

Using SailFin and Eclipse to build SIP-based and RESTful applications

Presented by: Fatna Belqasmi,
PhD, Researcher associate at Concordia University



Agenda

- Download and installation
- Configuration
- Create your first SIP Servlet
- Create your first RESTful service
- Make the SIP Servlet and the RESTful service communicate

Download & Installation

- Download Eclipse Helios (or other version)
 - Go to: <http://www.eclipse.org/downloads/packages/release/helios/sr2>
 - Download: [Eclipse IDE for Java EE Developers](#)
- Download SailFin
 - Go to: http://sailfin.java.net/downloads/download_windows.html
 - Download: [sailfin-installer-v2-b31g-windows.jar](#)
- Download Jersey
 - Go to: http://jersey.java.net/nonav/documentation/latest/chapter_deps.html
 - Download: A [zip of Jersey](#) containing the Jersey jars, core dependencies (it does not provide dependencies for third party jars beyond those for JSON support) and JavaDoc.

Download & Installation

- Install SailFin

- The installation procedure is described in the following website and is presented below: <http://sailfin.java.net/downloads/instructions.html>

1. Open a command line

2. Go to the directory where you downloaded the SailFin jar file:

1. `cd C:\sailfin`

3. Run the following command

1. `java -Xmx256m -jar filename.jar` (replace 'filename' with the name of the jar file you downloaded)

4. A new directory with name 'sailfin' will be created. Change directory to "sailfin«

1. `cd sailfin`

5. Run the following command

1. `lib\ant\bin\ant -f setup.xml` (see the web site for installation on linux)

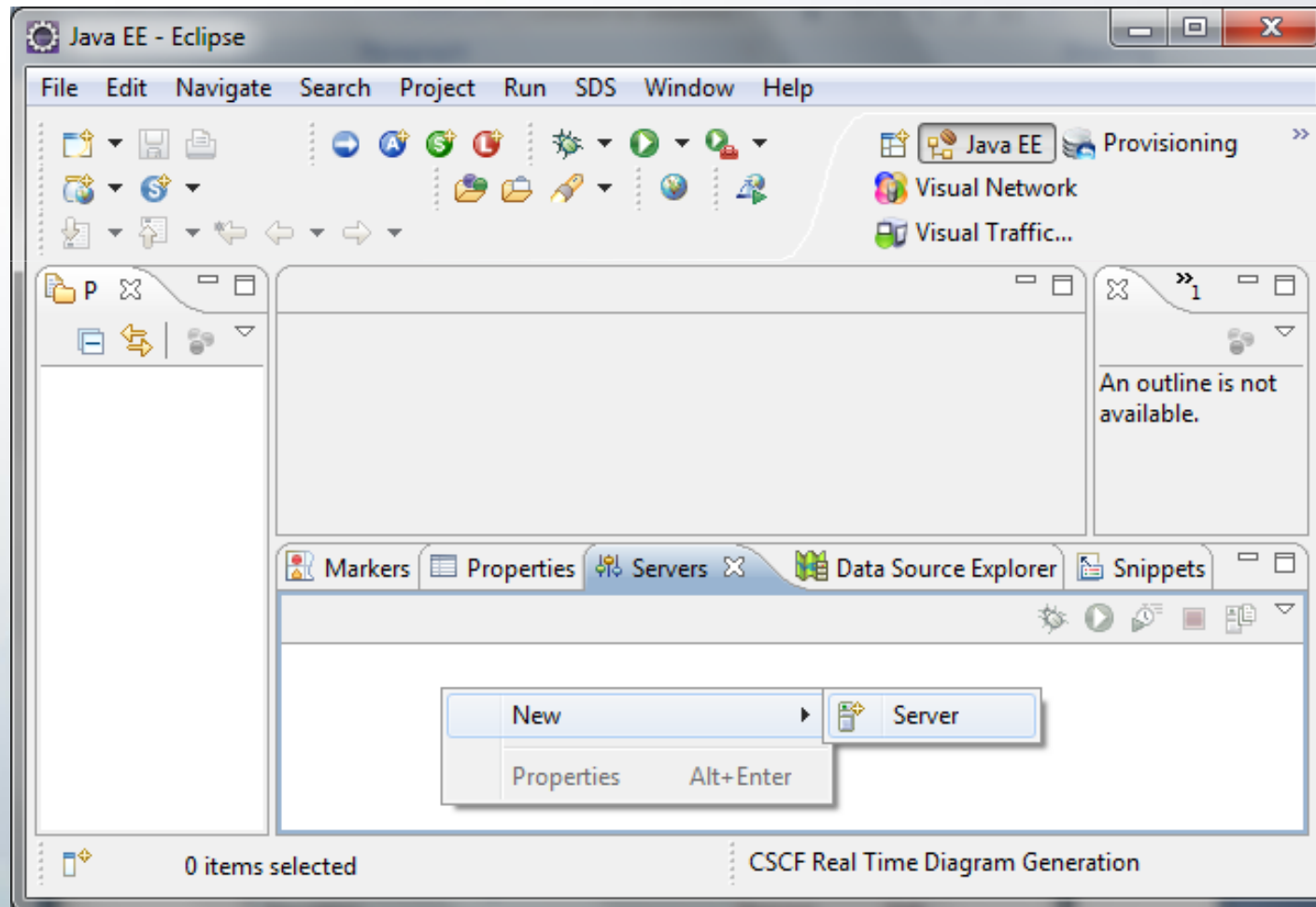
2. If you get an error, check that your JAVA_HOME is set correctly

3. You can set JAVA_HOME using the following command:

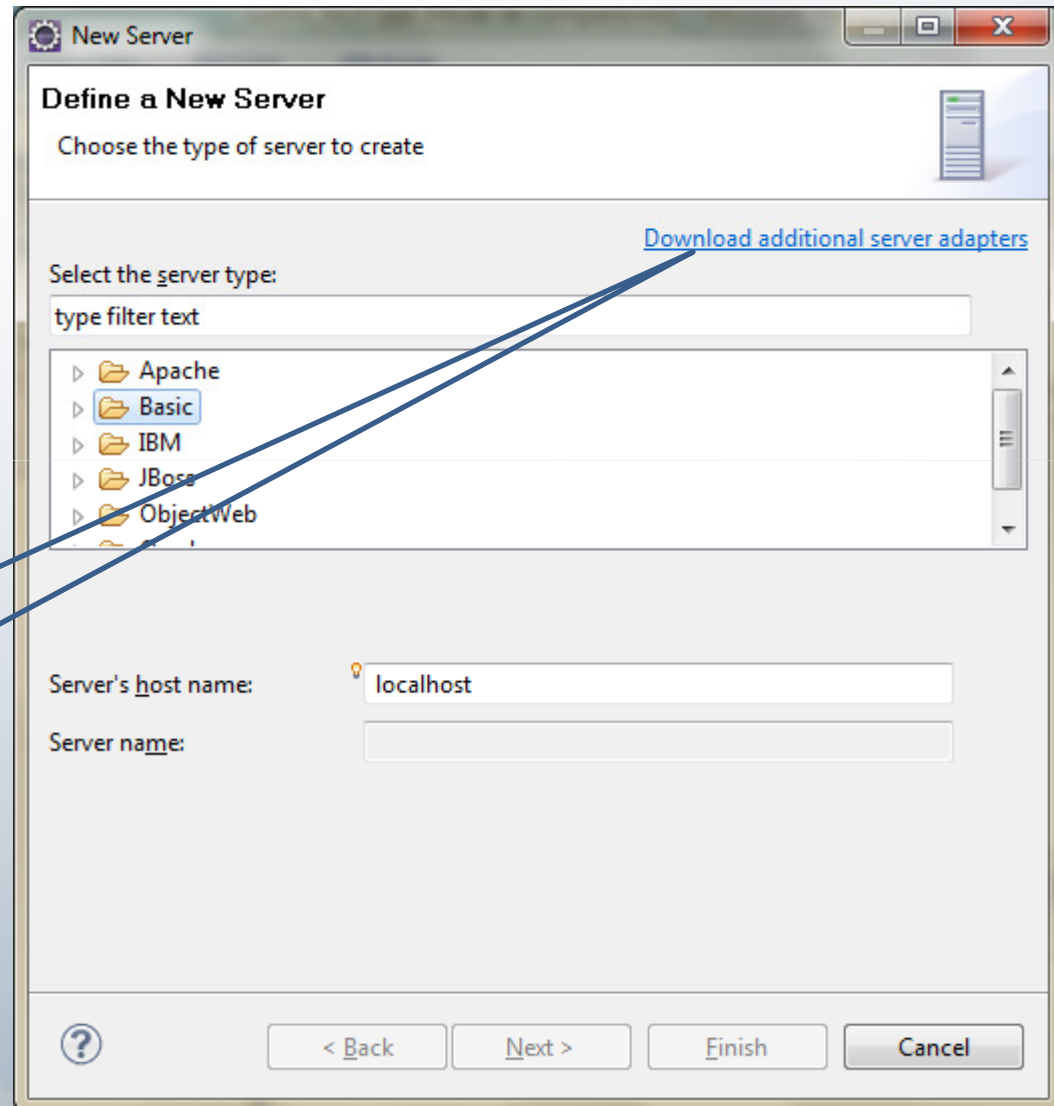
1. `set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_15\`

Configuration

1. Run Eclipse
2. Add a new server

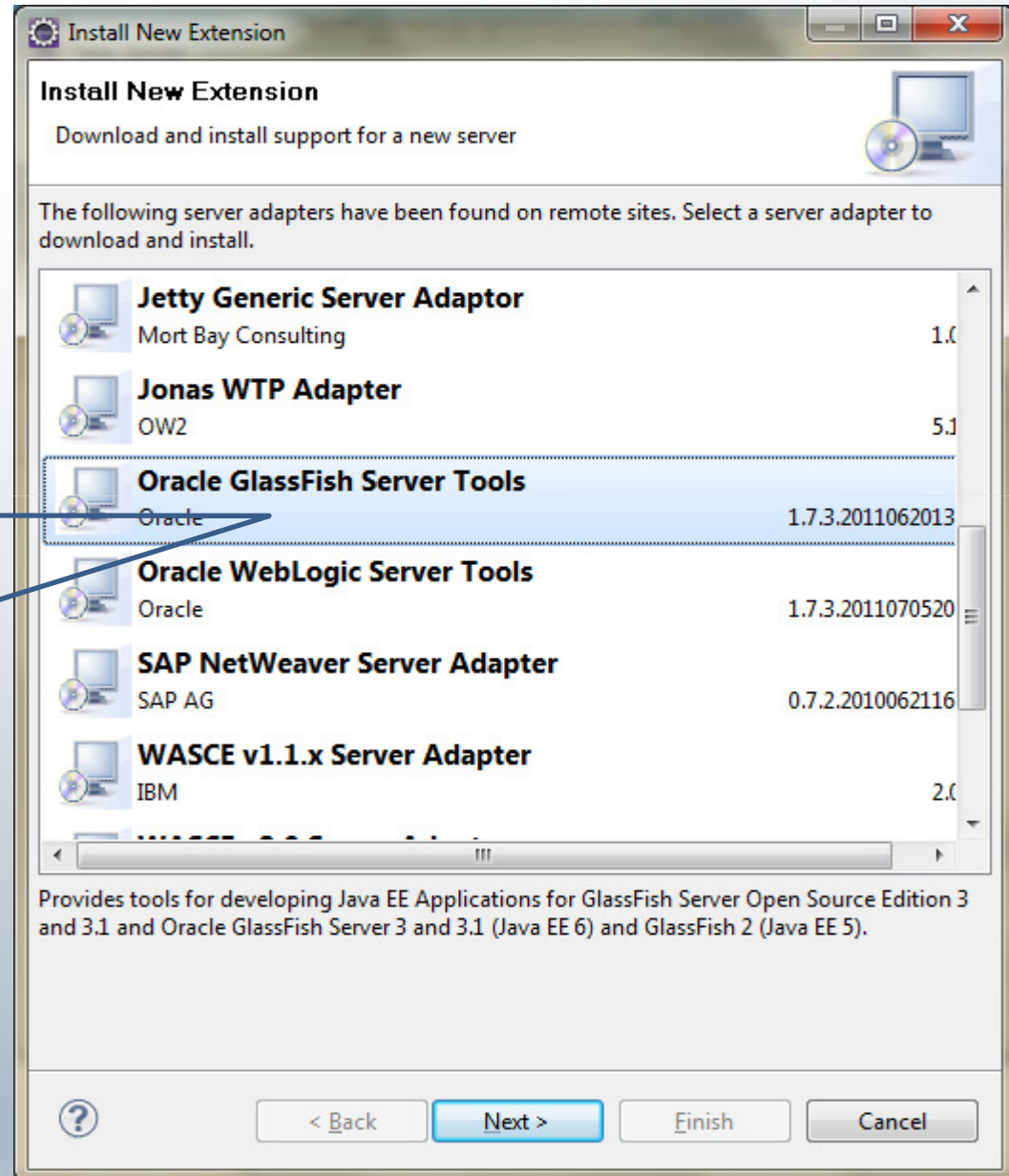


Configuration



Click 'download additional server adapters'

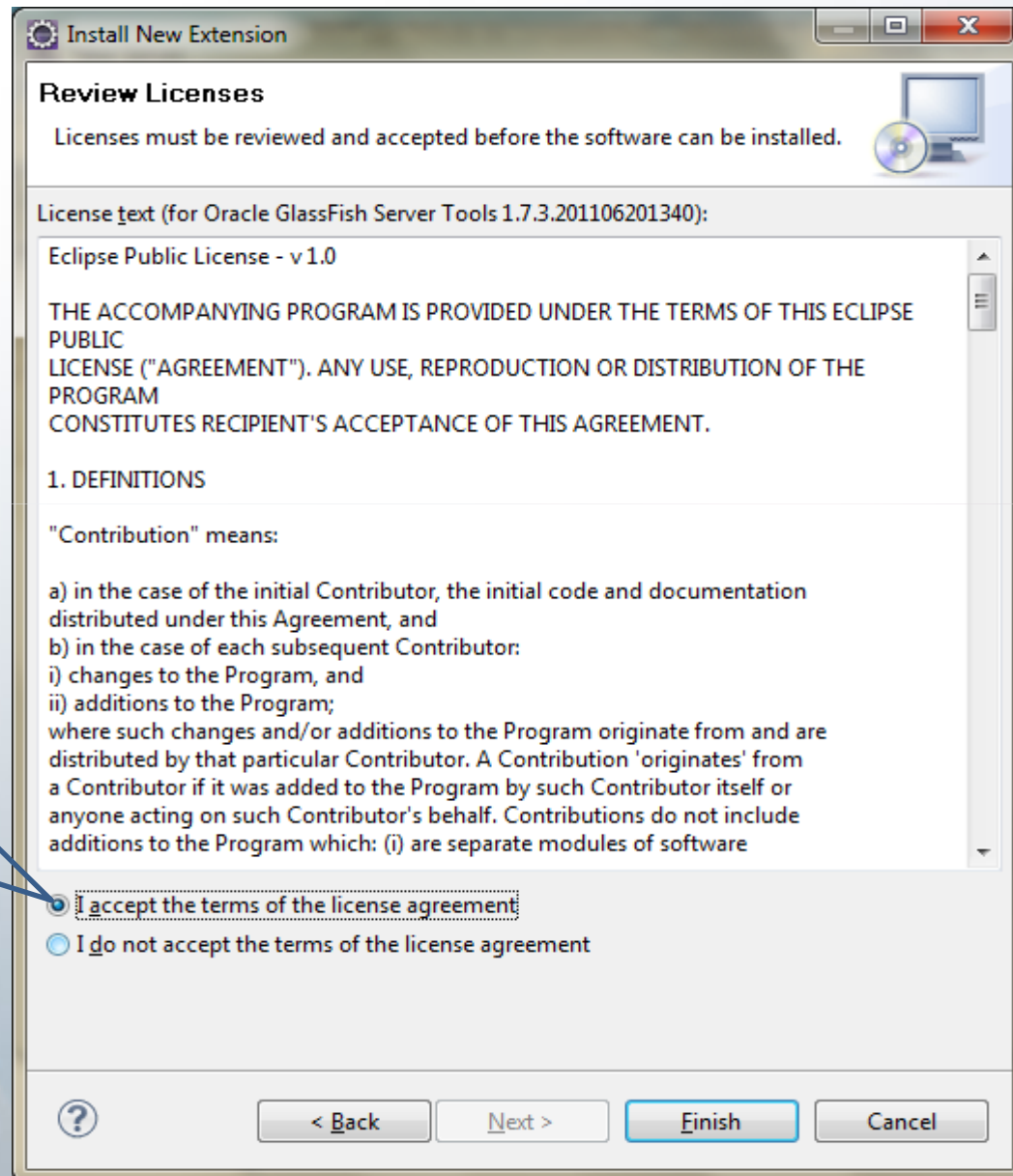
Configuration



Select 'Oracle GlassFish Server Tools' and click 'Next'

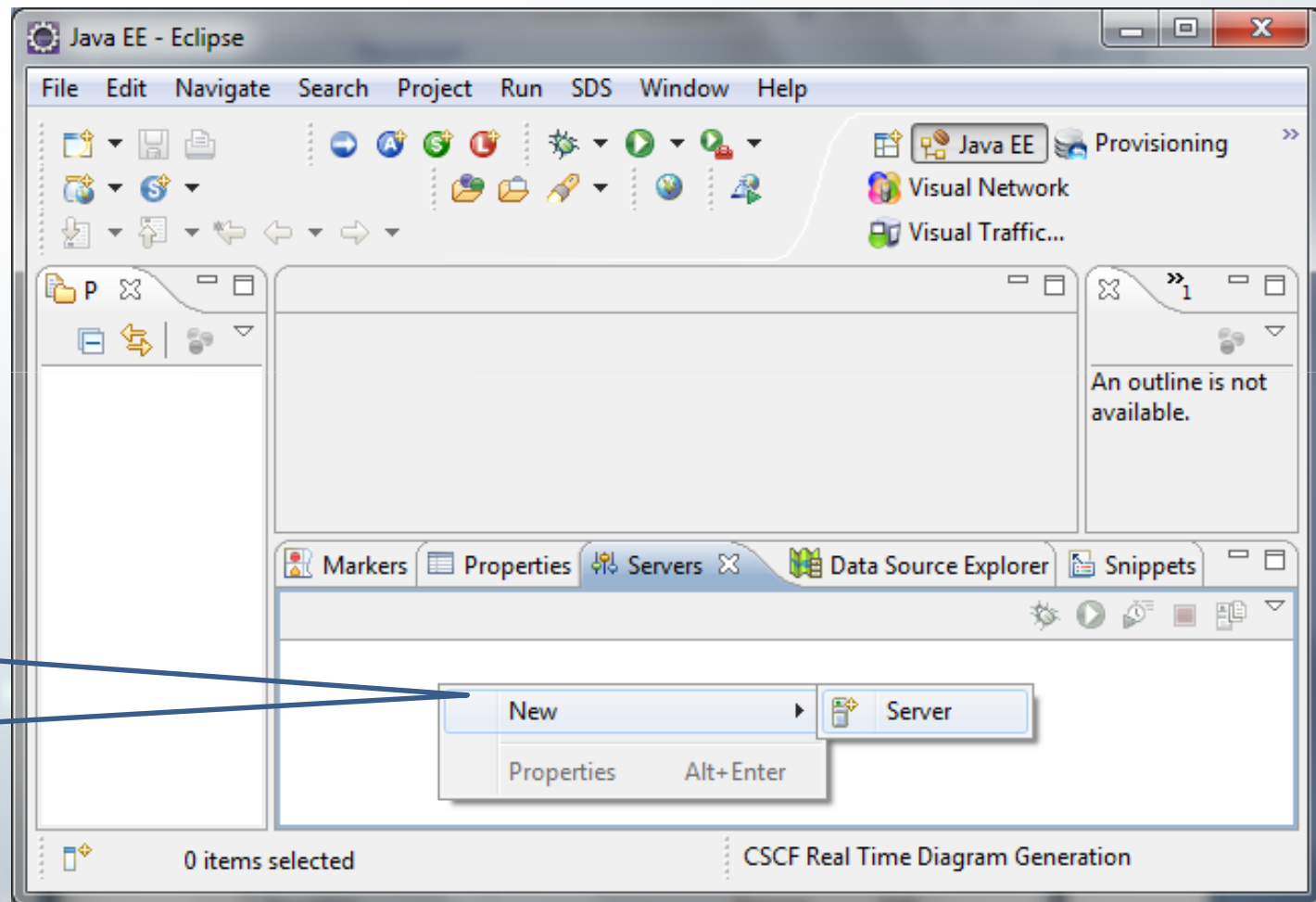
- Under other versions of Eclipse, this may be 'GlassFish Server' or something similar

Configuration



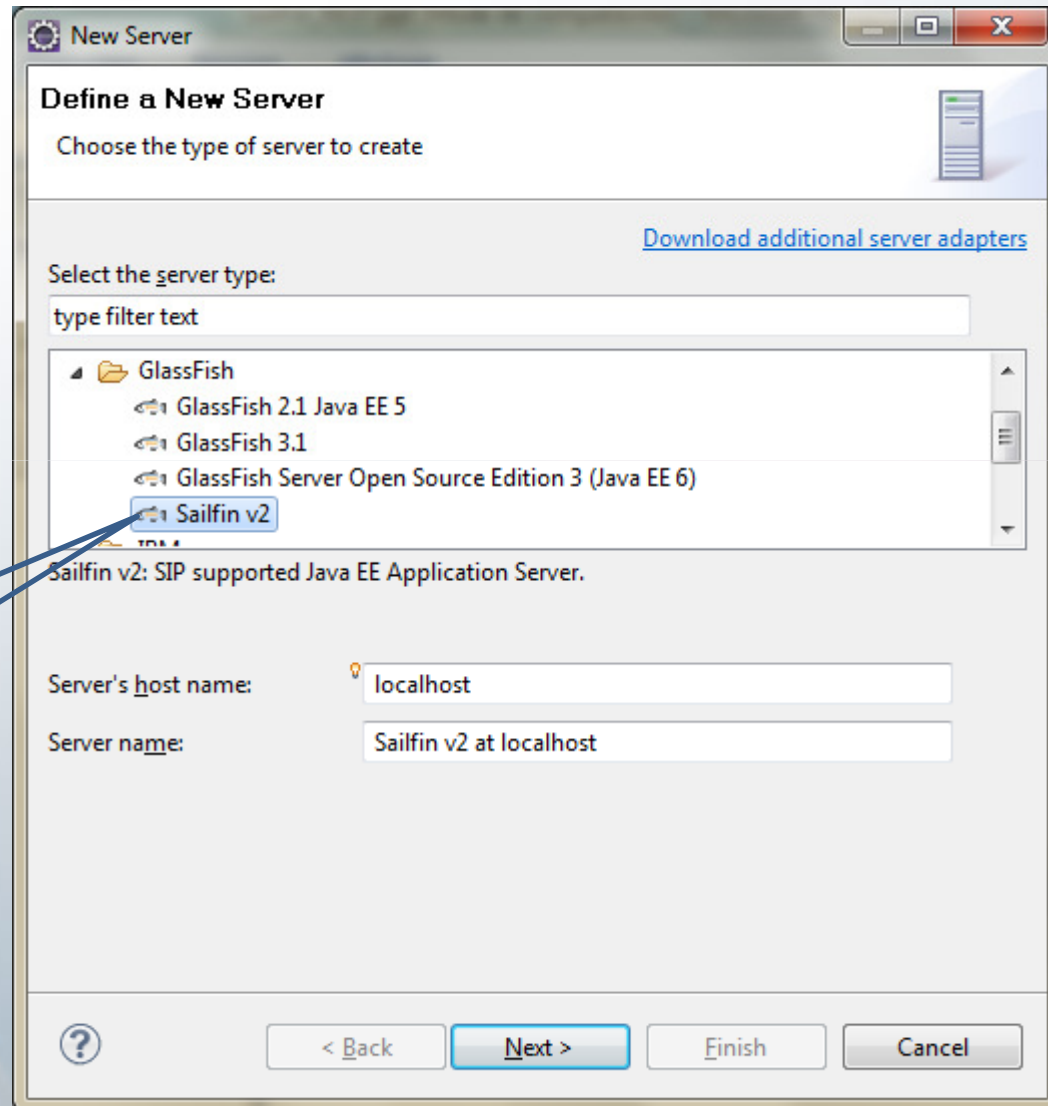
Accept the license agreement and click 'Finish'

Configuration



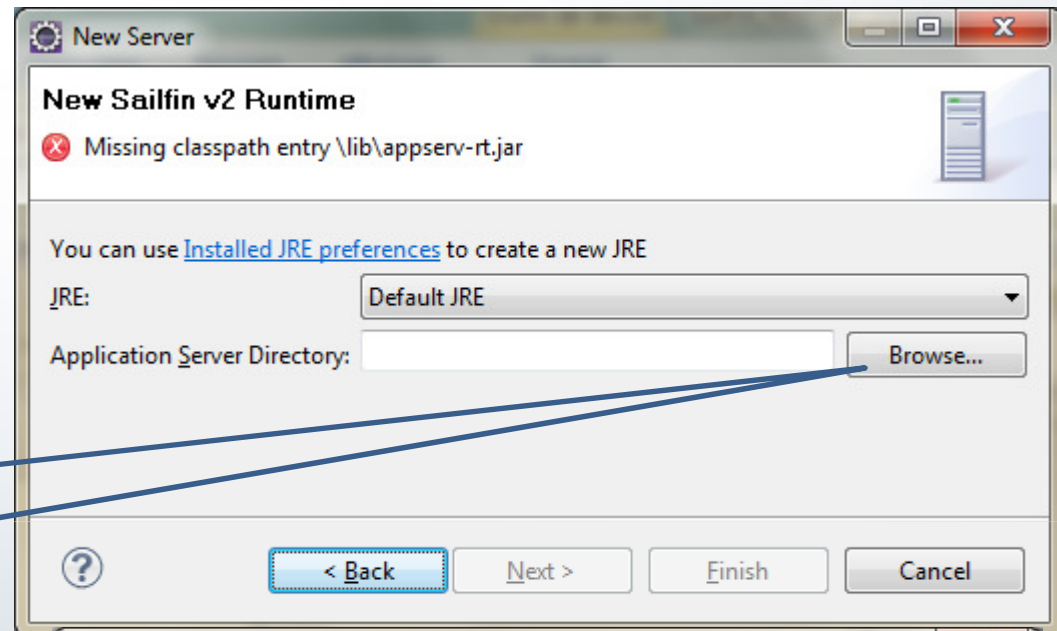
Choose add
new
server
again

Configuration



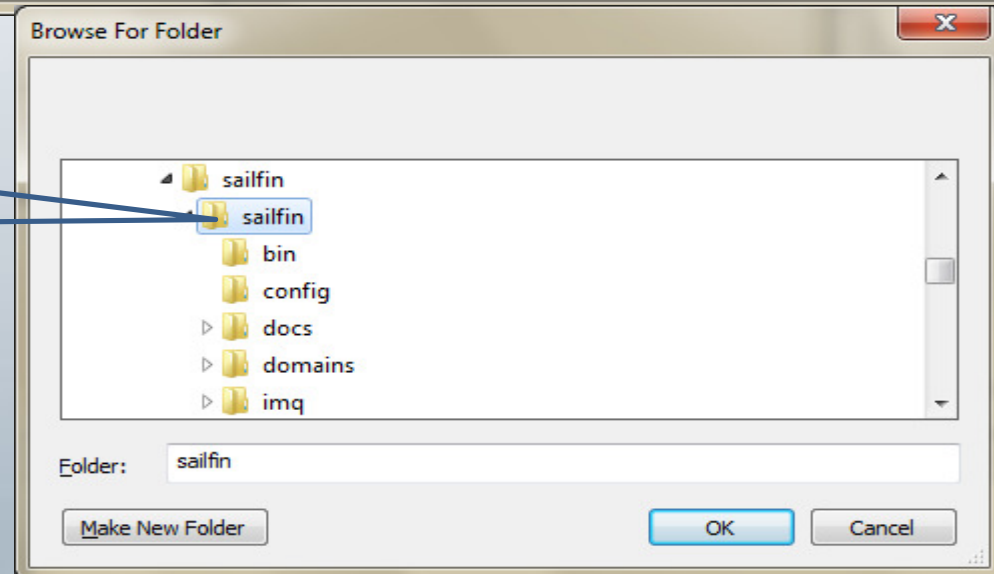
Choose 'sailfin V2'
and click 'Next'

Configuration



Choose 'sailfin V2'
and click 'Next'

Choose the
directory
where you
installed sailfin
and click 'OK'



Configuration

New Server

New Sailfin v2 Runtime
Define a new Sailfin v2 runtime

You can use [Installed JRE preferences](#) to create a new JRE

JRE:

Application Server Directory:

Click 'Next'

New Server

New Sailfin v2 Server
Create a new Sailfin v2 server

Domain Directory:

Administrator Id:

Administrator Password:

Configure your server and click 'Finish'

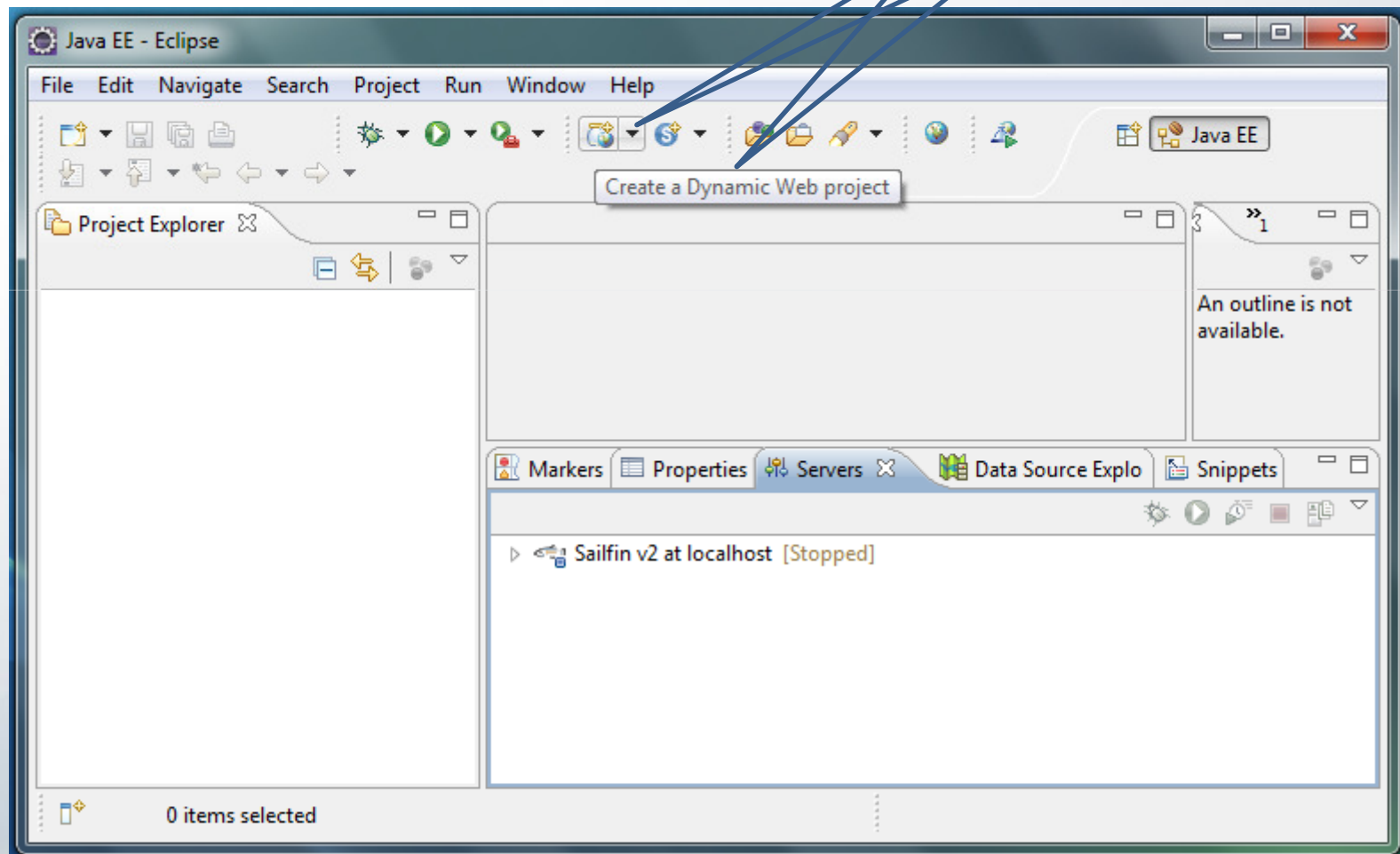
Create your first SIP Servlet

Create your first SIP Servlet

- Example: a simple “Hello World” SIP Servlet
 - Create a SIP Servlet
 - Create a Dynamic Web project
 - Create a package
 - Create a SIP Servlet
 - Add the Servlet methods (e.g. doInvite())
 - Start SailFin AS
 - Deploy and test the Servlet
 - Run the SIP Servlet on the sailfin server
 - Run a SIP client (e.g. X-Lite)

Create your first SIP Servlet

- Create a Dynamic Web project



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

Use default location

Location:

Target runtime

Dynamic web module version

Configuration

A good starting point for working with Sailfin v2 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name:

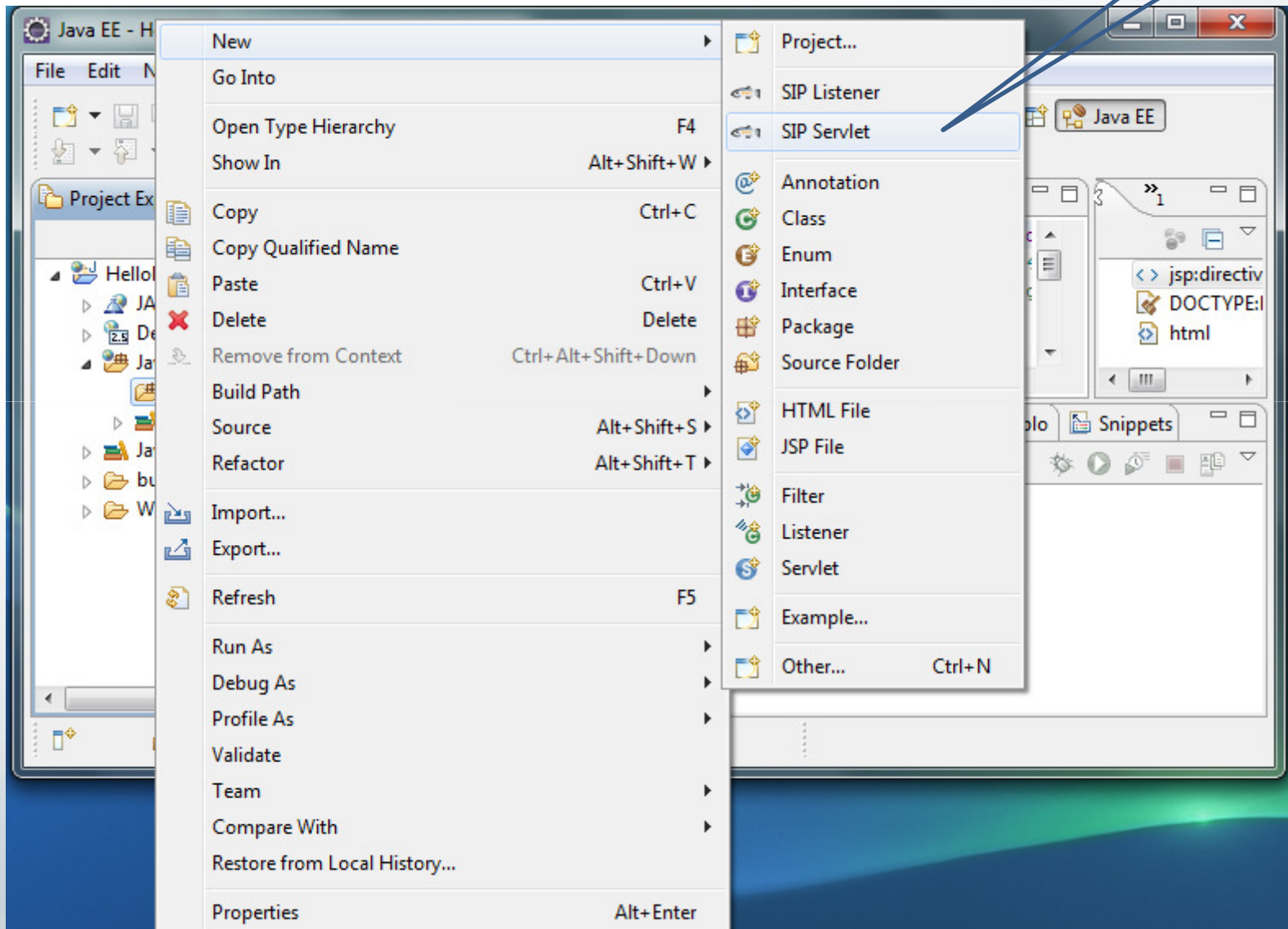
Working sets

Add project to working sets

Working sets:

Name your project
and click 'Finish'

Create your first SIP Servlet



Create your first SIP Servlet

Specify the Servlet name and package

SIP Servlet
Create a new SIP Servlet. Note that target project must contain the SIP Servlet Descriptors Files Facet.

Sailfin project: HelloProject

Source folder: \HelloProject\src

Java package: test.com

Class name: HelloSipServlet

Superclass: javax.servlet.sip.SipServlet

Create your first SIP Servlet

The screenshot displays the Eclipse IDE interface for a Java EE project named 'HelloProject'. The main editor window shows the source code for 'HelloSipServlet.java'. The code defines a package 'test.com', imports 'java.io.IOException', and implements the 'SipServlet' interface. It includes a 'serialVersionUID' and a 'ctx' field for the ServletContext. The code is as follows:

```
package test.com;

import java.io.IOException;

/**
 * SipServlet implementation class HelloSipServlet
 */
@javax.servlet.sip.annotation.SipServlet
public class HelloSipServlet extends SipServlet {

    private static final long serialVersionUID = 3978;

    //Reference to context - The ctx Map is used as a
    ServletContext ctx = null;

    /**
     * Demonstrates extension with a new "REPUBLISH"
```

The Project Explorer on the left shows the project structure: 'HelloProject' contains 'JAX-WS Web Services', 'Deployment Descriptor: HelloProje', 'Java Resources', 'src' (containing 'test.com'), and 'Libraries'. The 'test.com' package contains 'HelloSipServlet.java' and 'HelloSipServlet'. The bottom status bar indicates 'Sailfin v2 at localhost [Stopped]' and 'Writable Smart Insert 1:1'.

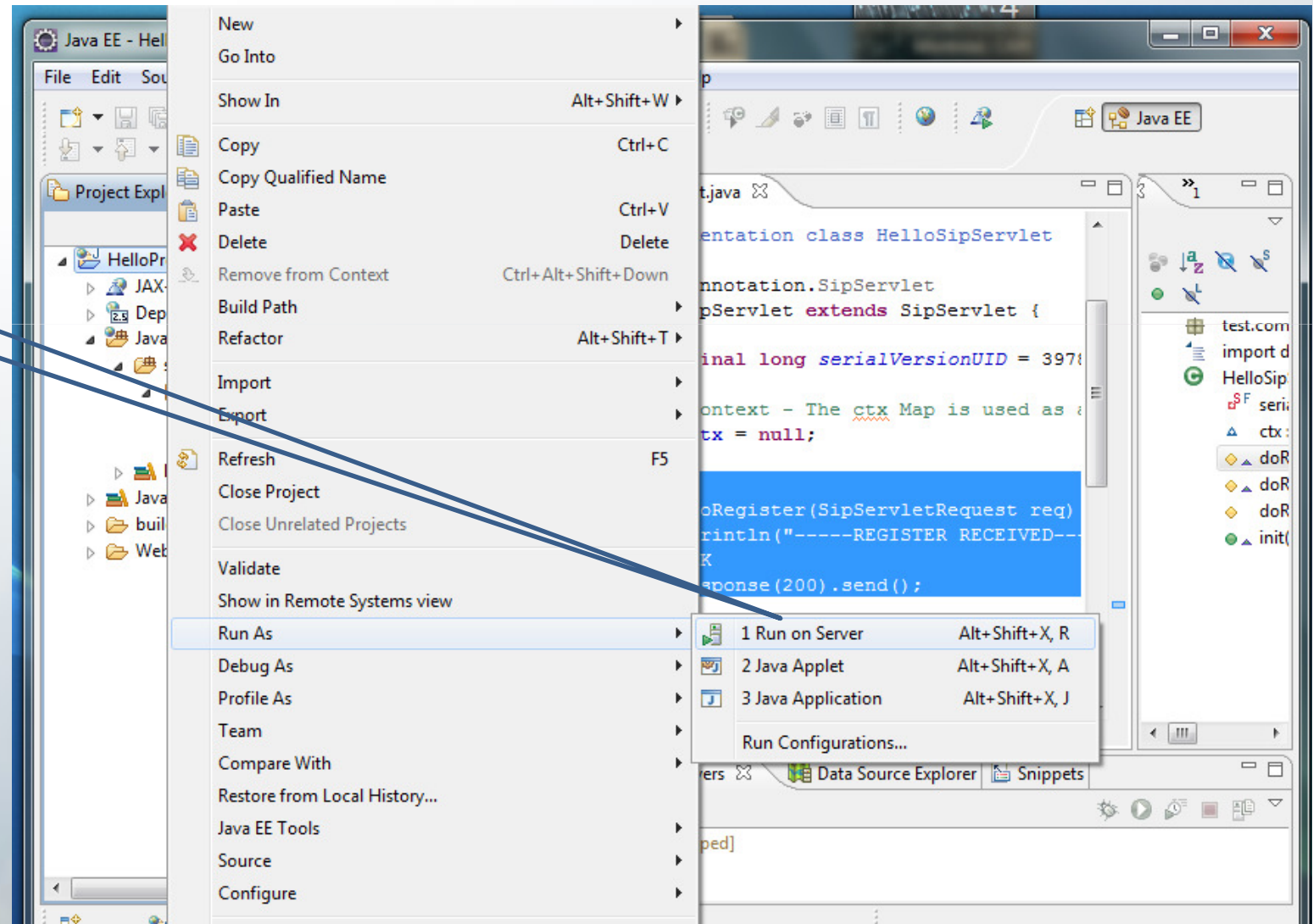
Create your first SIP Servlet

- Add the following to the code of the SIP Servlet to handle REGISTER requests

```
@Override
protected void doRegister(SipServletRequest req) throws
                        ServletException, IOException
{
    System.out.println("-----REGISTER RECEIVED-----");
    //send 200 OK
    req.createResponse(200).send();
}
```

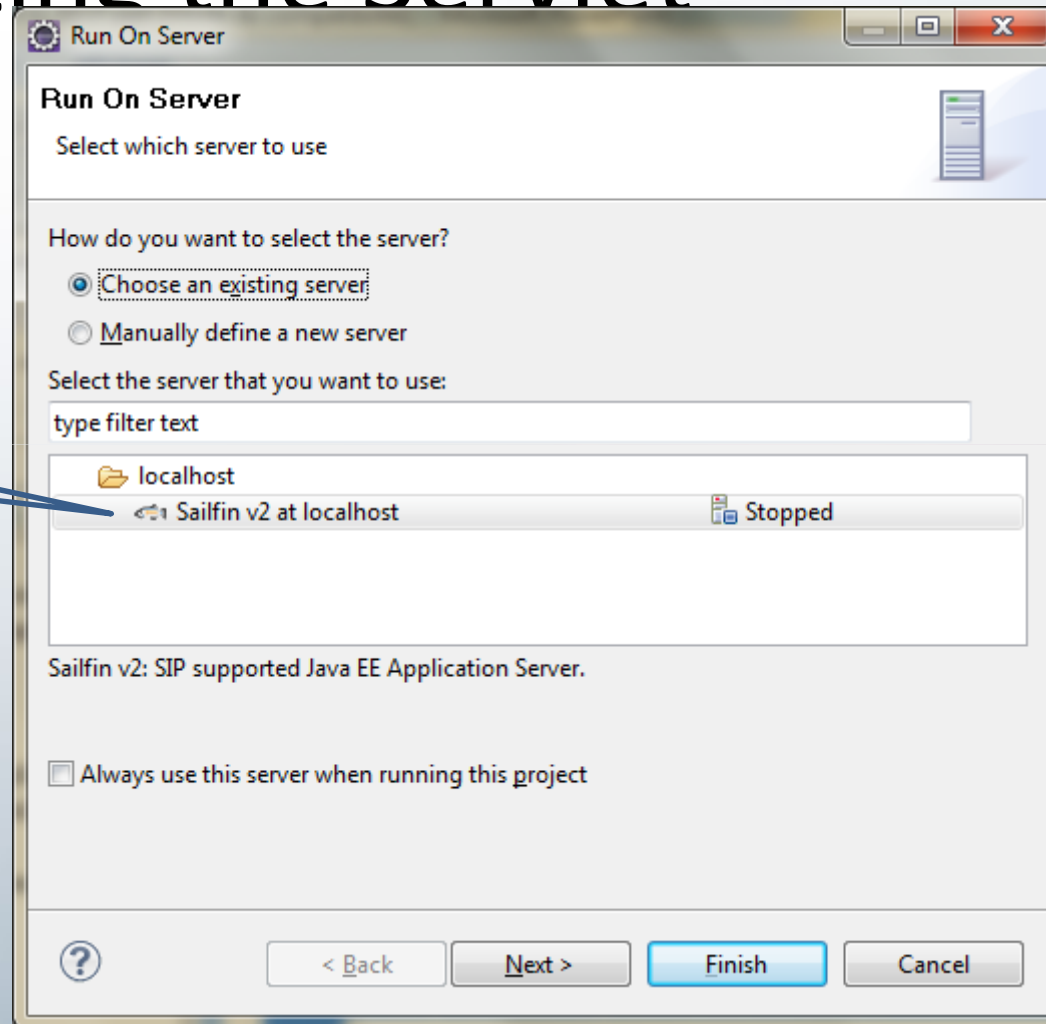
Create your first SIP Servlet: Testing the Servlet

Run your Servlet



Create your first SIP Servlet: Testing the Servlet

Choose 'SailFin
V2' and click
'Finish'



Create your first SIP Servlet:

The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The Project Explorer on the left shows a project named 'HelloProject' with sub-projects for JAX-WS Web Services, Deployment Descriptor, Java Resources, Libraries, JavaScript Resources, build, and WebContent. The Java Resources folder is expanded to show 'src' and 'test.com', with 'HelloSipServlet.java' selected. The main editor area displays 'index.jsp' and 'HelloSipServlet.java', with a preview of a 'GlassFish JSP Page' showing 'Hello World!'. The Console window at the bottom shows a list of consoles, with the selected console displaying the following output:

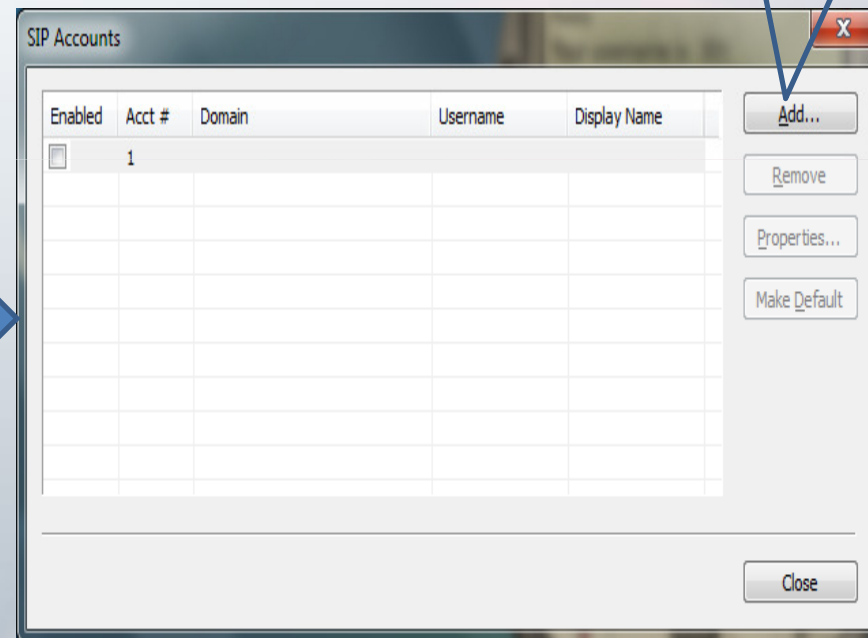
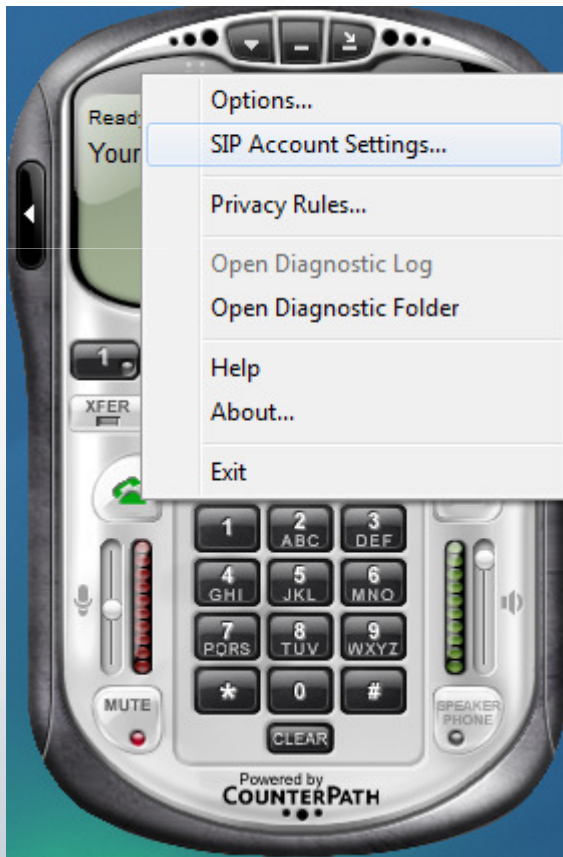
```
<terminated> C:\Program Files\Java\jre6\bin\javaw.exe (2012-03-08 10:48:46 PM)
1 Sailfin v2 at localhost [GlassFish Application Server]
2 C:\myPrograms\eclipse\sailfin\sailfin\domains\domain1\logs\server.log
3 <terminated> C:\Program Files\Java\jre6\bin\javaw.exe (2012-03-08 10:48:46 PM)
tools:
addressources:
replace.module.name.space.
[delete] Deleting: C:\Users\fatna\workspace\.metadata\.plugins\org.e
tools:
deploydir:
[exec] Command deploydir executed successfully.
[echo] Application dir Deployed at: http://localhost:8080/HelloPro:
BUILD SUCCESSFUL
Total time: 3 seconds
```

Two callout boxes provide instructions:

- 1- Click to get the list of consoles
- 2- Choose the SailFin log

Create your first SIP Servlet

- Run X-Lite to test your Servlet



Choose 'Add'

Create your first SIP Servlet

Configure a new account

The IP of the machine where the SIP Servlet is running

The user is registered

The screenshot shows a dialog box titled "Properties of Account 1" with several tabs: "Account", "Voicemail", "Topology", "Presence", and "Advanced". The "Account" tab is selected. Under "User Details", the fields are: Display Name (Alice), User name (301), Password (masked with dots), Authorization user name (empty), and Domain (mydomain.com). Under "Domain Proxy", the checkbox "Register with domain and receive incoming calls" is checked. The "Send outbound via:" section has "domain" unselected and "proxy" selected, with an "Address" field containing "127.0.0.1". The "Dialing plan" field contains "#1|a|a.T;match=1;prestrip=2;". At the bottom are "OK", "Cancel", and "Apply" buttons.



Create your first SIP Servlet

The screenshot shows the Eclipse IDE interface for a Java EE project named 'HelloProject'. The Project Explorer on the left shows the project structure, including a 'src' folder with a 'test.com' package containing 'HelloSipServlet.java'. The main editor displays 'HelloSipServlet.java' and a preview of the 'index.jsp' page, which shows 'Hello World!' in a large font. The browser window at the bottom shows the URL 'http://localhost:8080/HelloProject/' and the rendered page content. The Console window at the bottom right shows the server logs, including the message 'INFO: -----REGISTER RECEIVED-----'. A callout box with a blue border and a pointer to the console log contains the text 'Your message is here'.

Java EE - http://localhost:8080/HelloProject/ - Eclipse

File Edit Navigate Search Project Run Window Help

Project Explorer

- HelloProject
 - JAX-WS Web Services
 - Deployment Descriptor: HelloProje
 - Java Resources
 - src
 - test.com
 - HelloSipServlet.java
 - HelloSipServlet
 - Libraries
 - JavaScript Resources
 - build
 - WebContent

index.jsp HelloSipServlet.java GlassFish JSP Page

http://localhost:8080/HelloProject/

Hello World!

Markers Properties Servers Data Source Explorer Snippets Console

C:\myPrograms\eclipse\sailfin\sailfin\domains\domain1\logs\server.log

```
INFO: WEB0712: Starting Sun GlassFish Communications Server 2.0 HTTP/1.0
INFO: WEB0712: Starting Sun GlassFish Communications Server 2.0 HTTP/1.0
INFO: WEB0712: Starting Sun GlassFish Communications Server 2.0 HTTP/1.0
INFO: SMGT0007: Self Management Rules service is enabled
INFO: SIP module started and available.
INFO: Application server startup complete.

INFO: org.jvnet.glassfish.comms.deployment.annotations.handlers.SipServlet
INFO: deployed with moduleid = HelloProject

INFO: Application /HelloProject configured with SIP persistence type: memory
INFO: Context available with context root/HelloProject

INFO: -----REGISTER RECEIVED-----
```

Your message is here

Create your first RESTful service

Create your first RESTful service

- The following link gives a good step-by-step example for how to create a new RESTful service using Jersey
 - <http://www.vogella.de/articles/REST/article.html>

Create your first RESTful service

- Example: a simple “Hello” SIP Servlet
 - Create a RESTful service
 - Create a Dynamic Web project: [we will reuse the same project we already created for the SIP Servlet](#)
 - Create a new class
 - Add the class logic
 - Define Jersey Servlet dispatcher
 - Deploy and test the service
 - Run your RESTful service on the sailfin server
 - Test the service from a browser

Create your first RESTful service

- Create a RESTful service
 - Copy all jars from the \lib folder of your Jersey download into the folder “WebContent\WEB-INF\lib” of your Dynamic Web project
 - Create a new class ‘Hello’
 - Change the class implementation with the code in the next slide

Create your first RESTful service

```
package test.com;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

//Sets the path to base URL + /hello
@Path("/hello")
public class Hello {

    // This method is called if TEXT_PLAIN is
    request
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayPlainTextHello()
    {
        return "PLAIN TEXT: Hello Jersey";
    }
}
```

```
// This method is called if XML is request
@GET
@Produces(MediaType.TEXT_XML)
public String sayXMLHello()
{
    return "<?xml version='1.0'?>" + "<hello>
        Hello Jersey" + "</hello>";
}

// This method is called if HTML is request
@GET
@Produces(MediaType.TEXT_HTML)
public String sayHtmlHello()
{
    return "<html> " + "<title>" + "Hello Jersey"
        + "</title>"
        + "<body><h1>" + "HTML: Hello Jersey" +
        "</body></h1>" + "</html> ";
}
}
```

Create your first RESTful service

- Create a RESTful service
 - **Define Jersey Servlet dispatcher**
 - Open the file “WebContent\WEB-INF\web.xml” and add the following to the file, just before the closing </web-app>

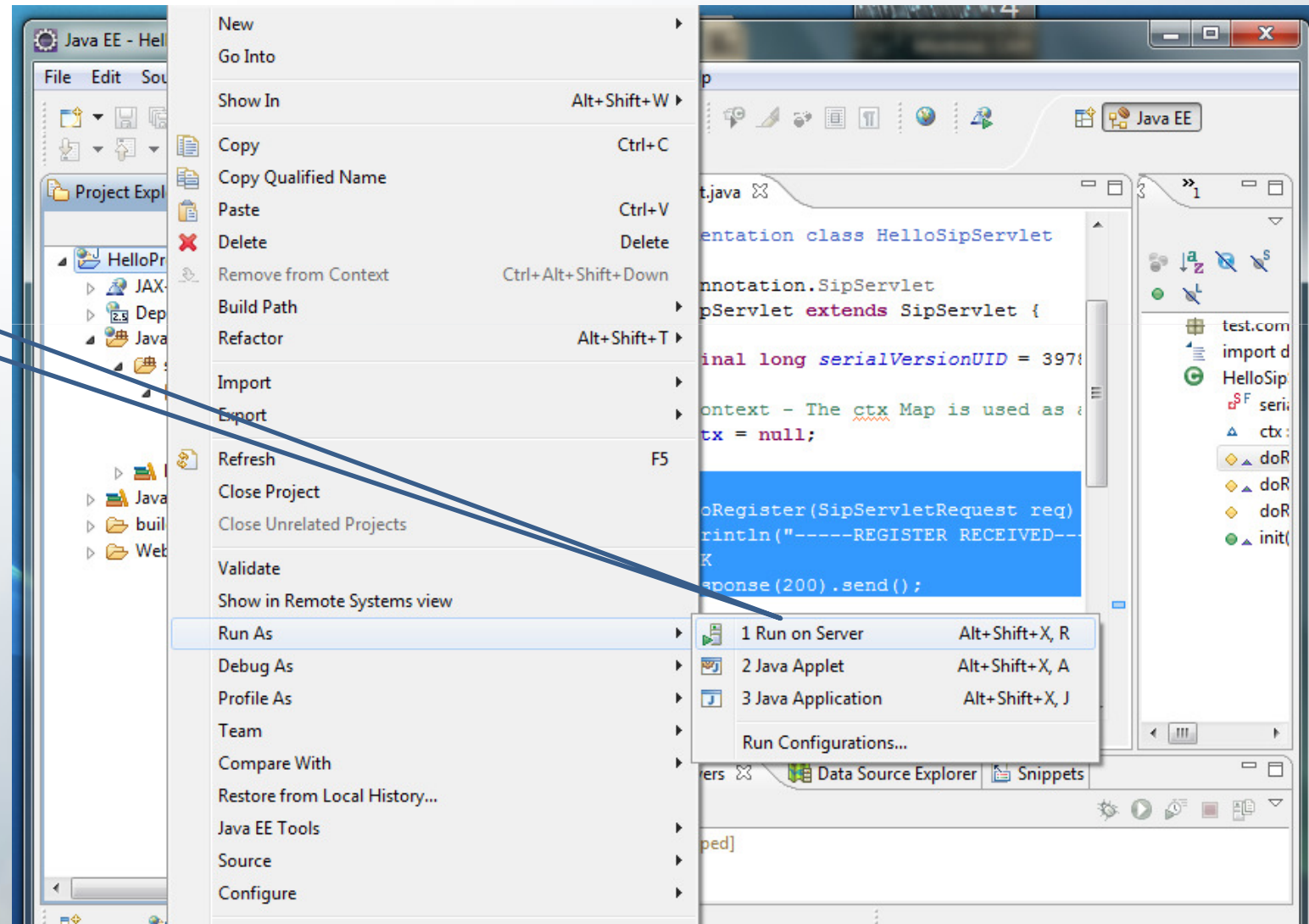
```
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>test.com</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```


Create your first RESTful service

- Deploy and test the service
 - **Run your RESTful service** (see the next 2 slide)
 - Test your REST service (using a browser) under:
<http://localhost:8080/HelloProject/rest/hello>
 - This name is derived from:
 - The "display-name" defined in the "web.xml" file augmented with
 - the servlet-mapping url-pattern (always from the "web.xml" file) and
 - the "hello" @Path annotation from your class file.
 - You should get the message "HTML: Hello Jersey" (The browser requests the HTML representation of your resource)

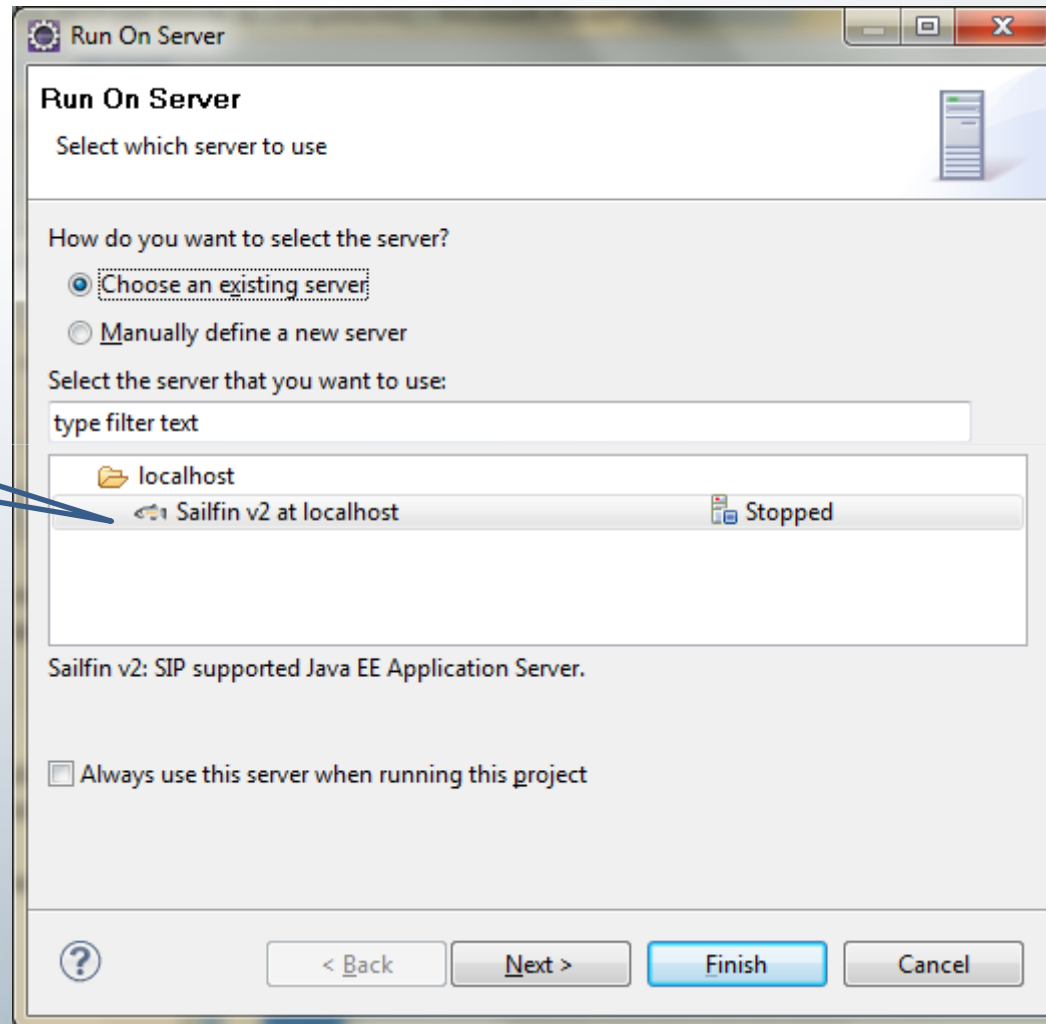
Create your first RESTful service

Run your Servlet



Create your first RESTful service

Choose 'SailFin V2' and click 'Finish'



Create your first RESTful service

- Deploy and test the service
 - The link bellow gives examples on how to
 - Write a client that can get different formats (i.e. XML, plain-text, HTML) of the resource
 - Write a RESTful service accepting different methods (e.g. GET, PUT, POST, DELETE)

<http://www.vogella.de/articles/REST/article.html>

Make the SIP Servlet and the RESTful service communicate

- Add the following to the **web.xml** file

```
<listener>
  <listener-class> test.com.HelloSipServlet </listener-class>
</listener>
```

- Add the following to the REST java class (**Hello.java**)

```
public class Hello {
  @Context ServletContext context;
  .....
  @GET
  @Produces(MediaType.TEXT_HTML)
  public String sayHtmlHello() {
    testVariable = "testVariable from Hello REST resource";
    context.setAttribute("myAttribute", testVariable);
    return "<html> " + "<title>" + "Hello Jersey" + "</title>"
      + "<body><h1>" + "HTML: Hello Jersey" + "</body></h1>" + "</html> ";
  }
}
```

Make the SIP Servlet and the RESTful service communicate

- Add the following to the SIP Servlet ([HelloSipServlet.java](#))

```
public class HelloSipServlet extends SipServlet implements
ServletContextAttributeListener {
    static ServletContext context;

    .....

    public void attributeAdded (ServletContextAttributeEvent attrEv) {
        handleTrigger(attrEv); }
    public void attributeReplaced (ServletContextAttributeEvent attrEv) {
        handleTrigger(attrEv);}
    public void attributeRemoved (ServletContextAttributeEvent scab) {
        // TODO Auto-generated method stub}
    private void handleTrigger(ServletContextAttributeEvent attrEv){
        if (attrEv.getName().equalsIgnoreCase("myAttribute"))
            String value = (String) attrEv.getValue();
            System.out.println("Param value: "+value);
    }
}
```

Media Server

- You may try **MCU Media Server**:
<http://www.medooze.com/>
- It includes
 - A Video Multiconference Server
<http://www.medooze.com/products/mcu.aspx>
 - A Media Mixer Server
<http://www.medooze.com/products/media-mixer-server.aspx>

References

- <http://www.eclipse.org/downloads/packages/release/helios/sr2>
- http://sailfin.java.net/downloads/download_windows.html
- http://jersey.java.net/nonav/documentation/latest/chapter_deps.html
- <http://www.vogella.de/articles/REST/article.html>
- <http://www.medooze.com/>

Q&A