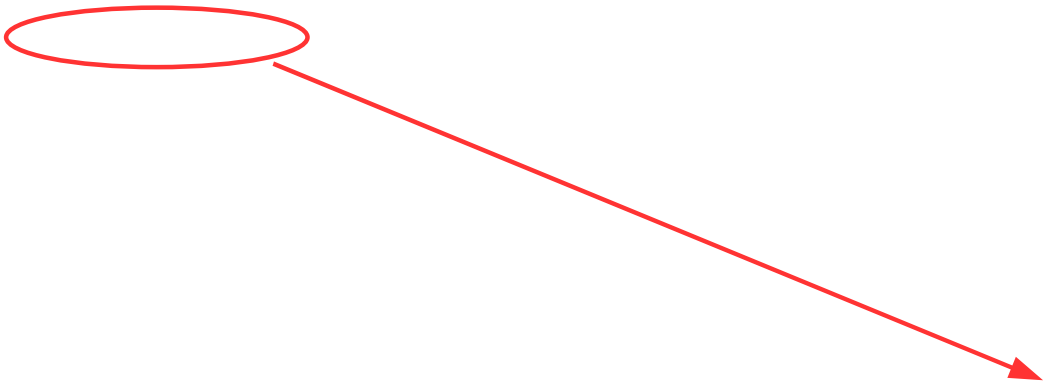


Update the dependencies

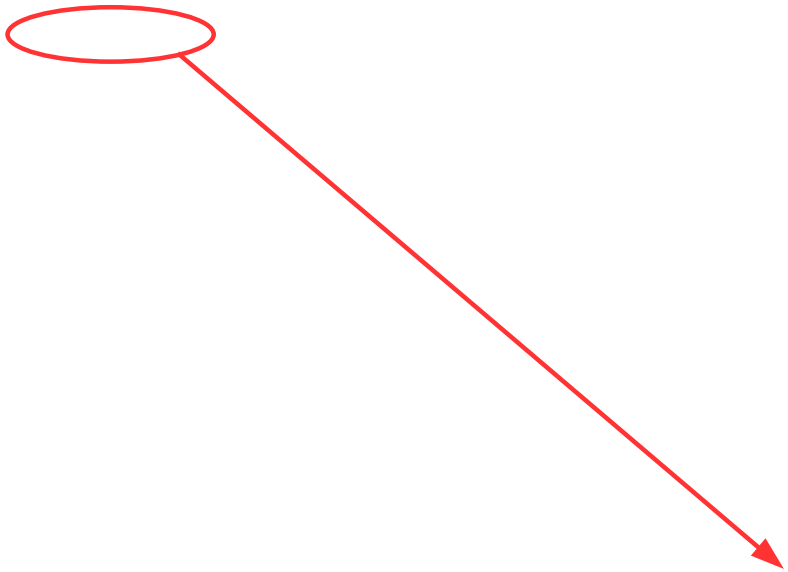


Configure the extension point contextRoot

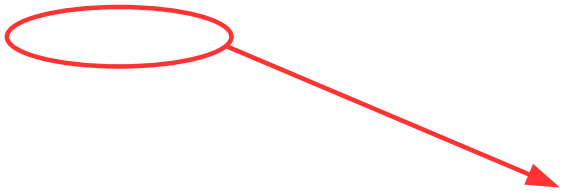
Configure the extension point contentLocation



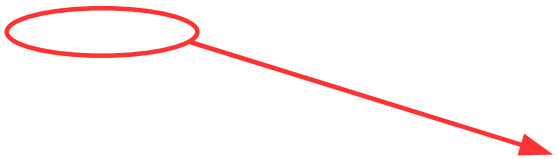
Create the “WebContent” folder



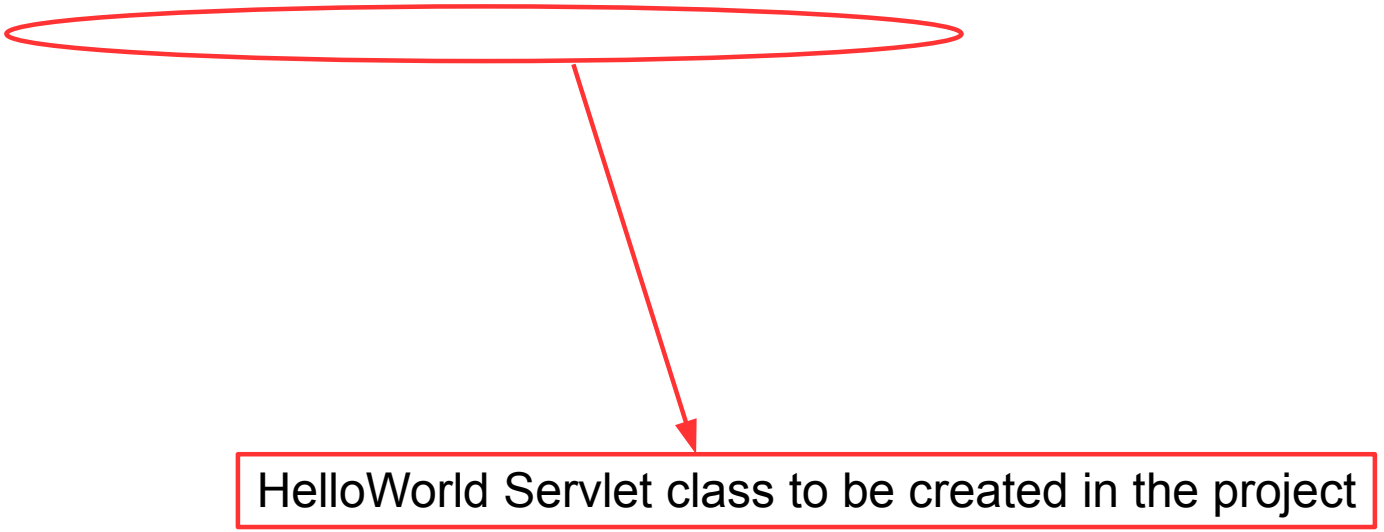
Create the “WEB-INF” directory under WebContent



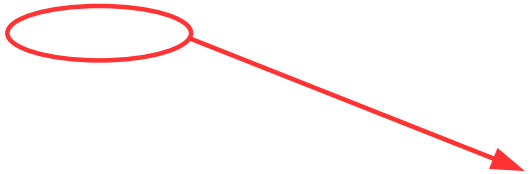
Create the “web.xml” file J2EE descriptor



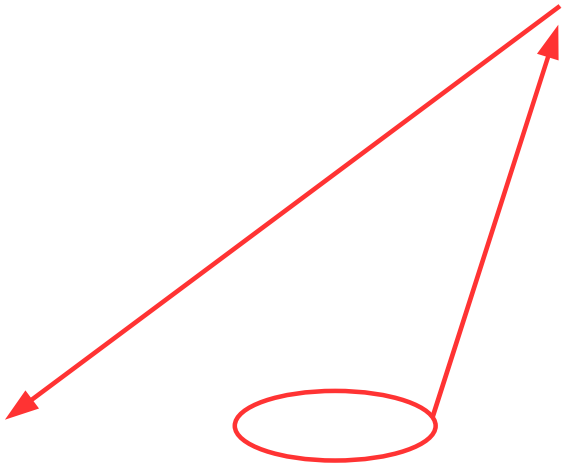
Add a reference to the HelloWorld Servlet in web.xml



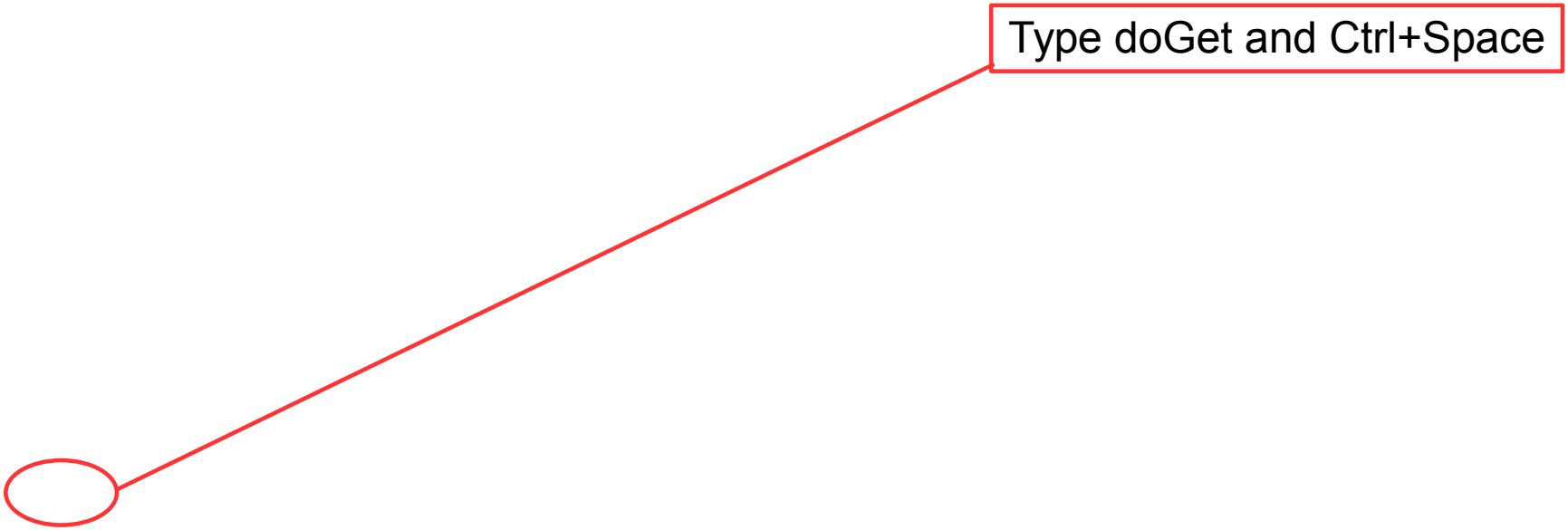
Create the HelloWorld Servlet class



Create the HelloWorldServlet class (2)

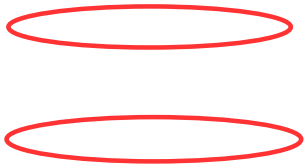


Edit HelloWorldServlet.java and override doGet method



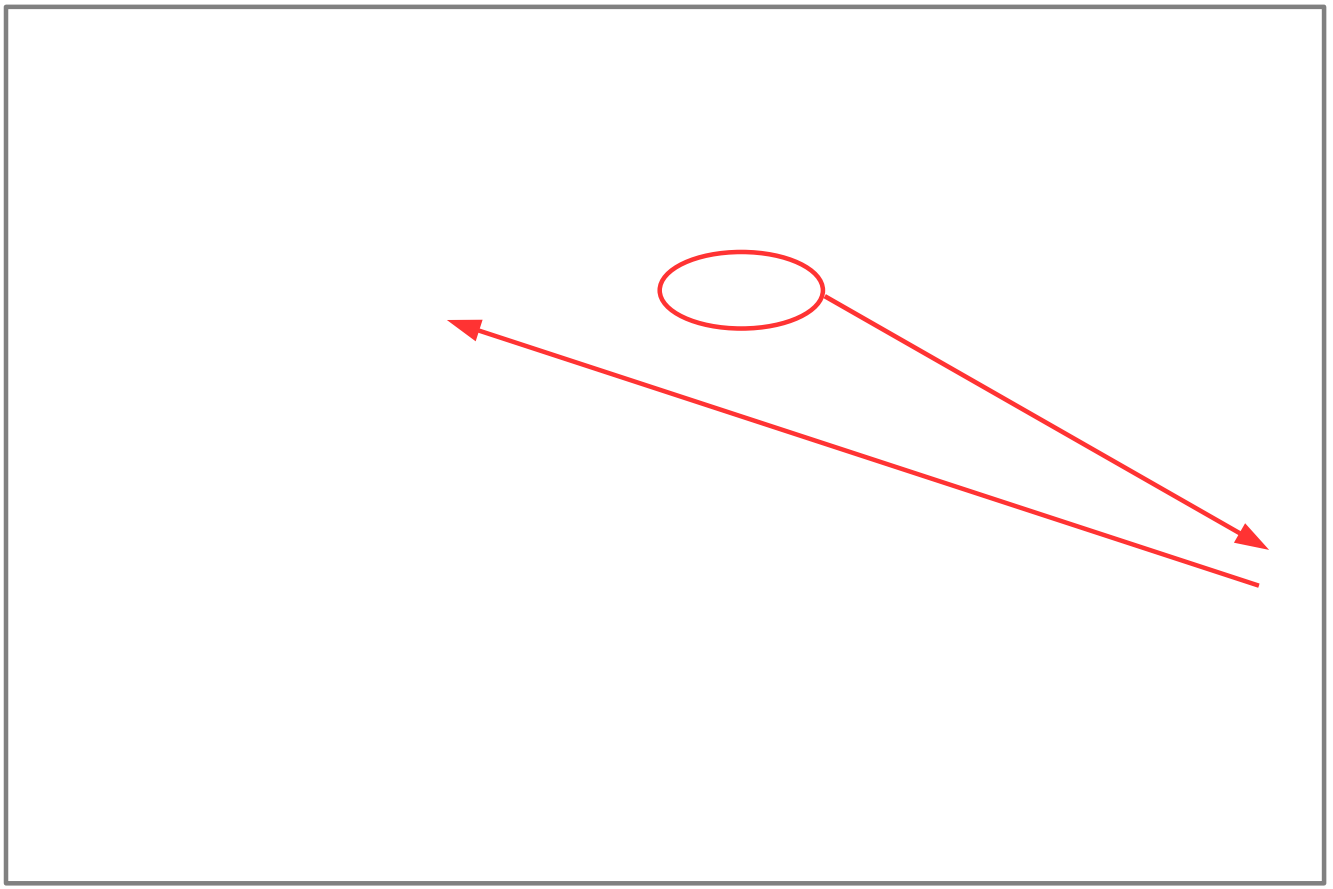
Add logic to doGet method

Update the build.properties file to add plugin.xml and WebContent entries

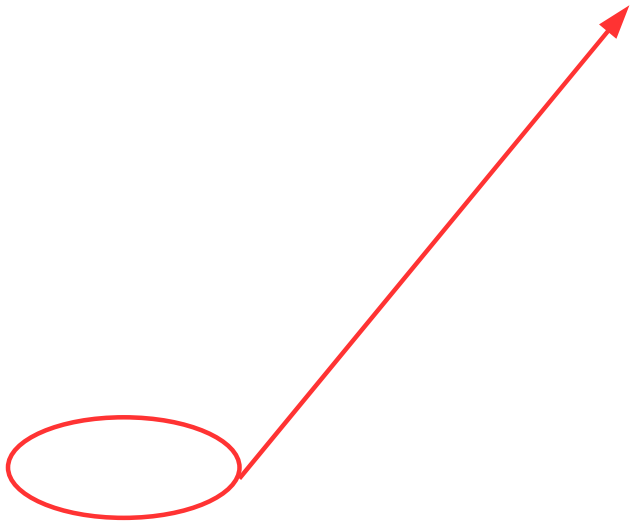


Create eclipse feature project

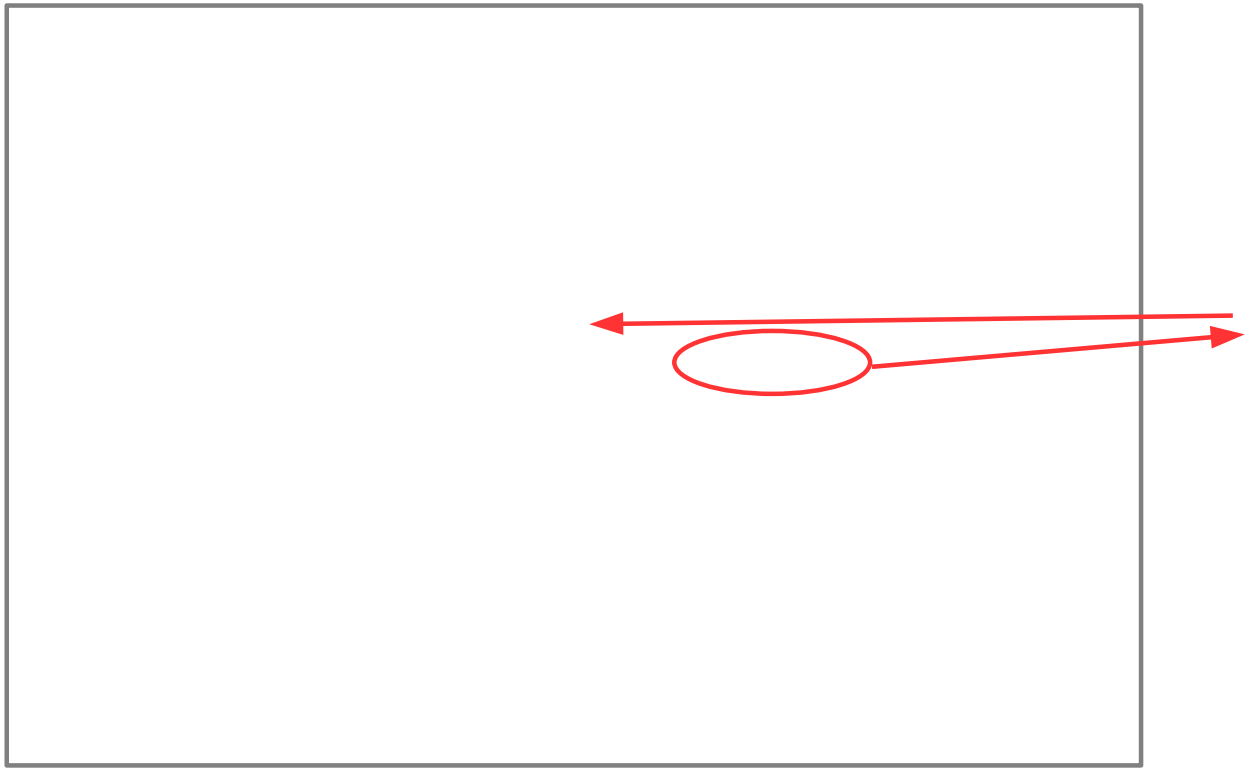
Add helloworld plugin to the feature



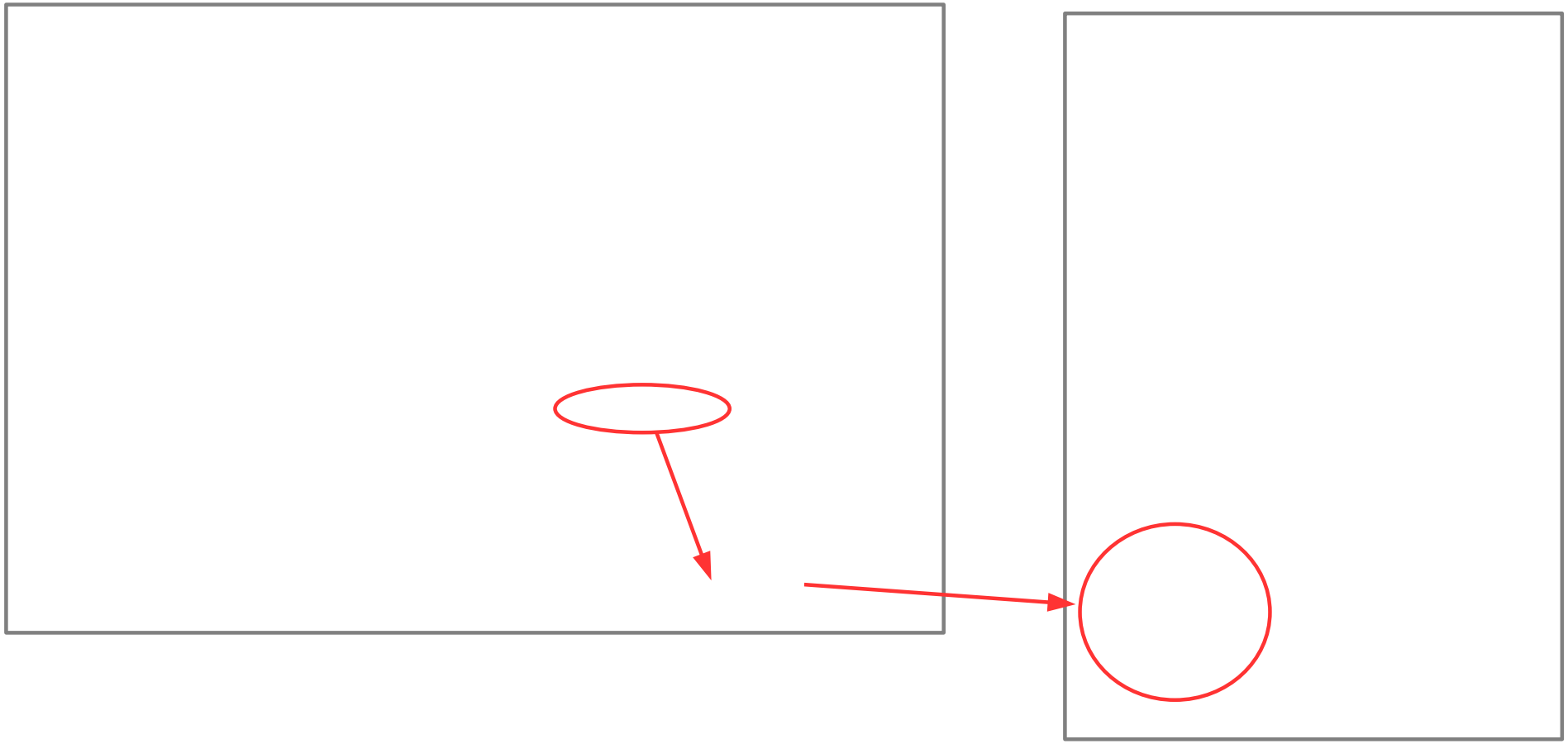
Create the eclipse site project



Add the helloworld feature to the site



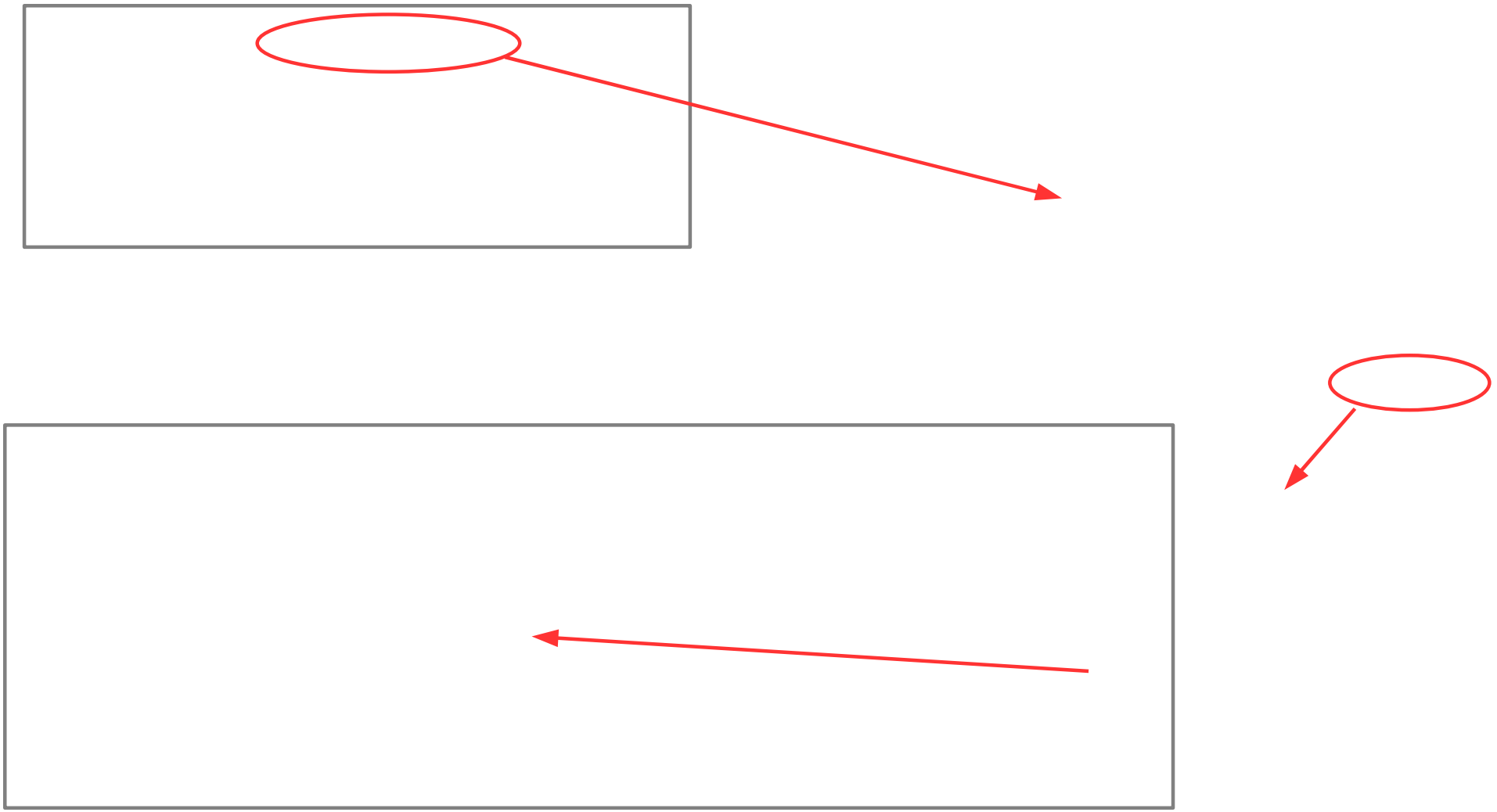
Build the site



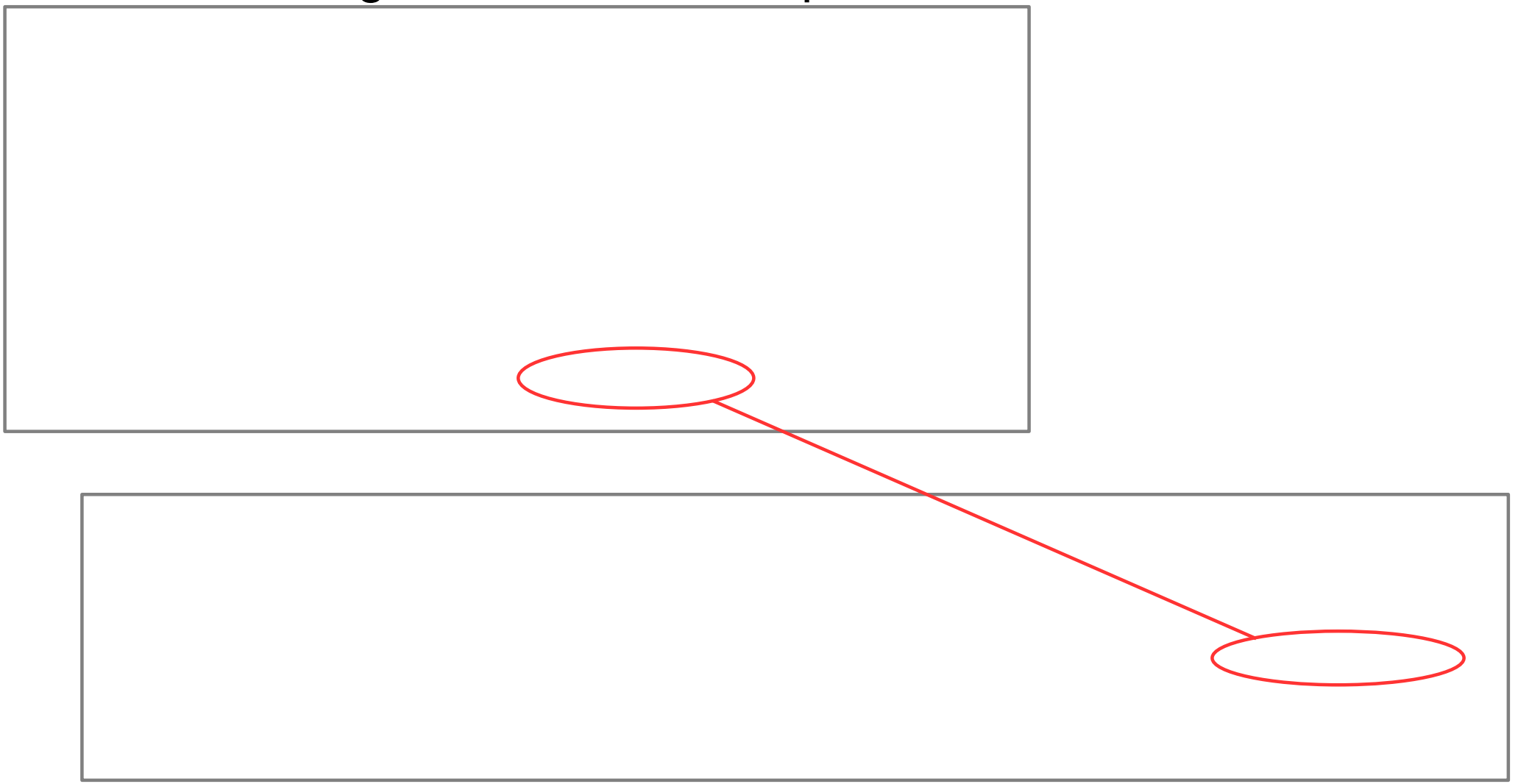
Create an updateSite database from Notes client

- Use File\New Application to create a database on the server that is based on the Eclipse Update Site template

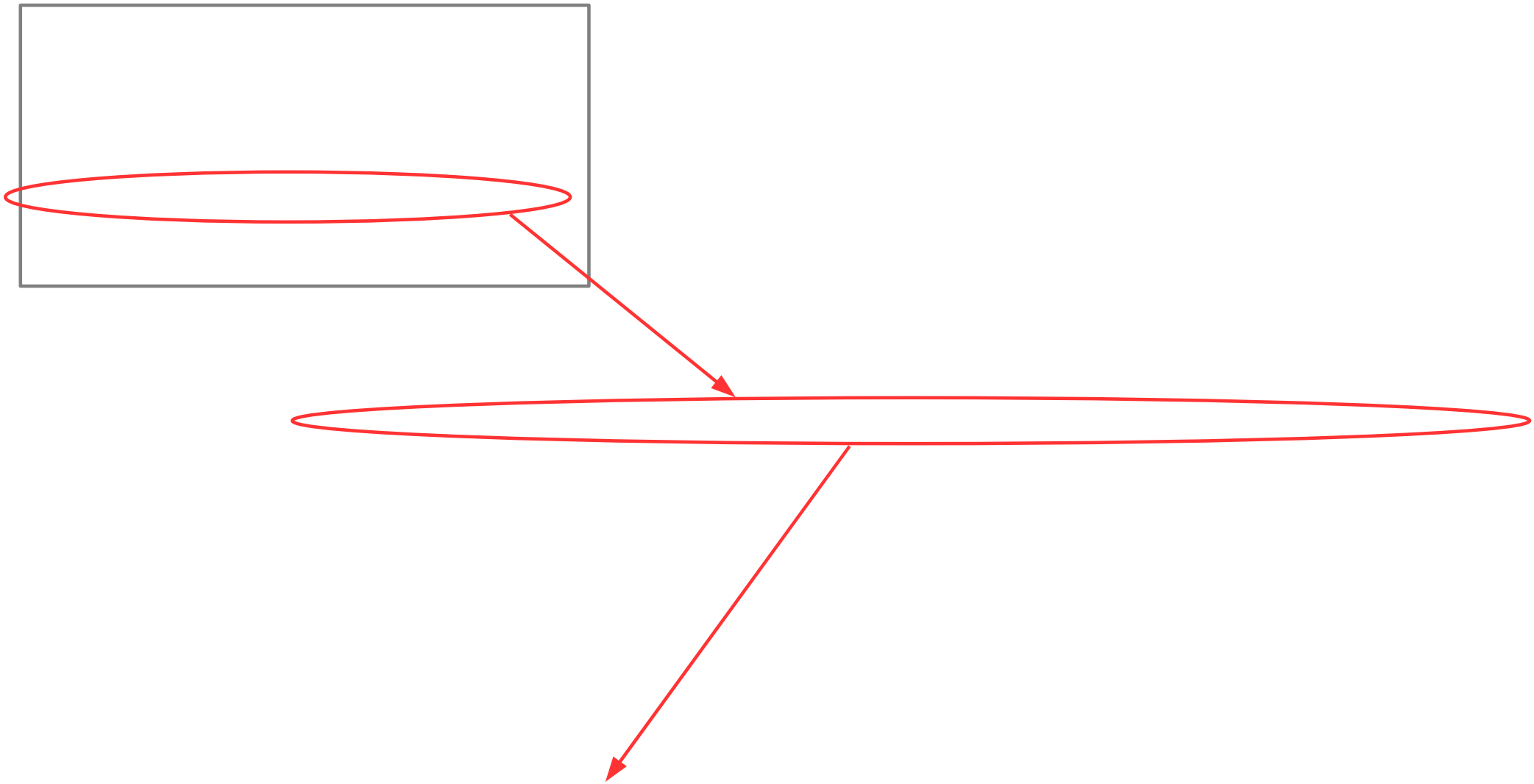
Import local update site



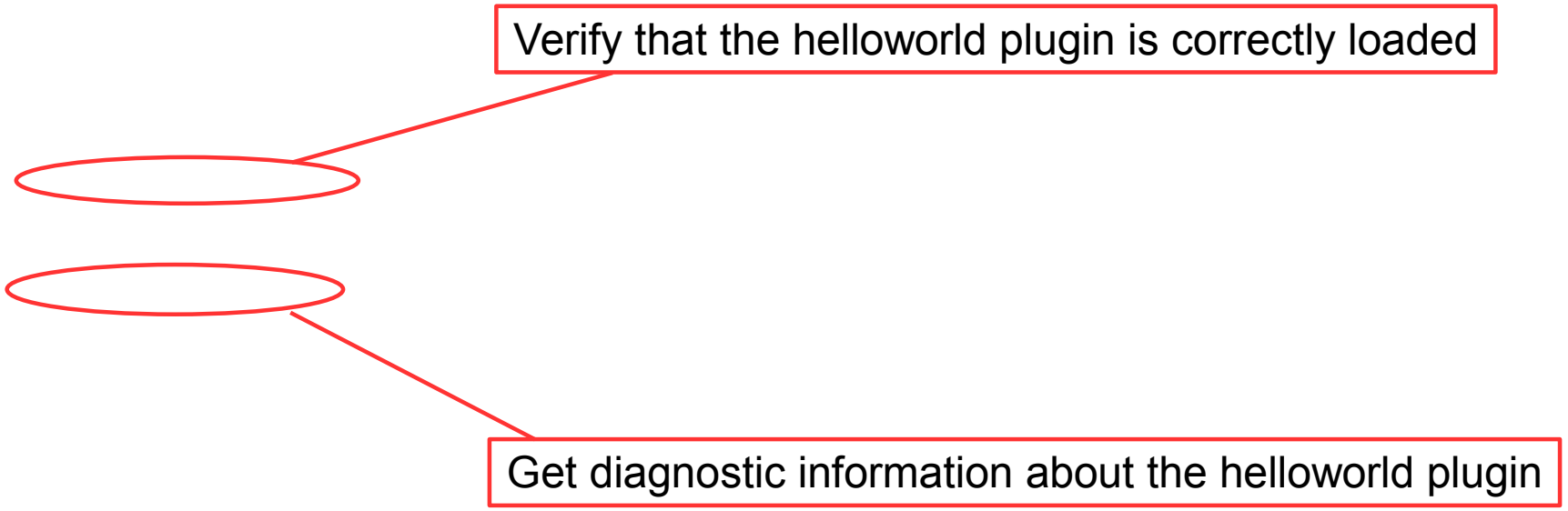
Give admin rights to user who imported the site



Configure the server notes.ini to use the updateSite db

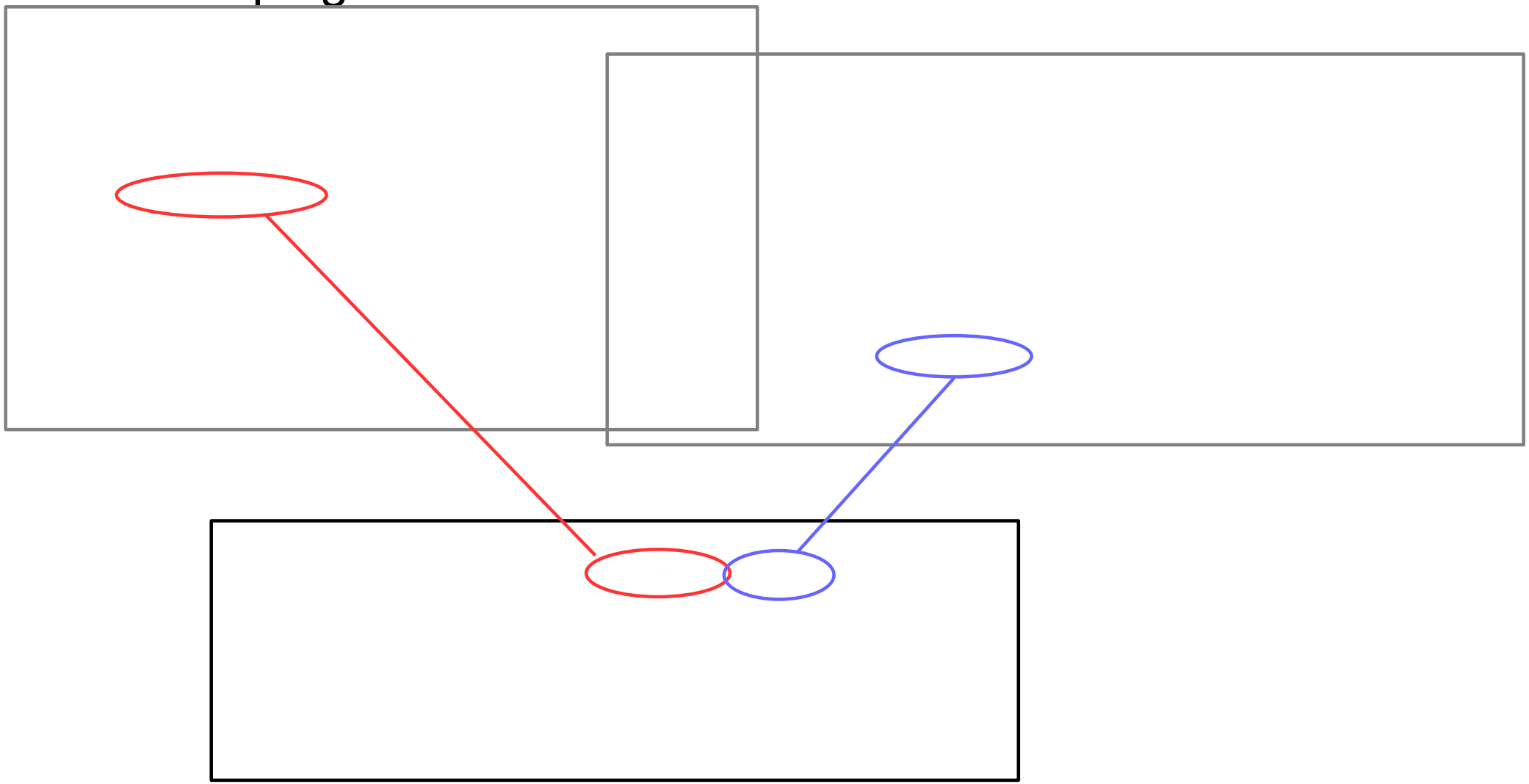


Verify that the plugin is correctly loaded in the server



Many OSGi console commands
- Use tell http osgi help
- Or go to http://www.eclipse.org

Test the plugin from a browser



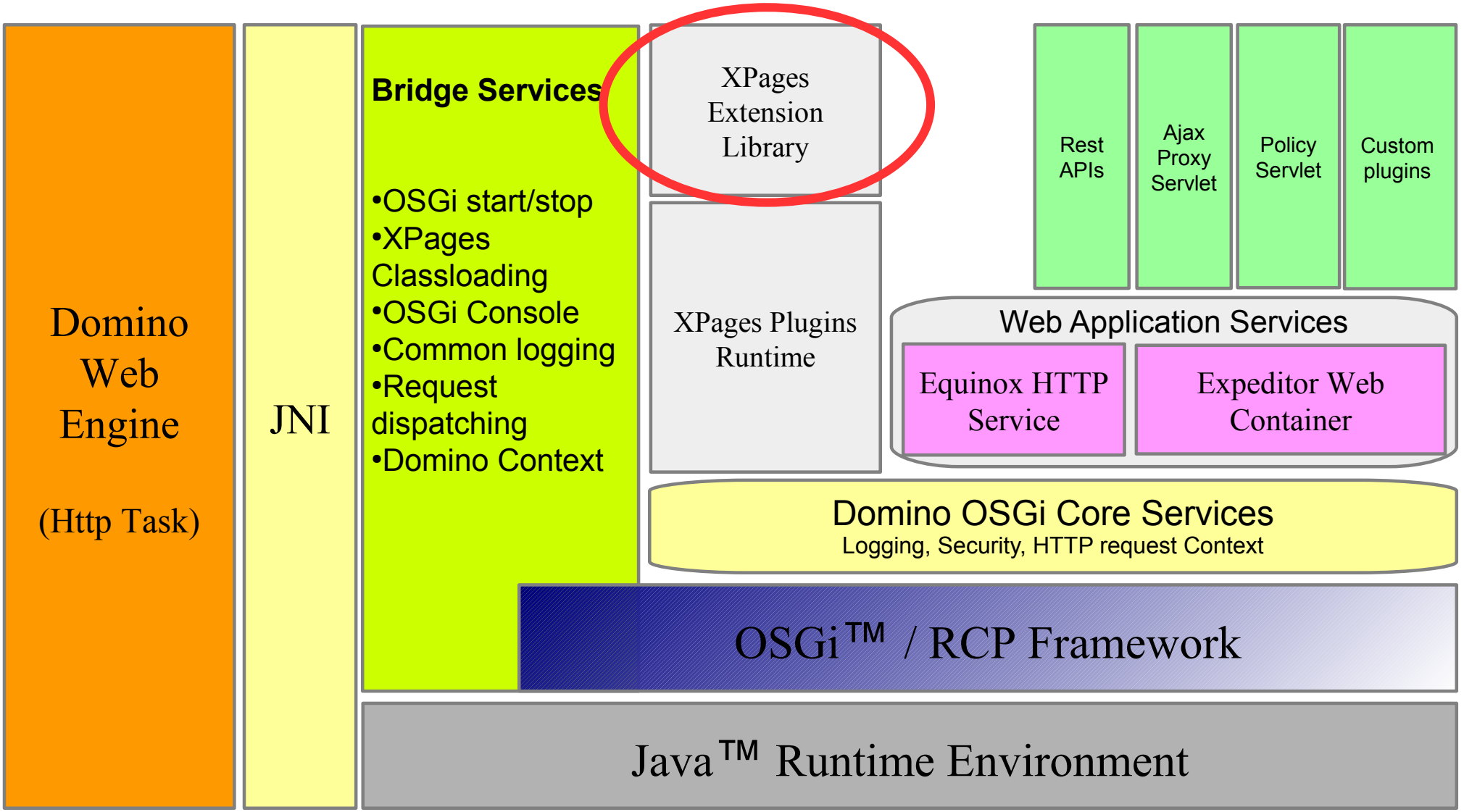
Summary of what we accomplished in this section

- We created a new OSGi plugin that contributes a J2EE web app to the eXpeditor Web Container
- We deployed the plugin using an NSF Based repository
- We tested the plugin in a browser

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Creating a plugin for the XPages Extension Library



Section Goals

- Set up your development environment to work with the XPages Extension Library.
- Create an OSGi plugin that contributes a Hello World custom control to the XPages Extension Library (the Hello World Control simply output a static html DIV tag with a hello World sentence)
- Deploy the plugin to the server using the PDE tool
- Create a feature and update site to be installed in Domino Designer
- Create a sample XPage from Designer using the XPages editor graphic capabilities to use the HelloWorld custom control
- Run the XPage from a browser.

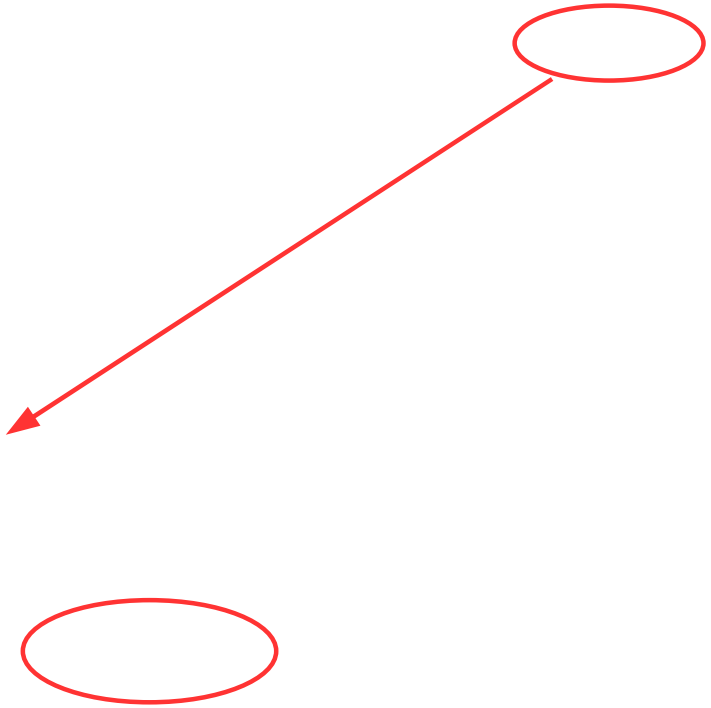
Download the XPages extlib plugins and add them to the target platform.

- Download from openNTF project:
  - http://extlib.openntf.org

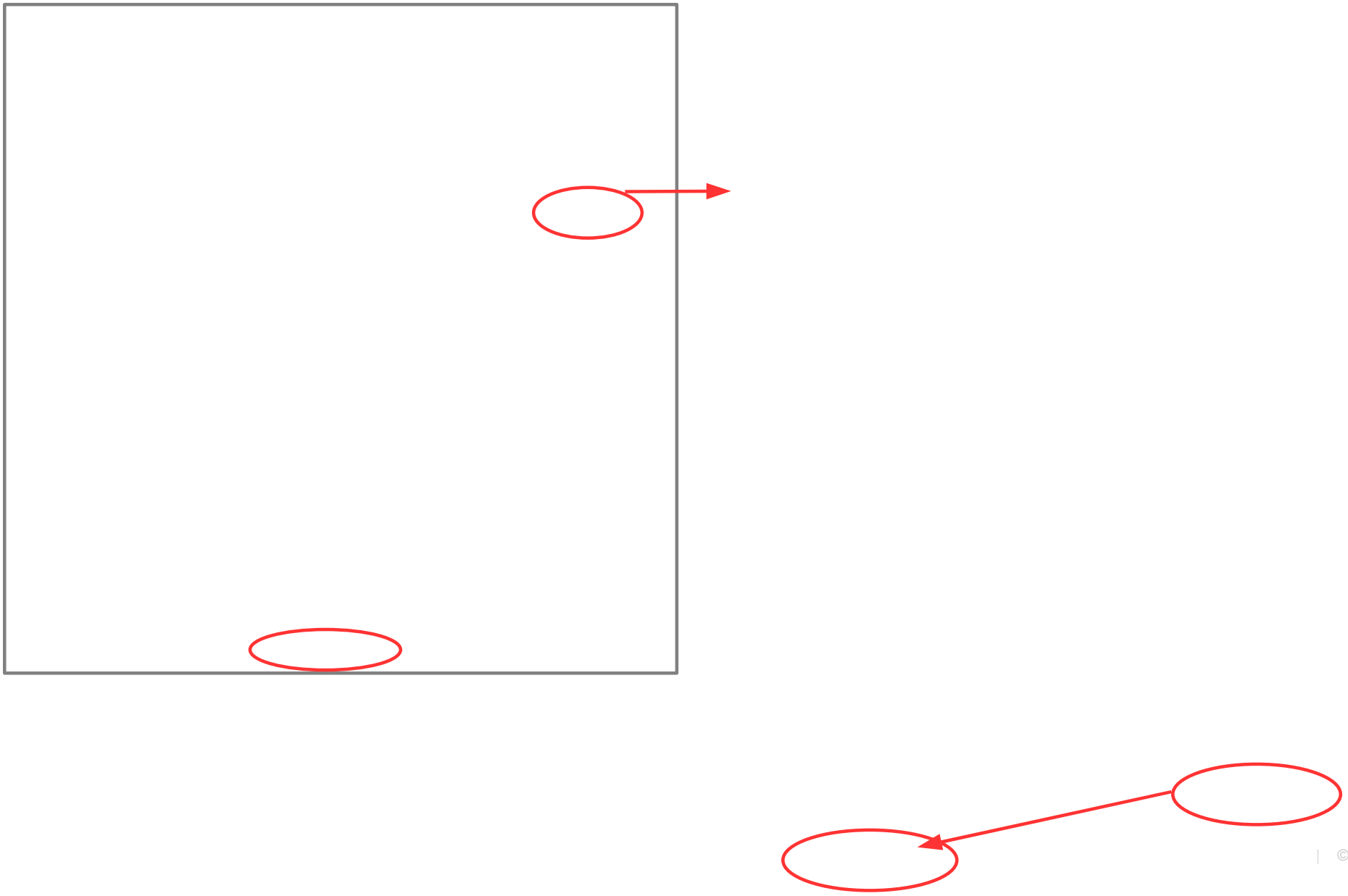
Standard New Plugin Project Wizard

Standard New Plugin Project Wizard(2)

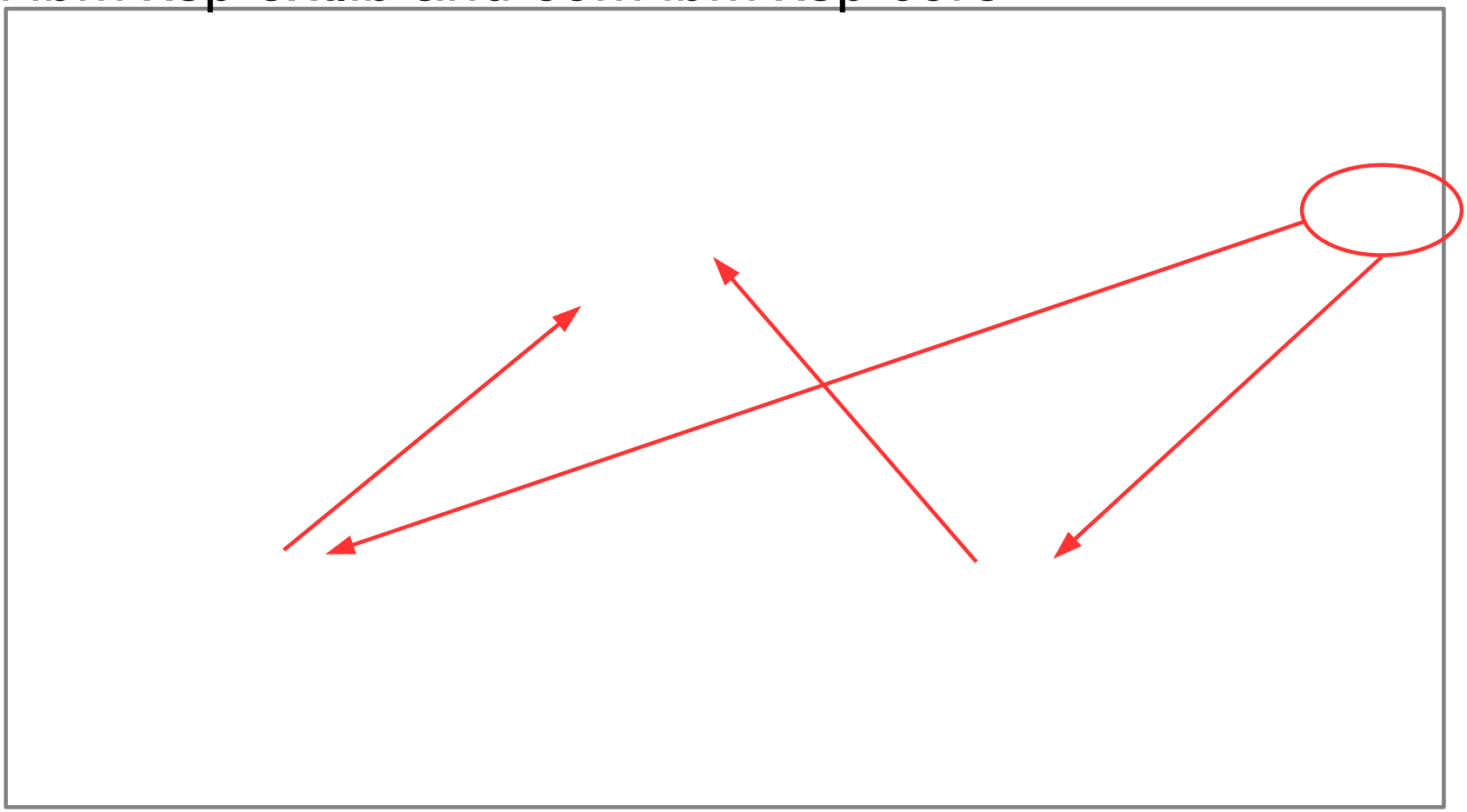
Plugin configuration



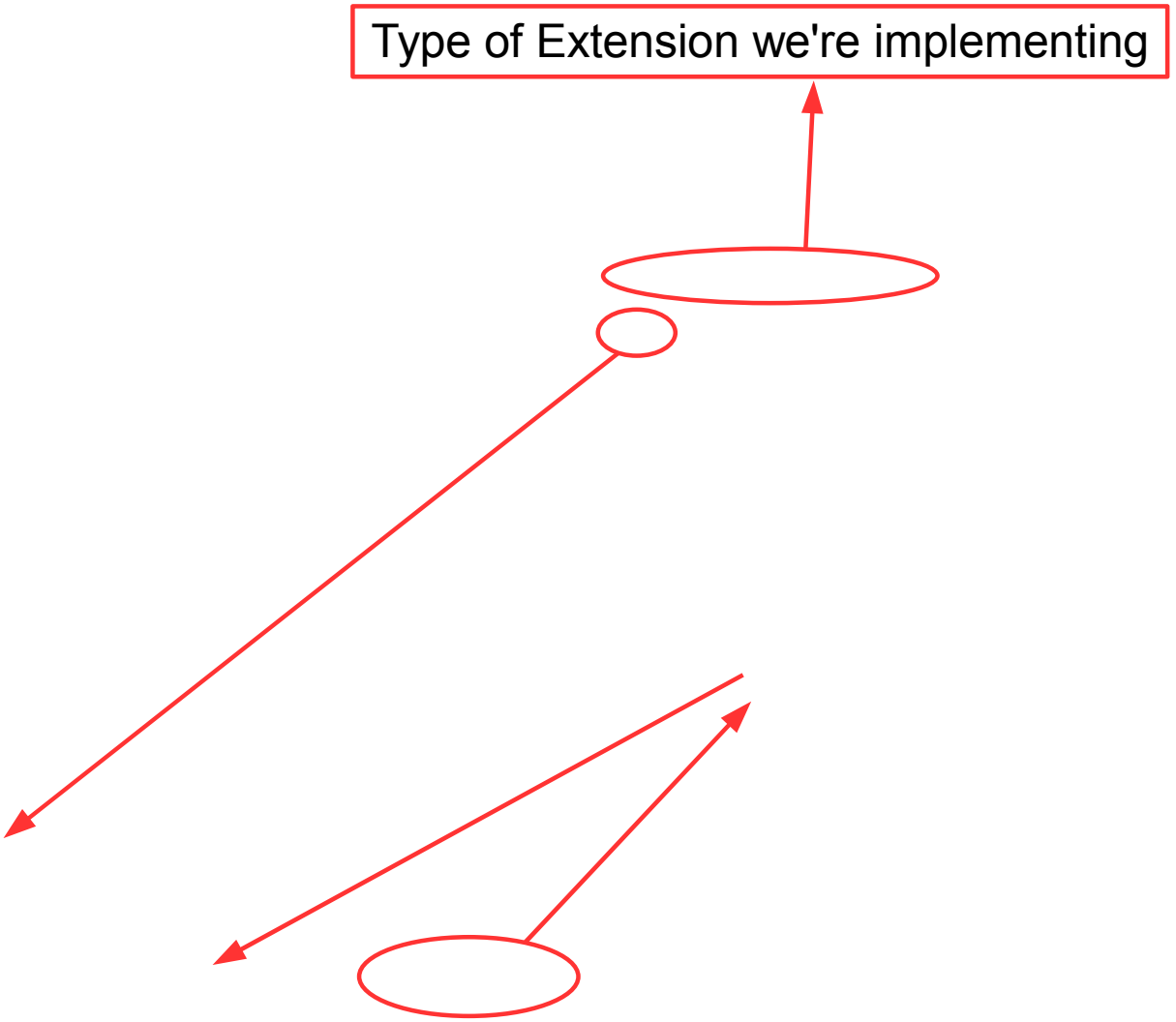
Create the extension point



Update the manifest dependencies add com.ibm.xsp.extlib and com.ibm.xsp.core

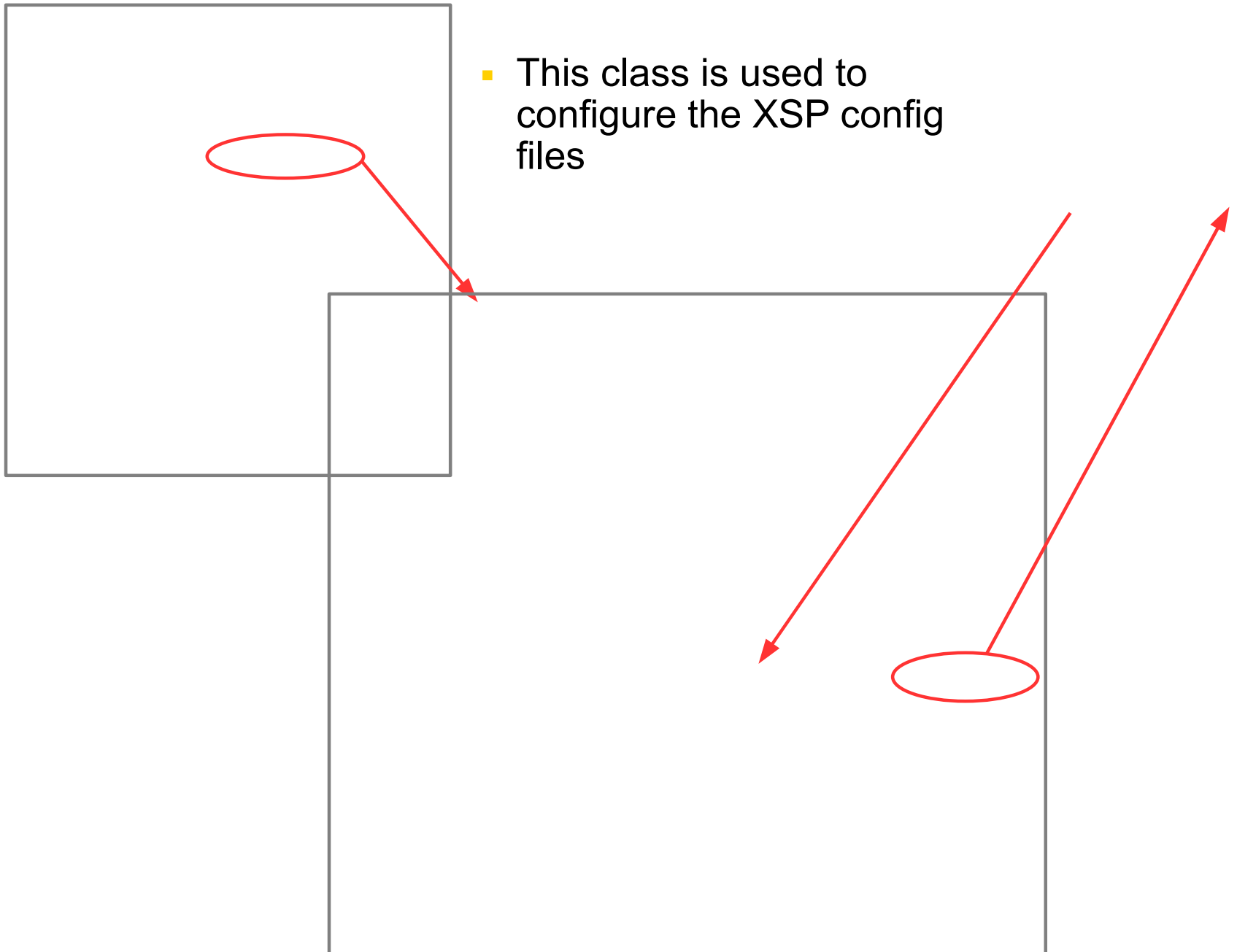


Configure the “com.ibm.commons.Extension” extension point



Create the extension library config class

- This class is used to configure the XSP config files



Wire the xsp xml files to the HelloWorldLibrary class

```
public class HelloWorldLibrary extends AbstractXspLibrary {

	/**
	 *
	 */
	public HelloWorldLibrary() {
	}

	/* (non-Javadoc)
	 * @see com.ibm.xsp.library.XspLibrary#getLibraryId()
	 */
	@Override
	public String getLibraryId() {
		return "com.ibm.ls2012.extlib.helloWorldLib";
	}

	@Override
	public String getPluginId() {
		return "com.ibm.ls2012.extlib.helloWorldControl";
	}

	@Override
	public String[] getXspConfigFiles() {
		return new String[]{
			"META-INF/extlib-helloWorld.xsp-config" // $NON-NLS-1$
		};
	}

	@Override
	public String[] getFacesConfigFiles() {
		return new String[]{
			"META-INF/extlib-helloWorld-faces-config.xml" // $NON-NLS-1$
		};
	}
}
```

Unique id for the library

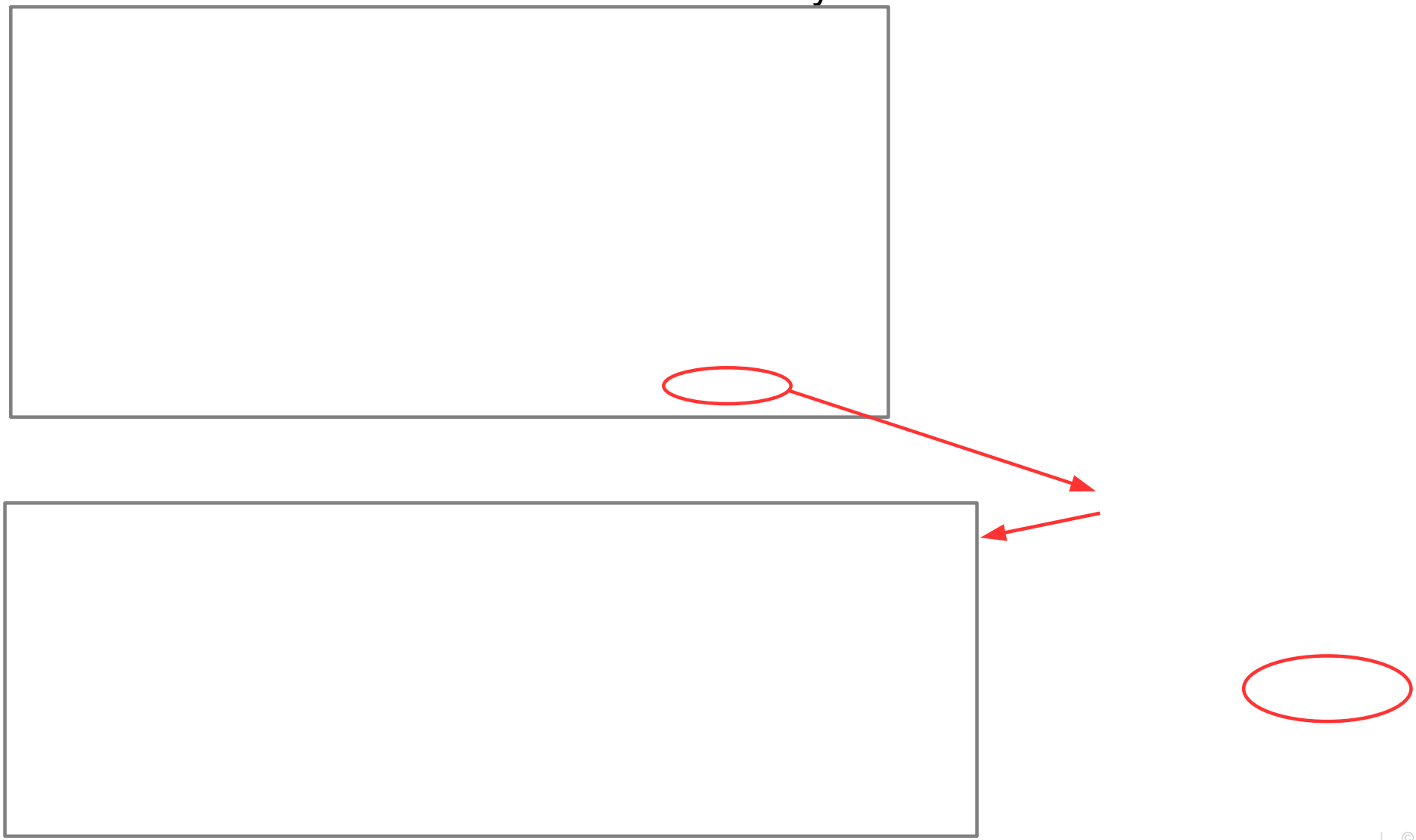
Current Plugin id

List of Xsp configs files for this control

List of Faces configs files for this control

Create the com.ibm.xsp.Library service file

- Under src\META-INF\services directory



Create the META-INF/extlib-helloWorld.xsp-config

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config>
	<faces-config-extension>
		<namespace-uri>http://www.ibm.com/xsp/coreex</namespace-uri>
		<default-prefix>xe</default-prefix>
	</faces-config-extension>

	<component>
		<description>A HelloWorld control for LS2012</description>
		<display-name>HelloWorldLS2012</display-name>
		<component-type>com.ibm.ls2012.extlib.helloworldcontrol.HelloWorldControl</component-type>
		<component-class>com.ibm.ls2012.extlib.helloworldcontrol.HelloWorldControl</component-class>

		<property>
			<description>Sample Property Description</description>
			<display-name>Sample Property</display-name>
			<property-name>sampleProperty</property-name>
			<property-class>java.lang.String</property-class>
			<property-extension>
				<designer-extension>
					<category>basics</category>
				</designer-extension>
			</property-extension>
		</property>

		<component-extension>
			<component-family>com.ibm.ls2012.extlib.hellowworldcontrol</component-family>
			<tag-name>helloWorld</tag-name>
			<designer-extension>
				<in-palette>true</in-palette>
				<category>Extension Library</category>
			</designer-extension>
		</component-extension>
	</component>
</faces-config>
```

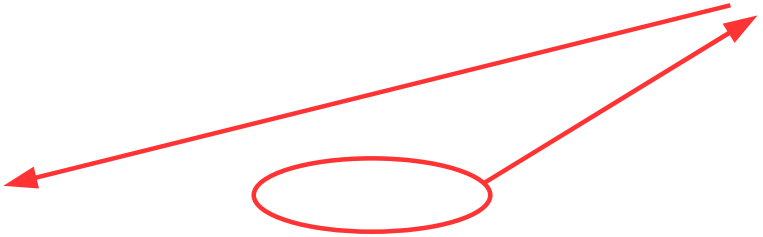
Component class we'll create next

Create the META-INF/extlib-helloWorld-faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config>

  <render-kit>
    <renderer>
      <component-family>com.ibm.ls2012.extlib.hellowworldcontrol</component-family>
      <renderer-type>com.ibm.ls2012.extlib.helloworldcontrol.html_basic_gadget</renderer-type>
      <renderer-class>com.ibm.ls2012.extlib.helloworldcontrol.HelloWorldControlRenderer</renderer-class>
    </renderer>
  </render-kit>

</faces-config>
```

Render class. We'll create one for html

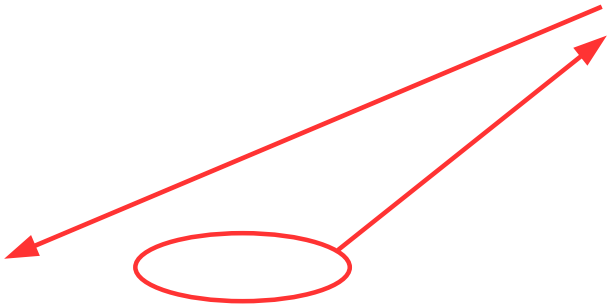
Create the HelloWorldControlRender class



Create the HelloWorldControlRender class (2)

```
public class HelloWorldControlRenderer extends Renderer {

	/**
	 *
	 */
	public HelloWorldControlRenderer() {
	}

	@Override
	public void encodeBegin(FacesContext context, UIComponent component)throws IOException {
		super.encodeBegin(context, component);
		ResponseWriter w = context.getResponseWriter();
		if( AjaxUtil.isAjaxNullResponseWriter(w) ) {
			return;
		}
		w.startElement("div", component);
		w.writeAttribute("style", "border:black solid thick", null);
		w.writeText("HelloWorld Control from LS2012 Show and tell", null);
	}

	@Override
	public void encodeEnd(FacesContext context, UIComponent component)throws IOException {
		super.encodeEnd(context, component);
		ResponseWriter w = context.getResponseWriter();
		if( AjaxUtil.isAjaxNullResponseWriter(w) ) {
			return;
		}
		w.endElement( "div" );
	}

}
```

Create the HelloWorldControl class



Create the HelloWorldControl class (2)

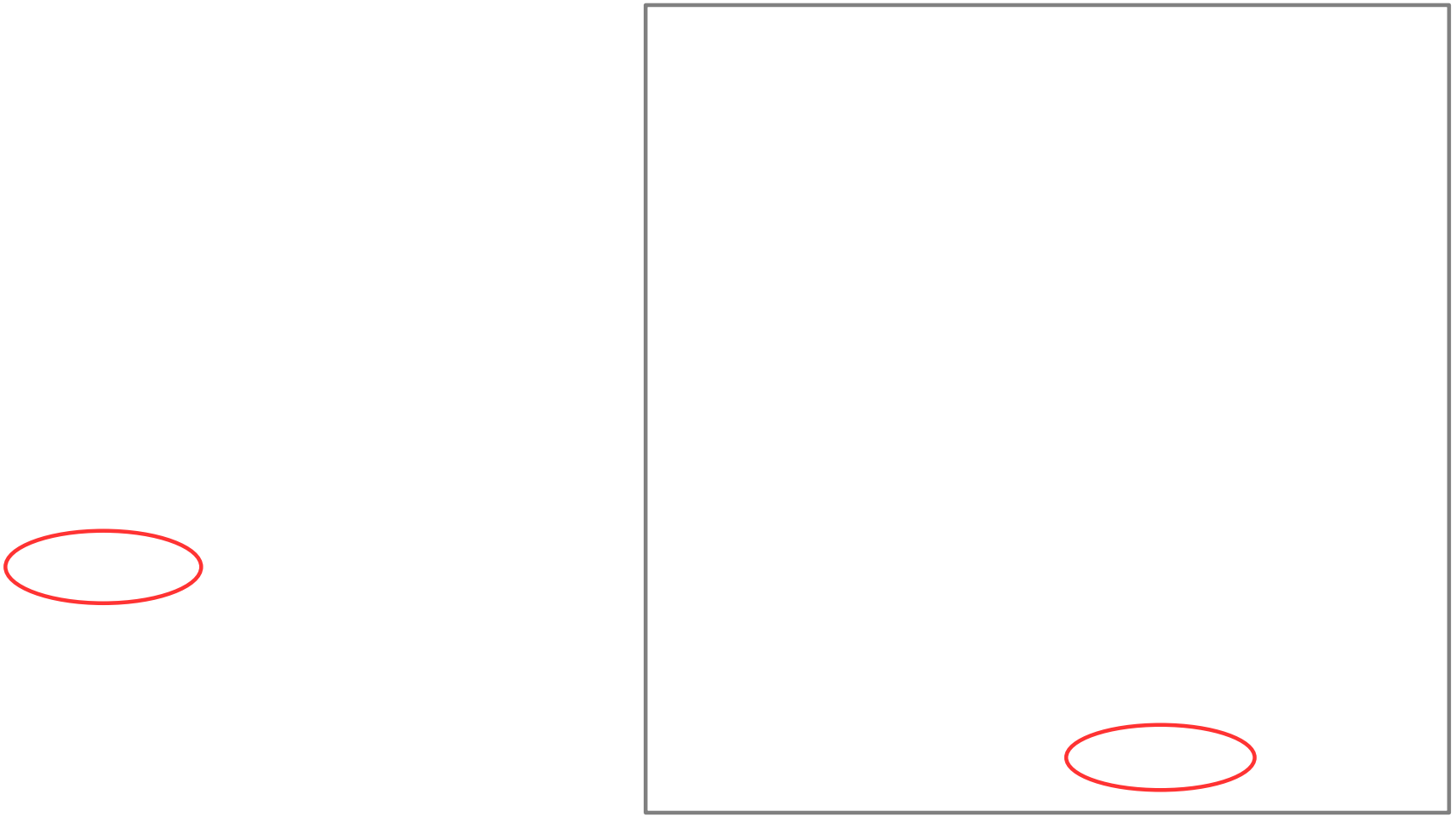
```
public class HelloWorldControl extends UIComponentBase {

	/**
	 *
	 */
	public HelloWorldControl() {
	}

	@Override
	public String getRendererType() {
		return "com.ibm.ls2012.extlib.helloworldcontrol.html_basic.gadget";
	}

	/* (non-Javadoc)
	 * @see javax.faces.component.UIComponent#getFamily()
	 */
	@Override
	public String getFamily() {
		return "com.ibm.ls2012.extlib.hellowworldcontrol";
	}

}
```

Matches the Render type in extlib-helloWorld-faces-config.xml

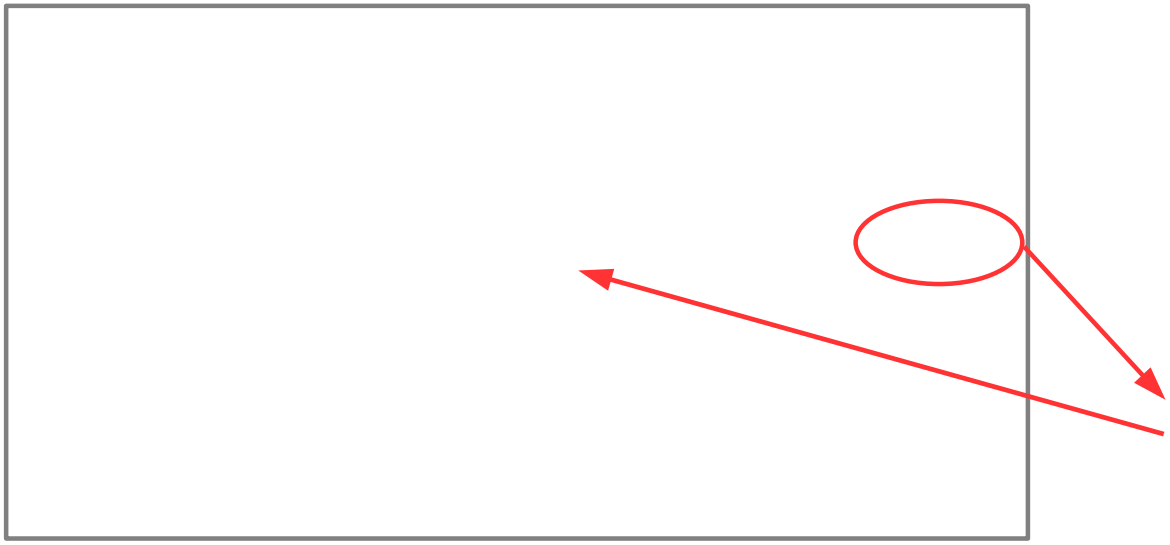
Matches the Component Family in extlib-helloWorld.xsp-configconfig.xml

Create a feature project for the helloWorldControl plugin

- Same procedure already described in the XPD WebContainer plugin section

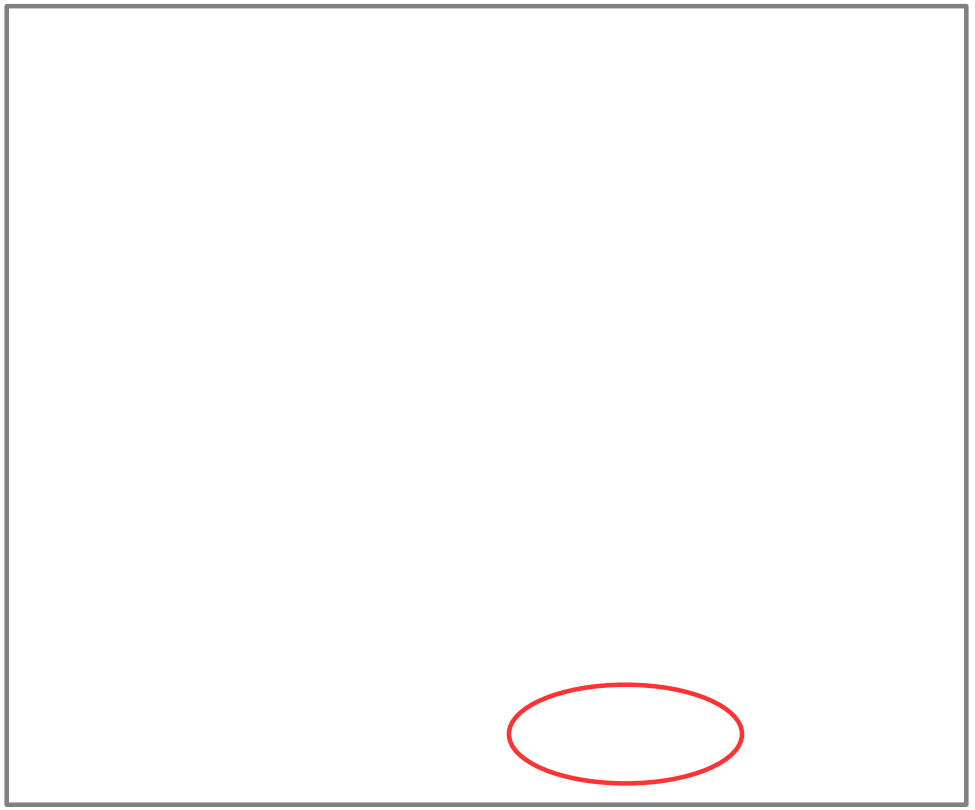


Configure the feature.xml to add the helloWorld plugin

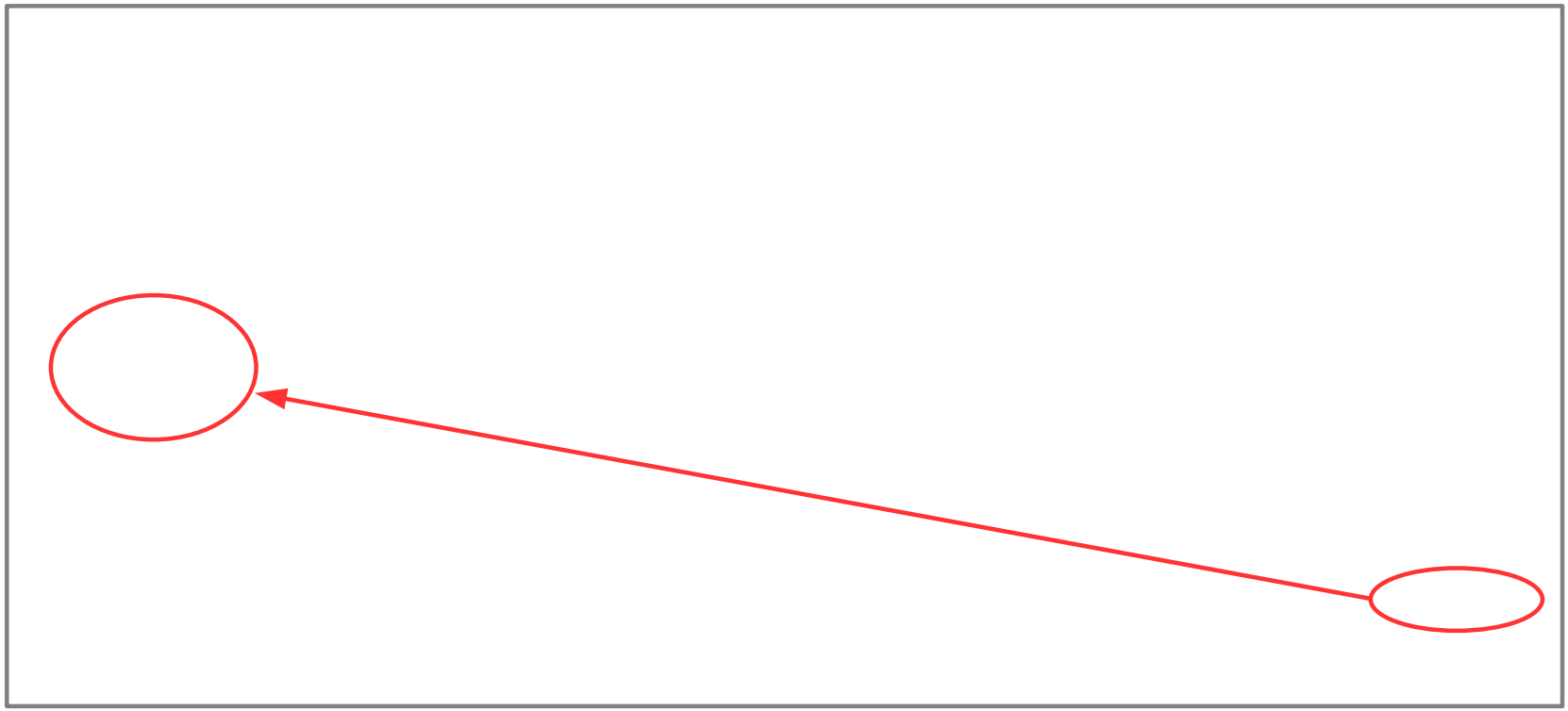


Create a update Site for the helloWorldControl plugin

- Same procedure already described in the XPD WebContainer plugin section



Configure the site.xml to add the helloWorldControl feature

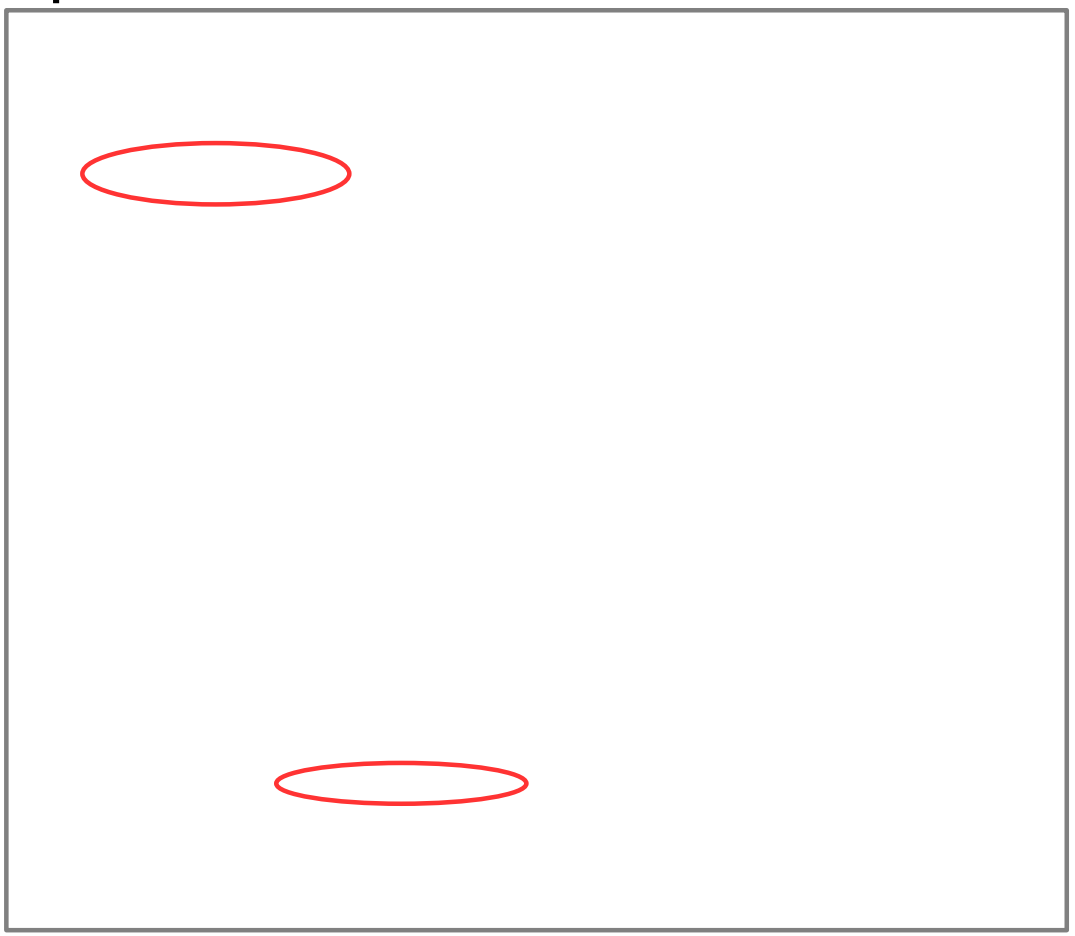


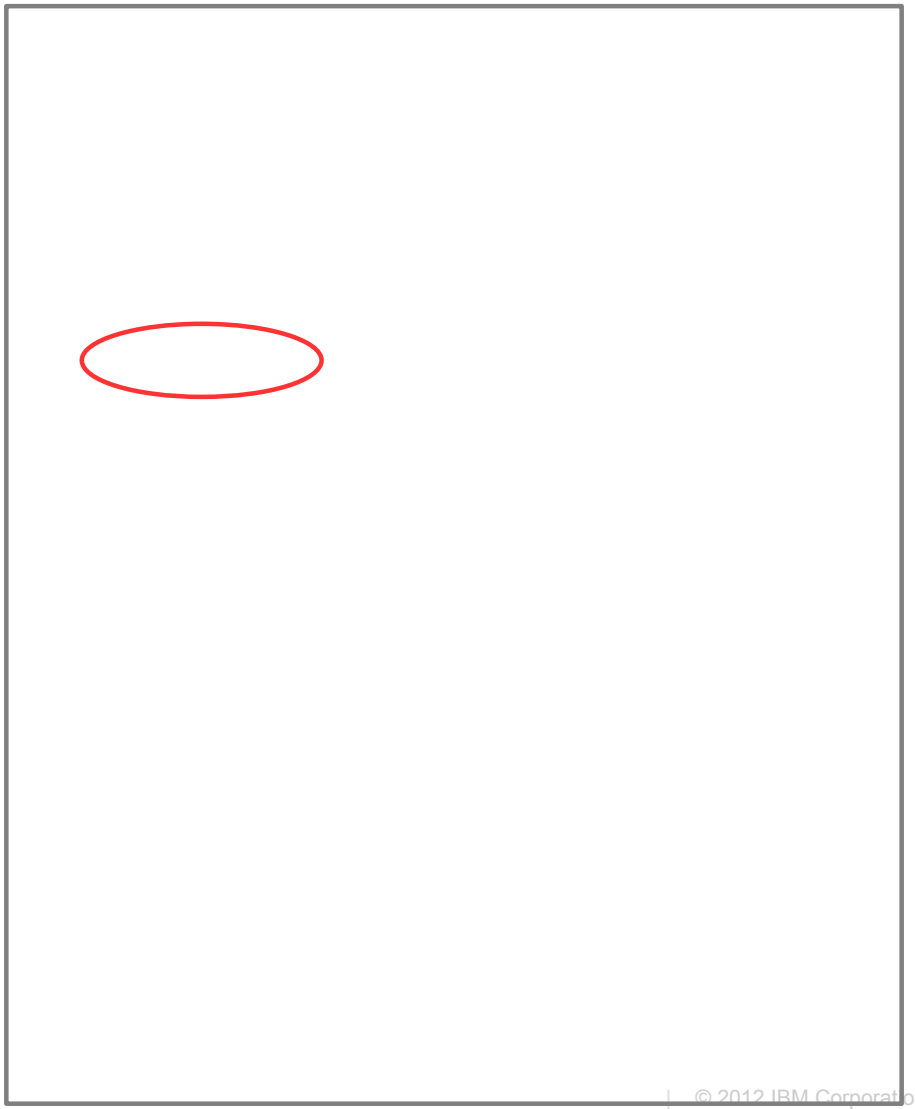
Enable Eclipse plug-in install from Designer preferences

Install HelloWorldControl plugin in Designer

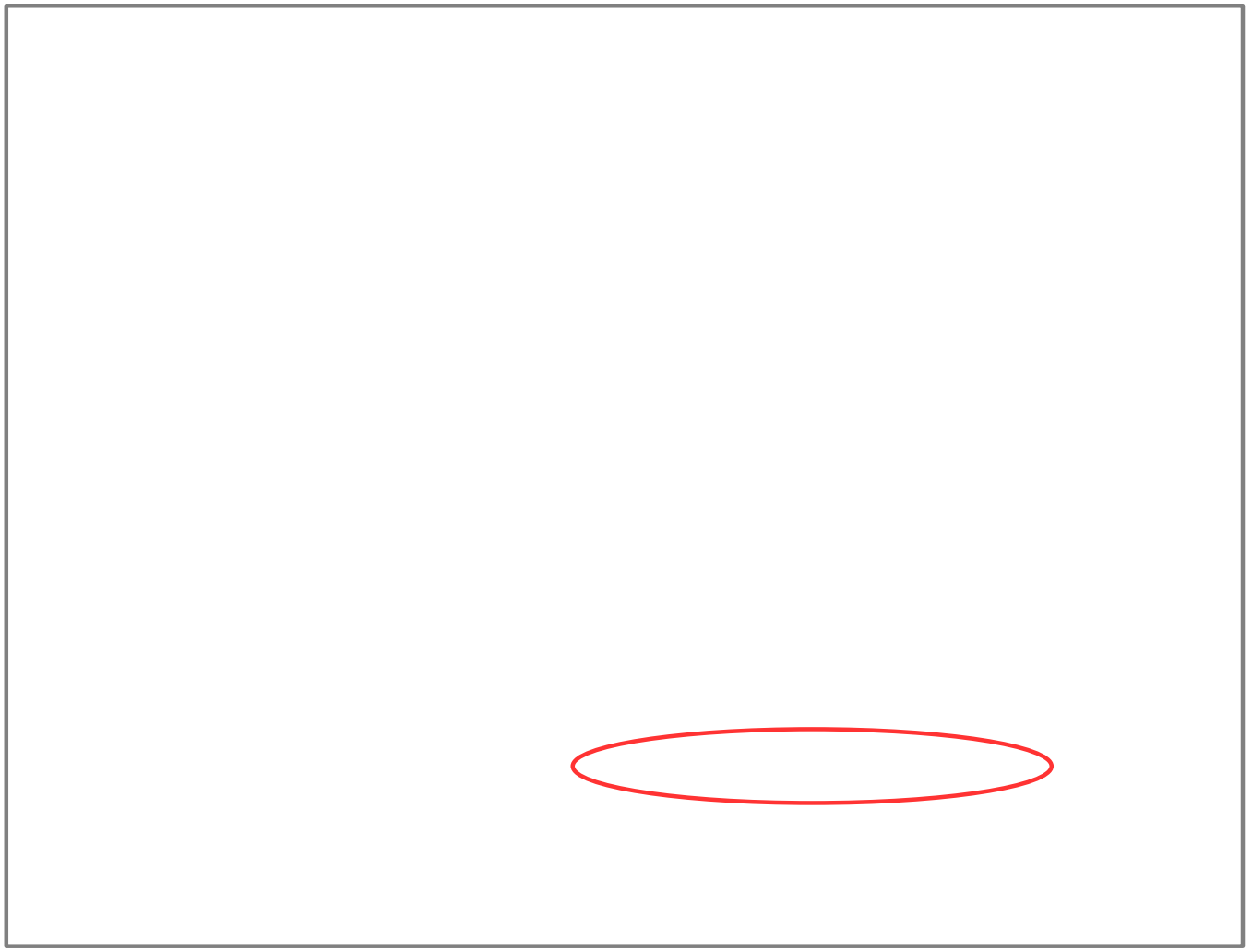
- File\Application\Install...

Create an XPage and select the Control from the palette

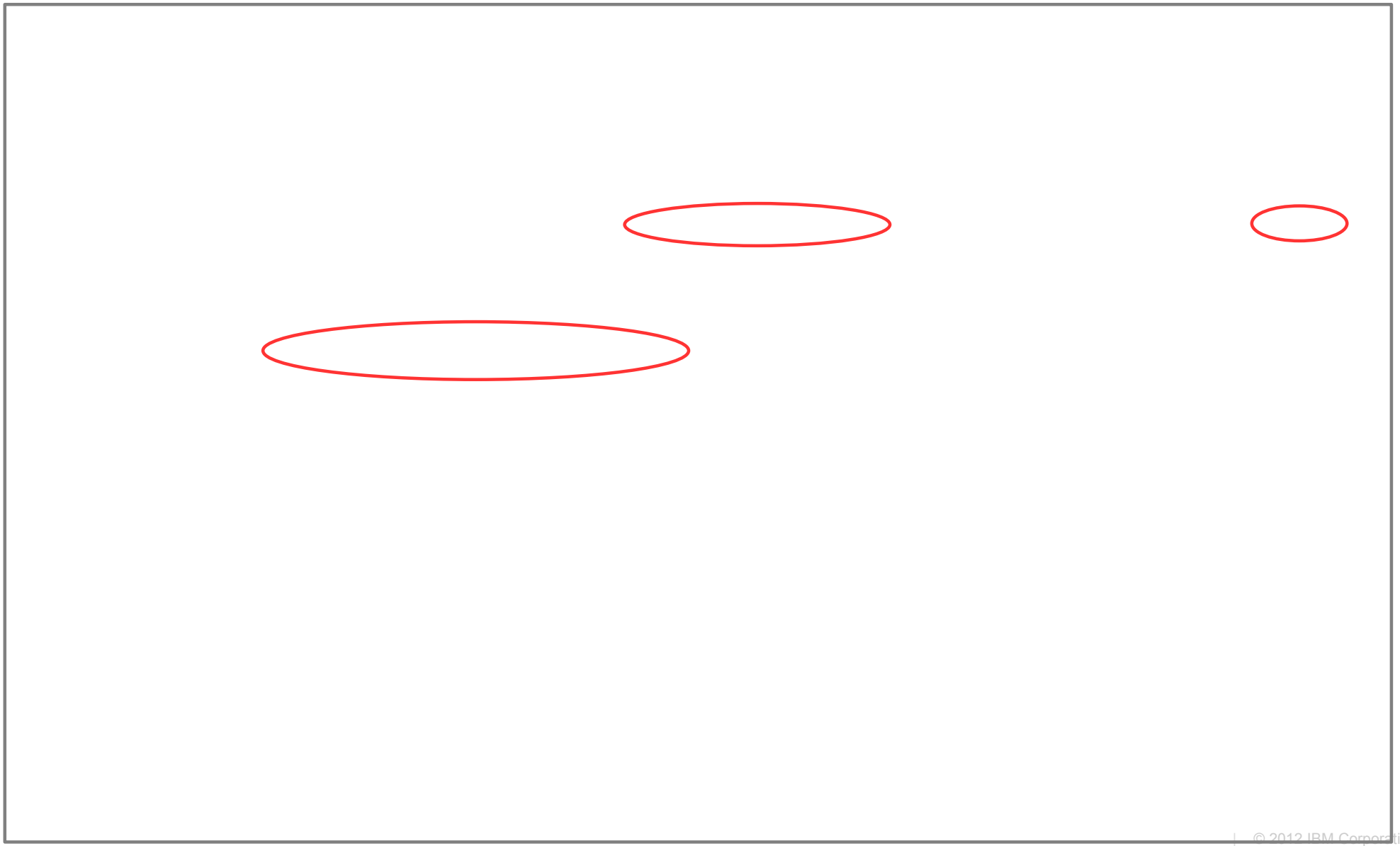




Select the helloWorldLib library in Application Properties page



Back to Eclipse, create an OSGi Launch configuration



Load HTTP task and access the control in browser

```
> l http
12/18/2011 12:24:02 AM  HTTP Server: Using Web Configuration View
12/18/2011 12:24:06 AM  JVM: Java Virtual Machine initialized.
12/18/2011 12:24:06 AM  HTTP Server: Java Virtual Machine loaded
12/18/2011 12:24:06 AM  HTTP Server: DSAPI Domino Off-Line Services HTTP extension Loaded successfully
12/18/2011 12:24:07 AM  HTTP JVM: WARNING: Using pde configuration Http OSGi configuration located in C:\Program Files
(x86)\IBM\Lotus\Domino\data\domino\workspace\pde.launch.ini
12/18/2011 12:24:09 AM  XSP Command Manager initialized
12/18/2011 12:24:09 AM  HTTP Server: Started
>
>
> tell http osgi ss ls2012
12/18/2011 12:24:15 AM  Framework is launched.
12/18/2011 12:24:15 AM  id      State       Bundle
12/18/2011 12:24:15 AM  110     <<LAZY>>    com.ibm.ls2012.extlib.helloWorldControl_1.0.0.qualifier
12/18/2011 12:24:15 AM  146     <<LAZY>>    com.ibm.ls2012.webcontainer.helloworld_1.0.0.201112111542
>
```

Browser

Summary of what we accomplished in this section

- We created a simple new XPages Custom Control that output a static html DIV tag.
- We learned how to deploy it in the Domino Server using the PDE Tool
- We also created an update site to host the plugin and installed it in Domino Designer
- We learned how to use this new control in a sample XPage from Designer using the XPages editor graphic capabilities
- We successfully ran the XPage from a browser.

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Environment setup for DOTS

- In the next section, we'll demonstrate Domino OSGi capabilities for running Server Addin Tasks using the Domino OSGi Tasklet Service (a.k.a. DOTS)
  - We will show how to schedule tasklets to run periodically and how to run them manually on demand.
- This aspect of Domino OSGi is using a different instance from the HTTP OSGi instance we've seen in the previous sections and is also using a different process
- In the next slide, we'll show you how to configure your eclipse development environment to work with the DOTS OSGi instance and also how to configure the PDE tool accordingly

Environment setup for DOTS

- Installing DOTS
  - Required Eclipse plug-ins and DOTS core files
- PDE Tool
  - The version installed during the HTTP environment setup will persist across workspaces so there is no action needed. On a clean setup you would follow the steps previous demonstrated in the HTTP environment setup.
- Target Platform
  - DOTS is just another OSGi setup so it's almost the same as the HTTP setup with a few minor differences.
- Notes.jar
  - This plug-in is just used for development and will be exactly the same as the version in the HTTP environment. Simply import from the other workspace. On a clean setup you would follow the steps previous demonstrated in the HTTP environment setup.

Installing DOTS (Required Eclipse plug-ins)

- The OSGi framework is located in {DominoBin}/osgi-dots/rcp/eclipse
- The minimum required set of plug-ins is comprised of the following list:
  - org.eclipse.core.contenttype
  - org.eclipse.core.jobs
  - org.eclipse.core.runtime.compatibility.auth
  - org.eclipse.core.runtime
  - org.eclipse.equinox.app
  - org.eclipse.equinox.common
  - org.eclipse.equinox.preferences
  - org.eclipse.equinox.registry
  - org.eclipse.osgi
  - org.eclipse.update.configurator
- You should be able to simply copy these 10 plug-ins into the {DominoBin}/osgi-dots/rcp/eclipse/plugins directory from any installed version of the Eclipse SDK.

Installing DOTS (DOTS core files)

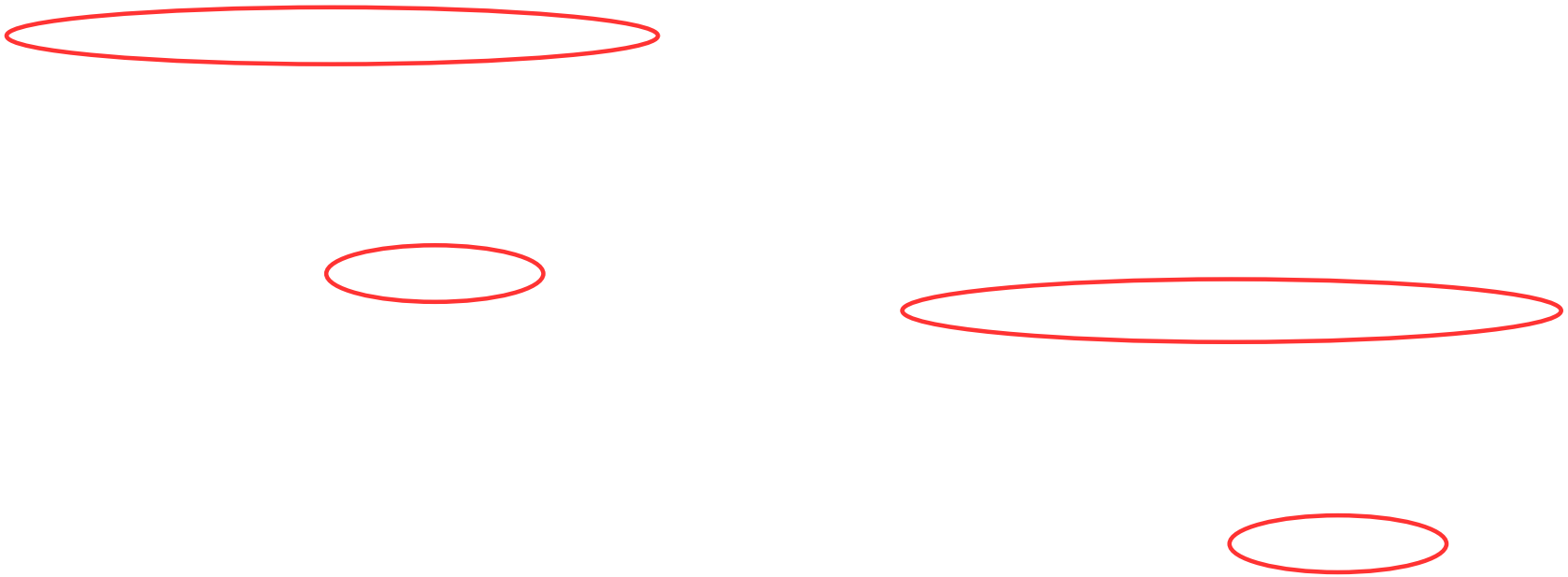
- Download the OSGI Tasklet Service for IBM Lotus Domino from OpenNTF
- The following files must be installed in the {DominoBin} directory
  - ndots.exe (Windows)
  - dots, dots.res (UNIX)
- The following files must be installed in the {DominoBin}/osgi-dots directory
  - launcher.jar
  - dotssec.jar
- The following files must be installed in the {DominoBin}/osgi-dots/shared/eclipse/plugins directory
  - com.ibm.dots_2.0.3.XXXX.jar
  - com.ibm.dots.samples_2.0.3.XXXX.jar (optional)

Environment setup for DOTS (PDE tool recap)

- An easy verification that the PDE Tool is installed is to check the OSGi Frameworks available to Eclipse
- You should see 3 frameworks
  - Domino OSGi Framework
  - Domino tasklet Framework (dots)
  - Equinox

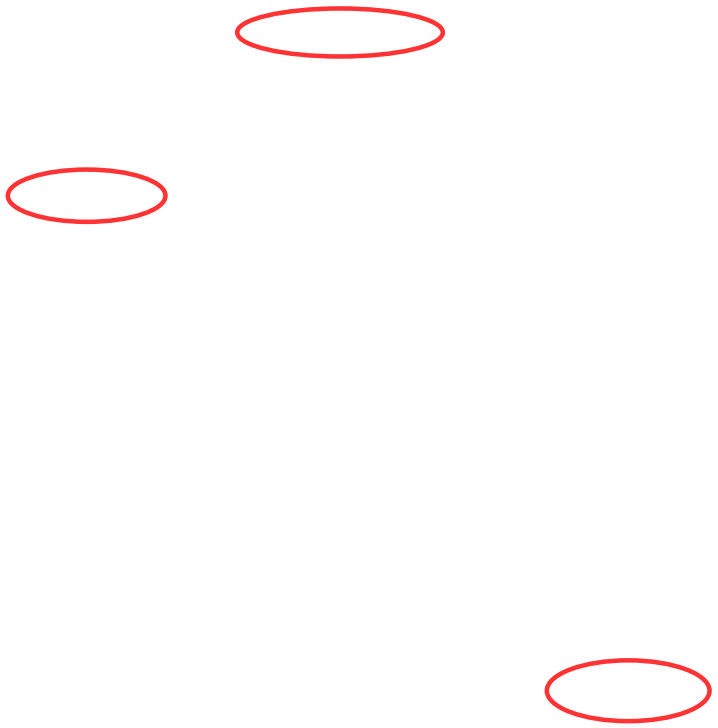
Environment setup for DOTS (Target Platform recap)

- Add the {DominoBin}\osgi-dots\rcp\eclipse directory and click the Finish button
- Repeat the process to add the {DominoBin}\osgi-dots\shared\eclipse directory



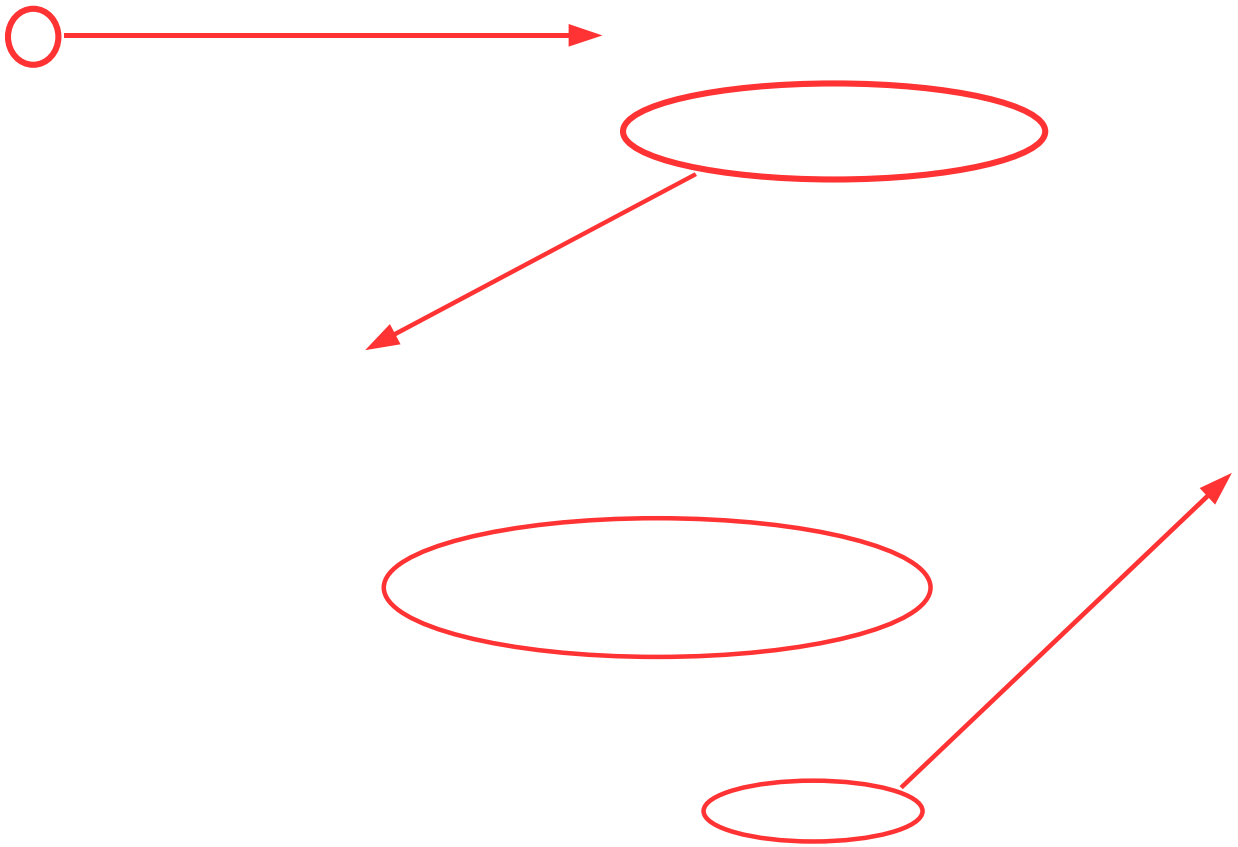
Environment setup for DOTS (Target Platform recap)

- Select the new Target definition you just created and click the OK button



Environment setup for DOTS (Notes.jar recap)

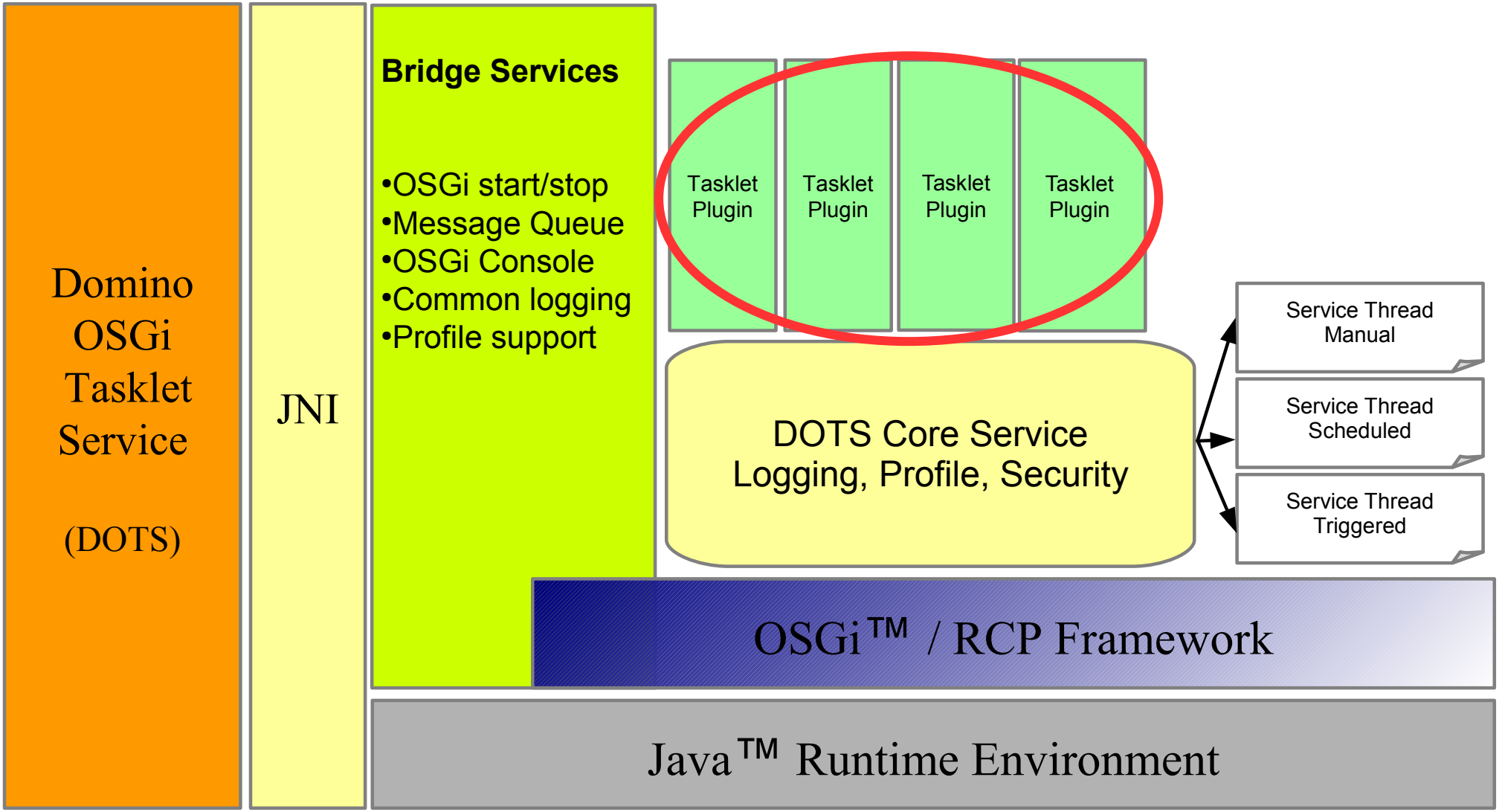
- Use the “Create Notes Java Api Project” option from the PDE tool



Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Creating DOTS tasklets



Section goals

- Create an OSGi plugin that contributes tasklets to the DOTS process
  - Schedule the tasklet using extension point and java annotation
  - Run the tasklet manually on demand using the “run” OSGi console command
- Show how to run the DOTS process in debug mode
- Deploy and debug the tasklet using the PDE Tool configured for DOTS

Creating DOTS tasklets

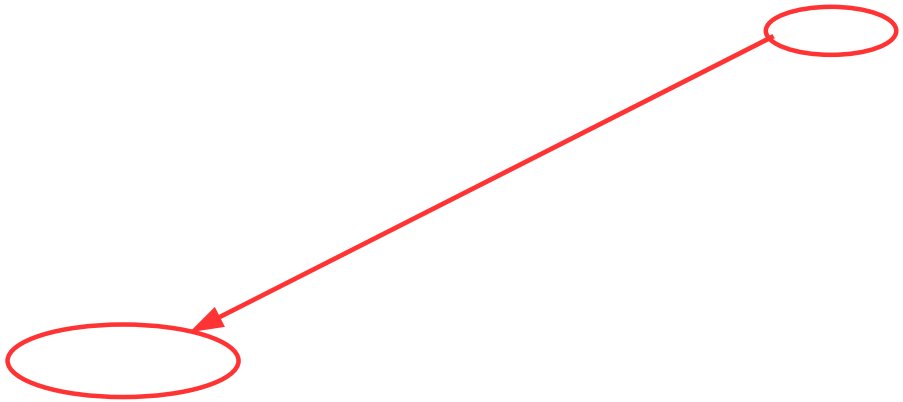
- Create a new workspace plug-in
  - File->New->Plug-in-Project

Creating DOTS tasklets

- Project Name: com.ibm.ls2012.dots
  - Select “an OSGi framework” radio button under the Target Platform section
  - Click on the Next button
- On the Content screen change the name to DOTSDemo
  - Click on the Finish button

Creating DOTS tasklets

- Go to the Overview tab, under Extension / Extension Point Content and click on Extensions
- Click Yes on the Extensions pages hidden screen

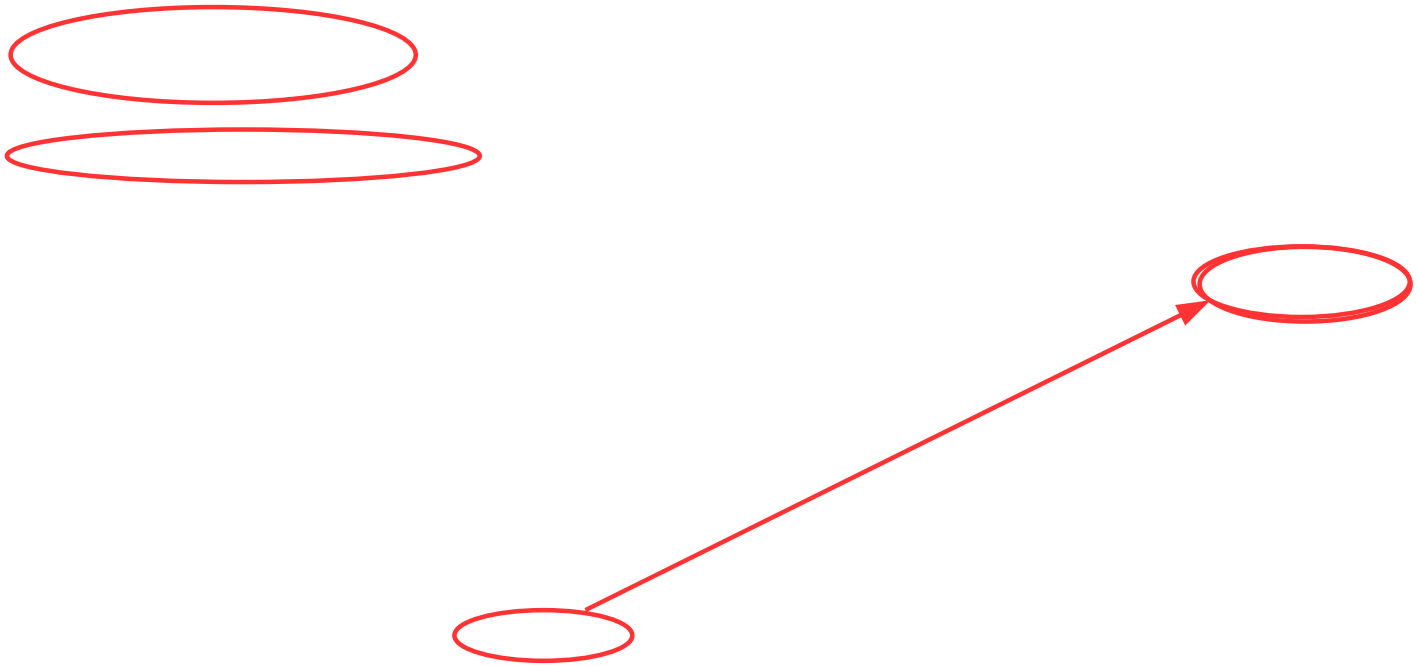


Creating DOTS tasklets

- Go to the Extensions tab and click the Add button...

Creating DOTS tasklets

- Deselect the “Show only extension points from the required plug-ins”
  - Select the com.ibm.dots.task Extension Point
  - Click on the Finish button
- On the New plug-in dependency screen, click the Yes button



Creating DOTS tasklets

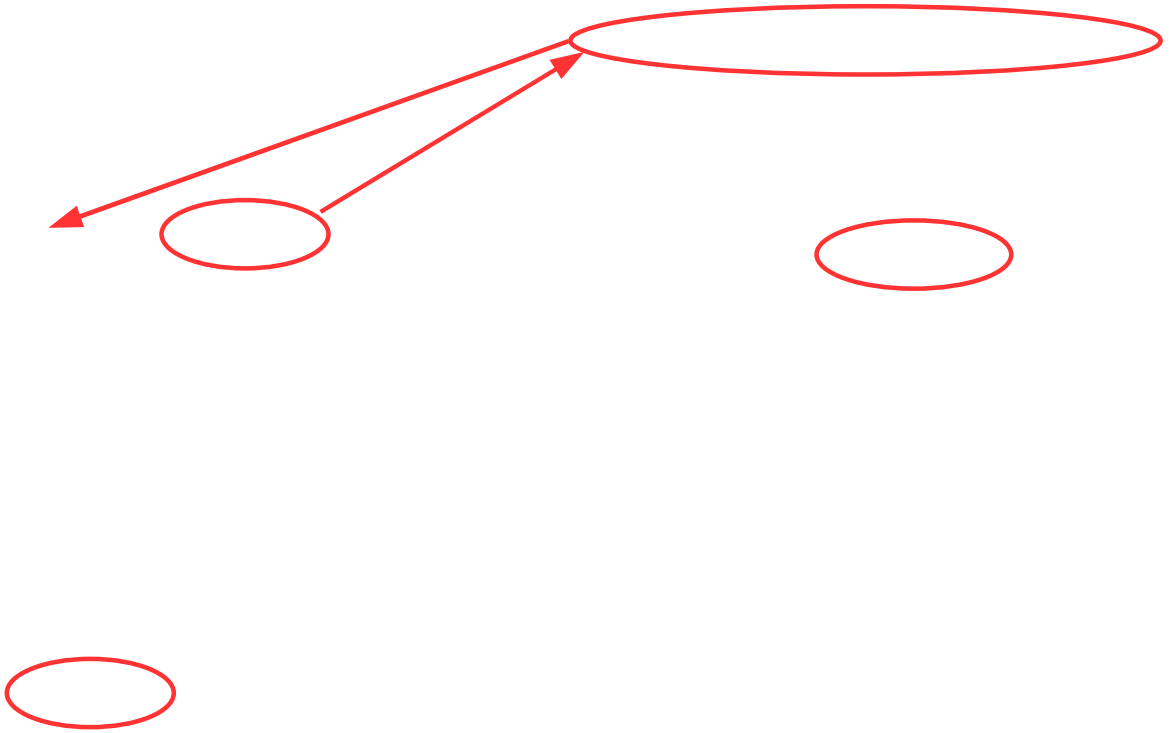
- Go to the MANIFEST.MF tab and click on the error and fix the singleton

Creating DOTS tasklets

- Right click on the com.ibm.ls2012.dots package and click New->Class

Creating DOTS tasklets

- Create a new Java class named MyScheduledTask extending com.ibm.dots.task.AbstractServerTaskExt



Creating DOTS tasklets

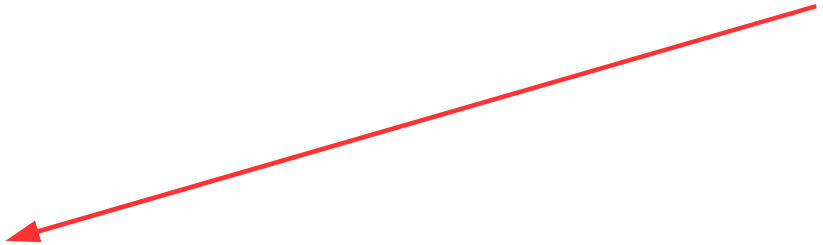
- In the MyScheduledTask class add unimplemented methods

Creating DOTS tasklets

- Resolving some of the warnings and errors displayed
  - On the Dependencies tab of MANIFEST.MF add lotus.domino to the list of Imported Packages

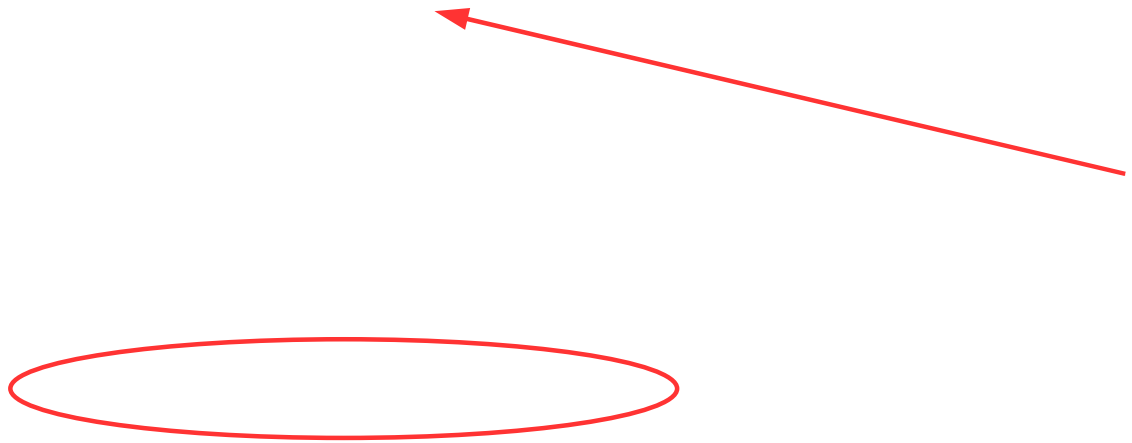
Creating DOTS tasklets

- Scheduling of the MyScheduledTask tasklet via plugin.xml



Creating DOTS tasklets

- Scheduling of the MyScheduledTaskAnnotated tasklet via annotations
  - Custom method using annotations
  - Tasklet still needs to be registered in plugin.xml



Creating DOTS tasklets

- MyManualTask tasklet with the final plugin.xml

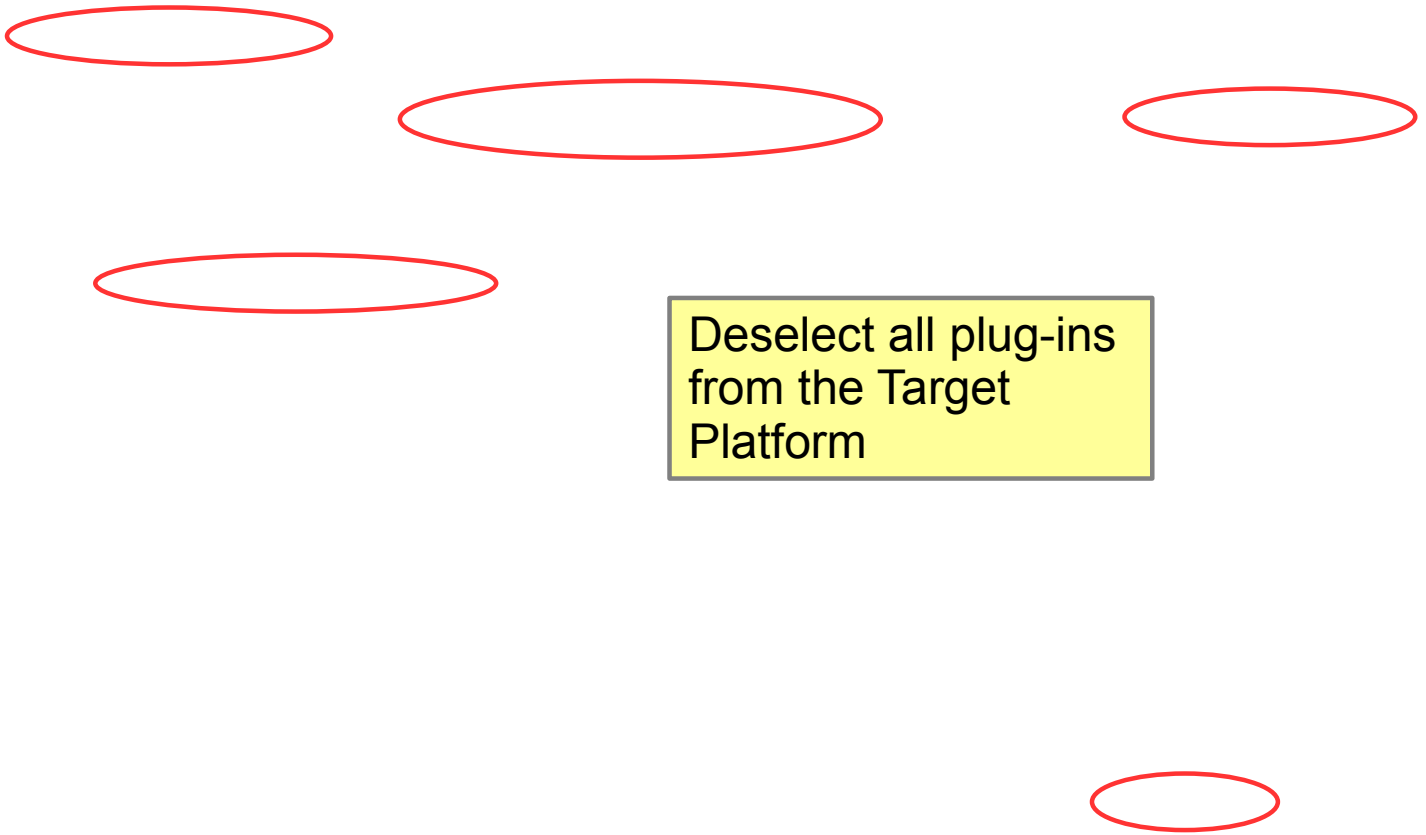
Deploying & debugging a DOTS tasklet

- Click on Run->Debug Configurations...

Deploying & debugging a DOTS tasklet

- Create a new OSGi Framework

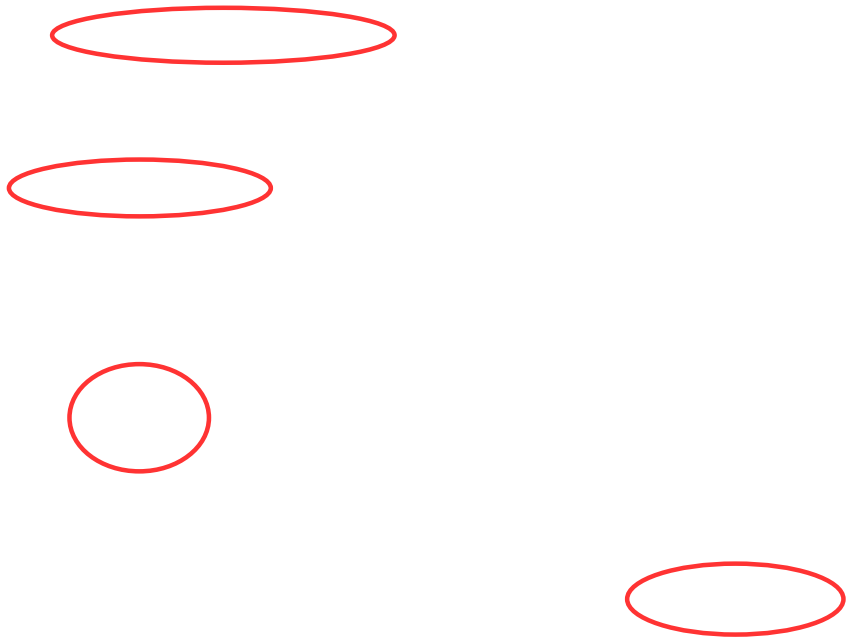
Deploying & debugging a DOTS tasklet



Deploying & debugging a DOTS tasklet

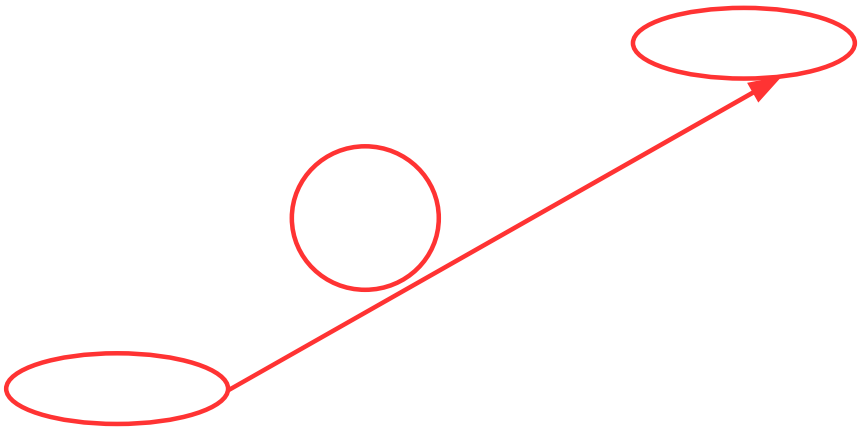
- Create a new Remote Java Application

Deploying & debugging a DOTS tasklet



Deploying & debugging a DOTS tasklet

- Click on the DOTS OSGi Configuration and then click the Debug button
  - This launches the PDE Tool
  - Select the appropriate Domino Binary & Data directories
  - Keep the default Profile of DOTS
  - Click on the OK Button
- If everything worked OK then you'll see a Success dialog
  - Click on the OK button



Deploying & debugging a DOTS tasklet

- Finally, time to run the tasklets and break into the debugger
- Enable debugging via notes.ini
  - DOTS_DEBUGADDRESS=8001
  - DOTS_DEBUGSUSPEND=y
- load dots on the server console
- Set a break point in your code
- Go back to Run->Debug Configurations...
  - Select DOTS Remote Debugging under Remote Java Applications
  - Click on the Debug button

Deploying & debugging a DOTS tasklet

- Confirm the switch to the Debug Perspective by clicking the Yes button

Deploying & debugging a DOTS tasklet

- Debug as required

Deploying & debugging a DOTS tasklet

- Output examples

Summary of what we accomplished in this section

- We learned how to create an OSGi plugin that contributes tasklets to the DOTS process
- We showed the different types of tasklets: scheduled and on-demand
- We deployed and debugged the tasklets using the PDE Tool
- You can find more comprehensive documentation about DOTS by downloading it from the OpenNTF project

Agenda

- Opening comments
- Introduction and technology overview
- Prerequisites
- Environment setup for HTTP
- Creating, deploying & debugging a simple servlet
- Creating & deploying a J2EE web app
- Creating & deploying an XPages Extension Library
- Environment setup for DOTS
- Creating, deploying & debugging DOTS tasklets
- Q & A

Questions or comments?

Legal disclaimer

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Quickr, Sametime, WebSphere, UC2, PartnerWorld and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.