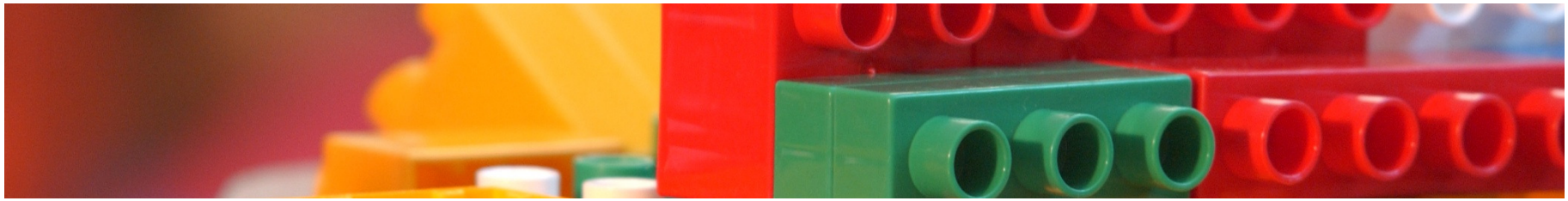
Using SailFin and Eclipse to build SIP-based and RESTful applications

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.

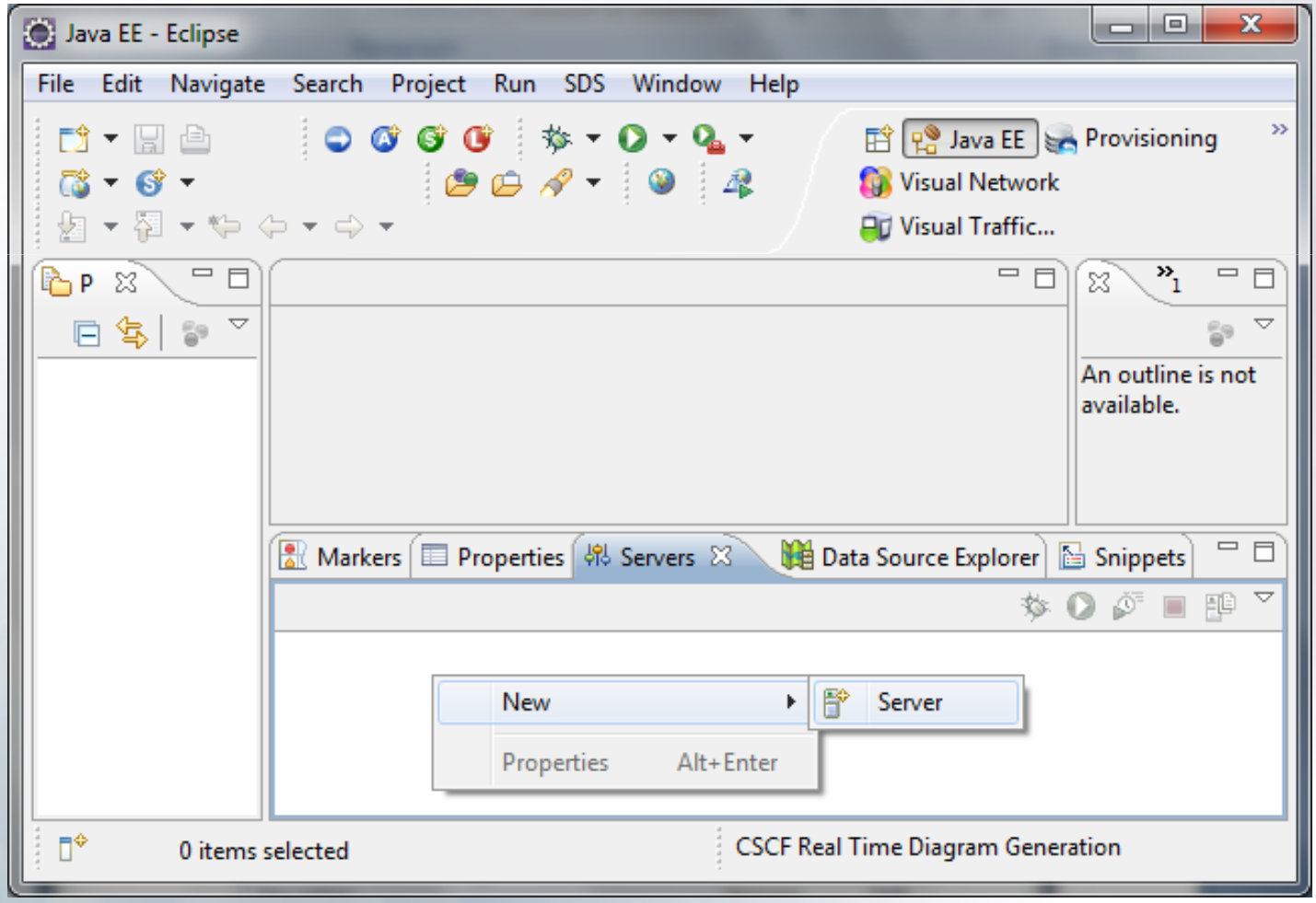
Download & Installation

- Install SailFin
  - The installation procedure is described in the following website and is presented bellow: 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\

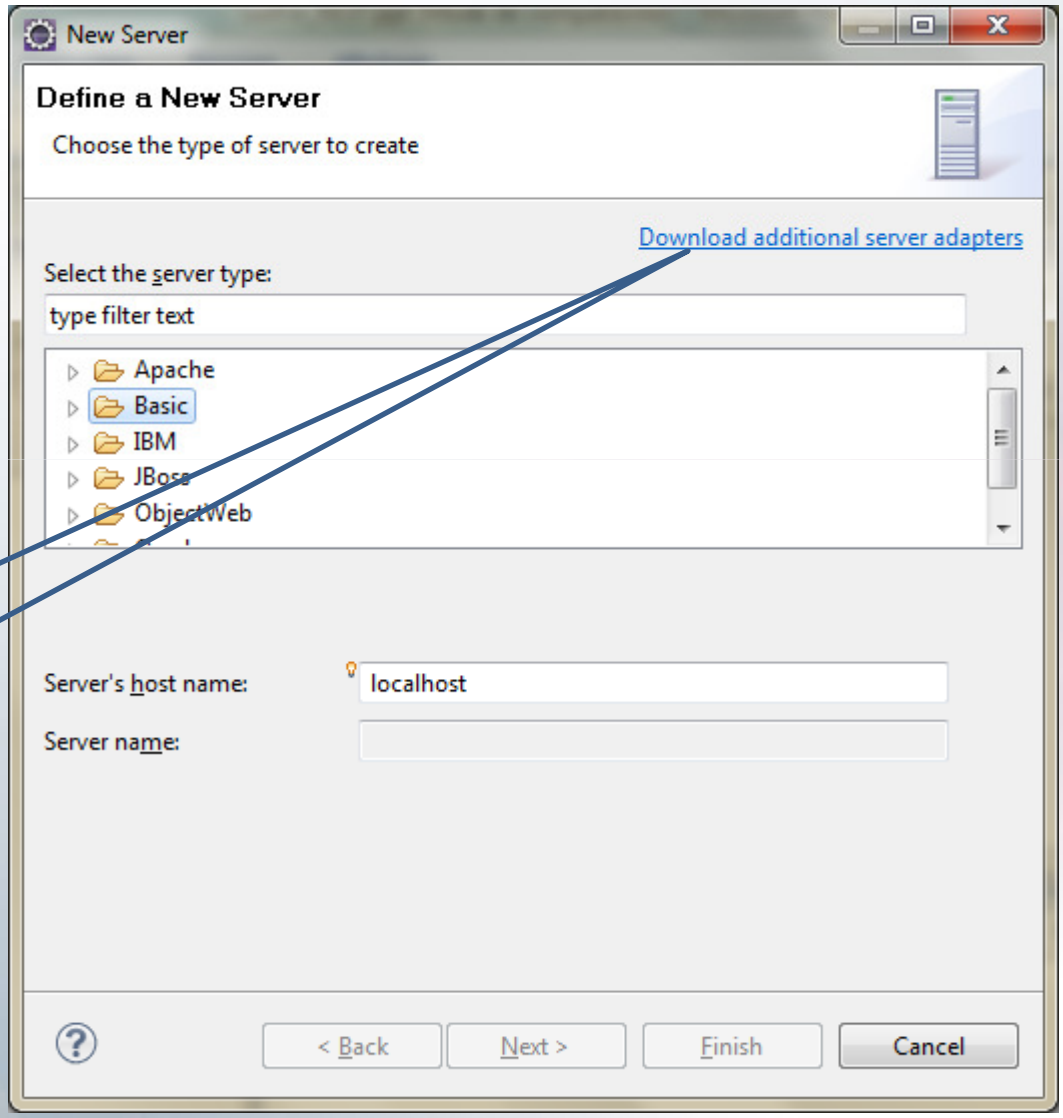
Configuration

1. Run Eclipse
2. Add a new server



Configuration

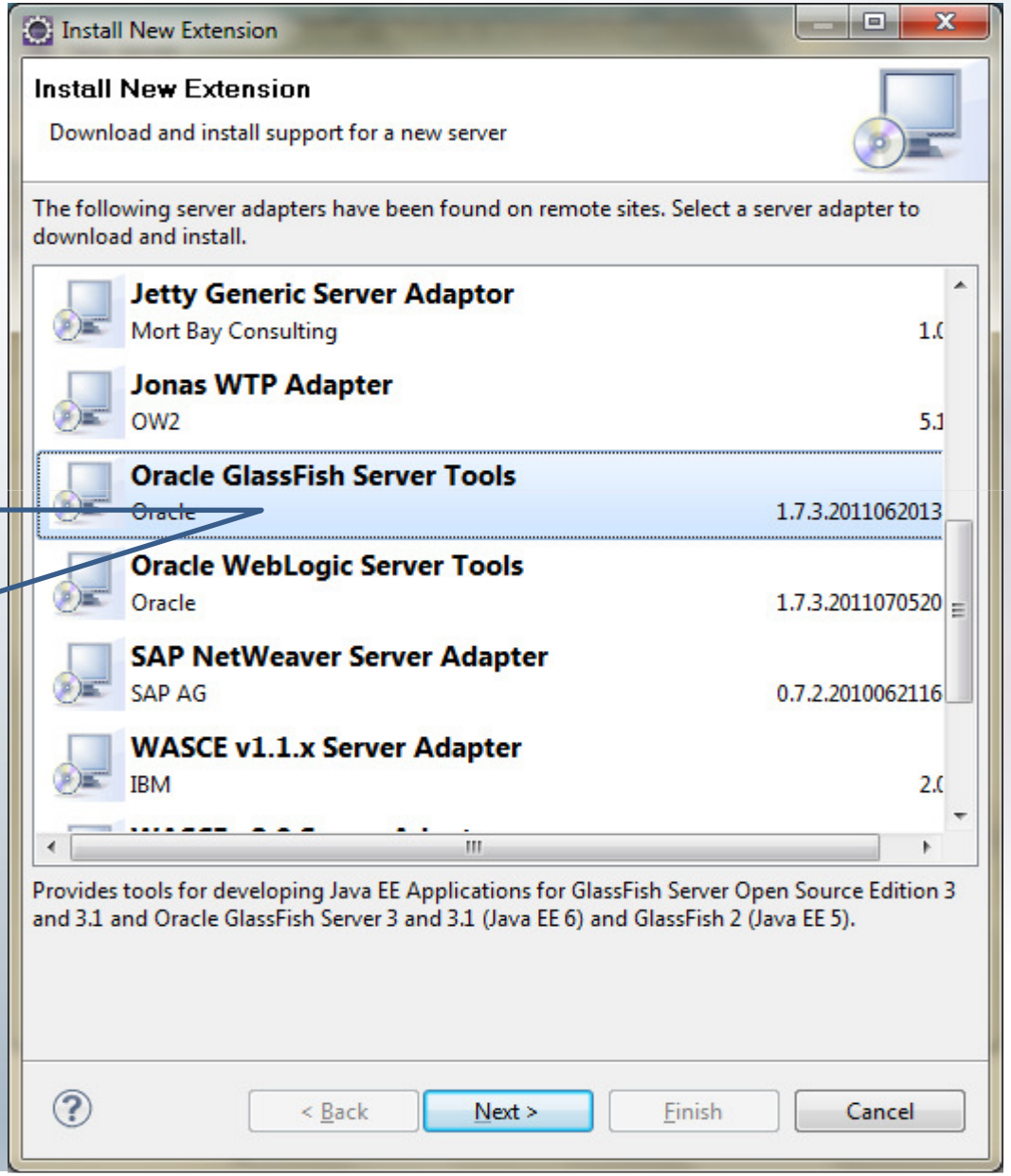
Click ‘download additional server adapters’



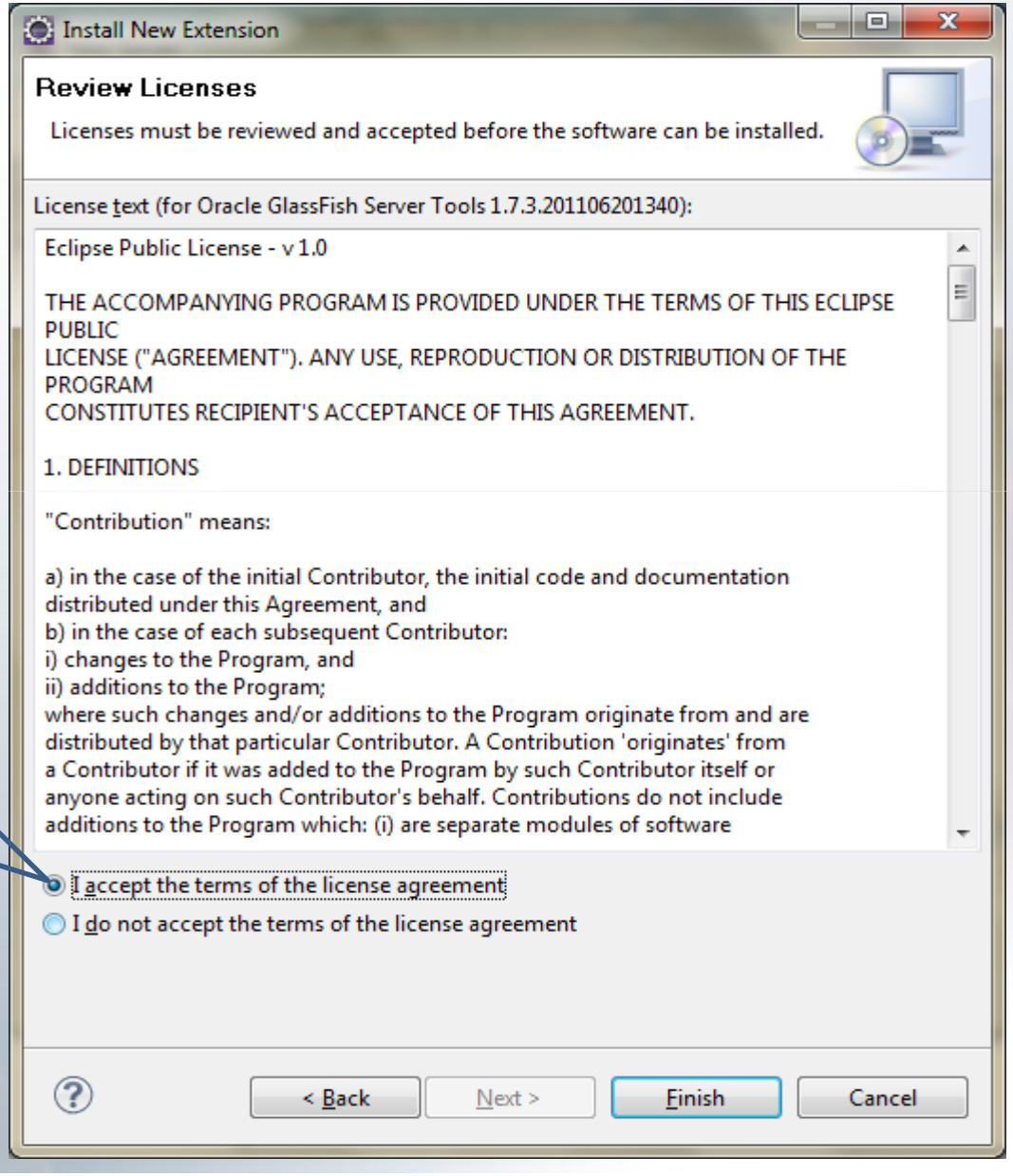
Configuration

Select ‘Oracle G;assFich Server Tools’ and click ‘Next’

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

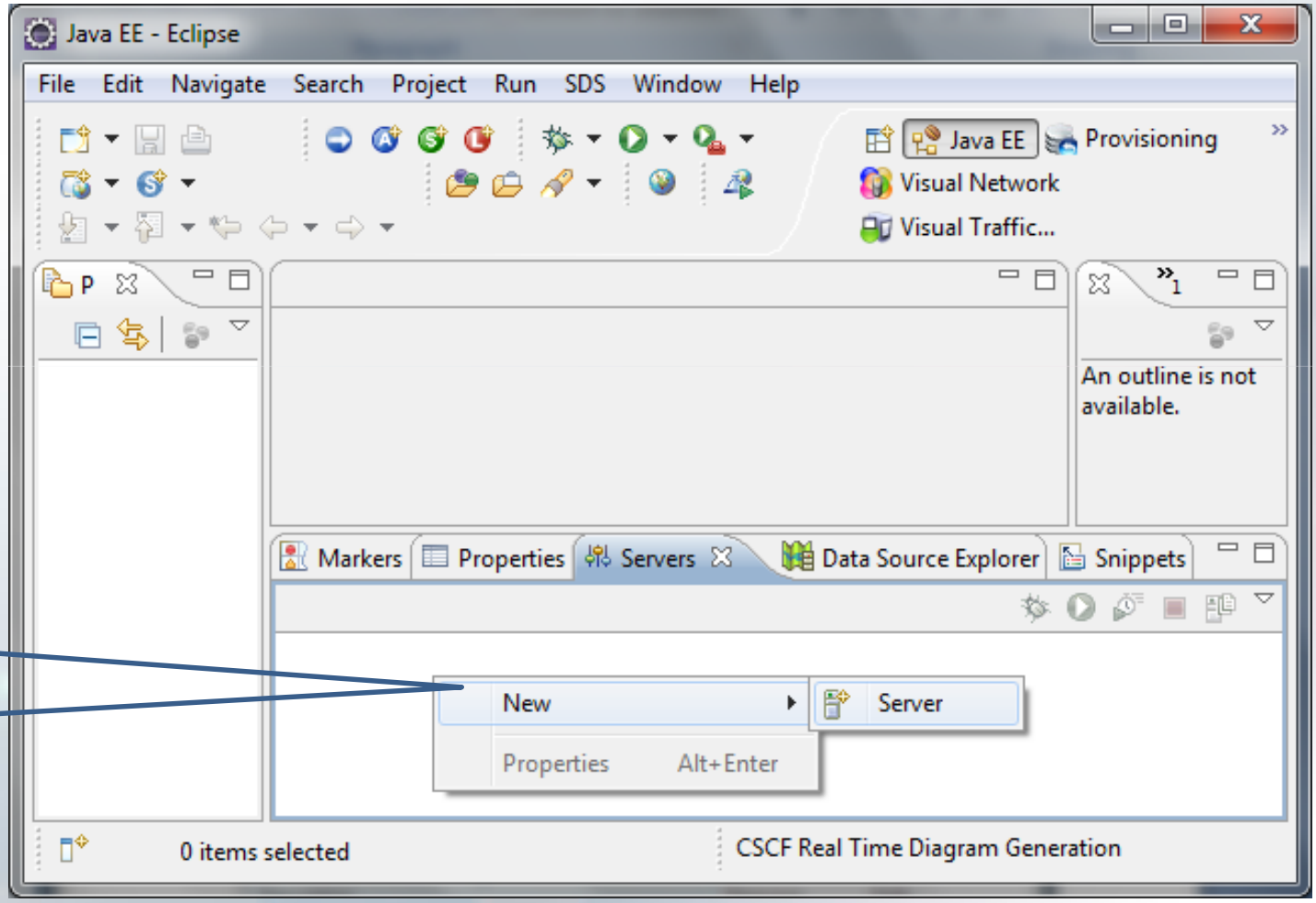


Configuration



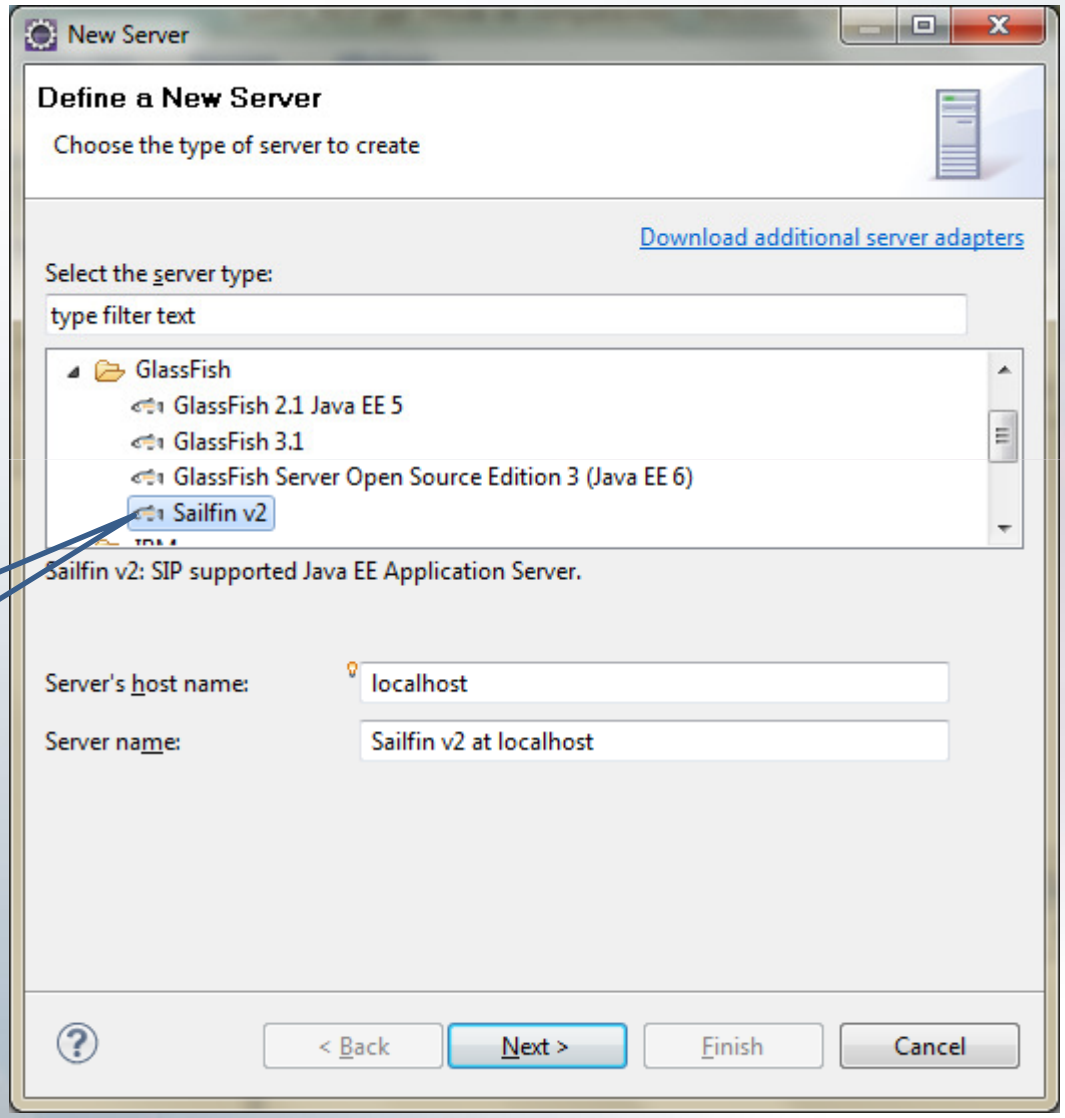
Accept the license agreement and click ‘Finish’

Configuration



Choose add new server again

Configuration

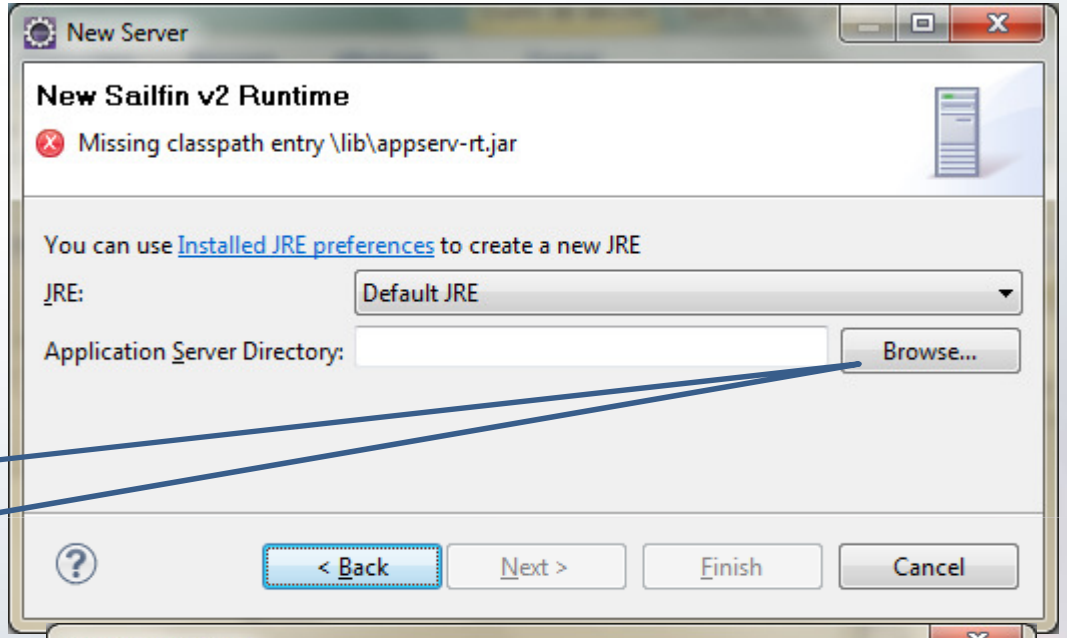


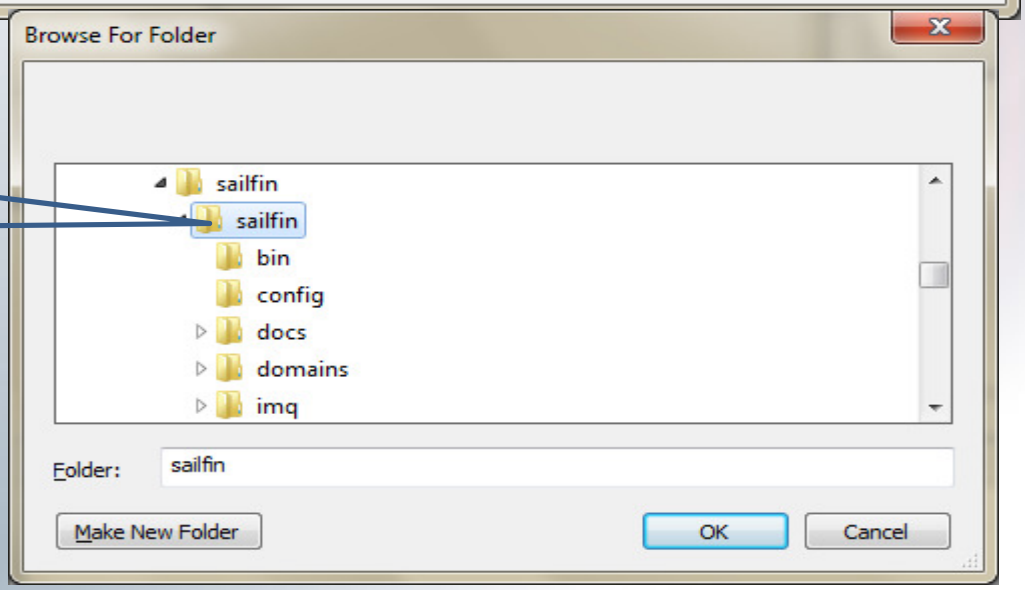
Choose ‘sailfin V2’
and click ‘Next’

Configuration

Choose ‘sailfin V2’ and click ‘Next’

Choose the directory where you installed sailfin and click ‘OK’

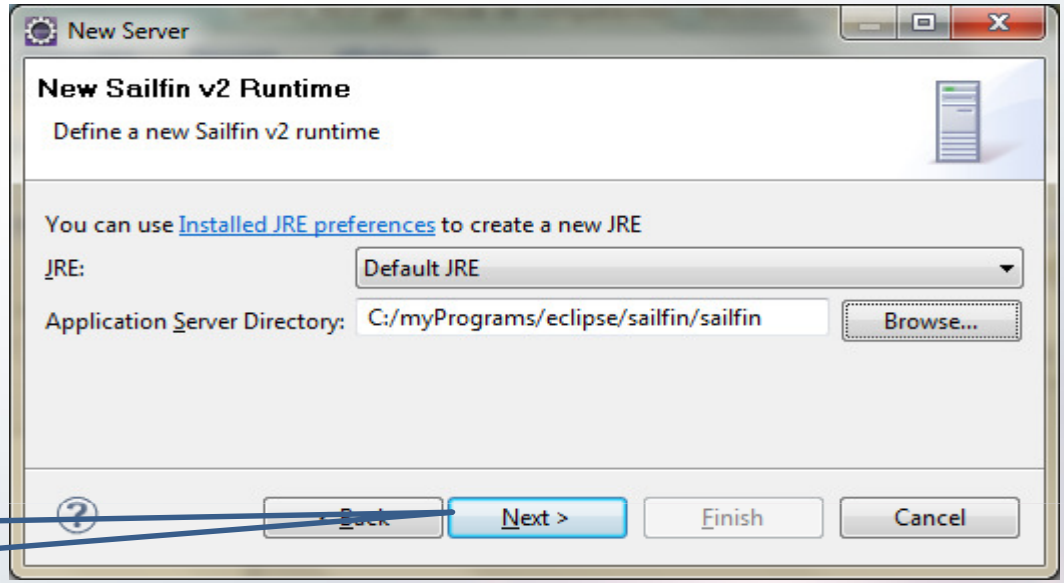


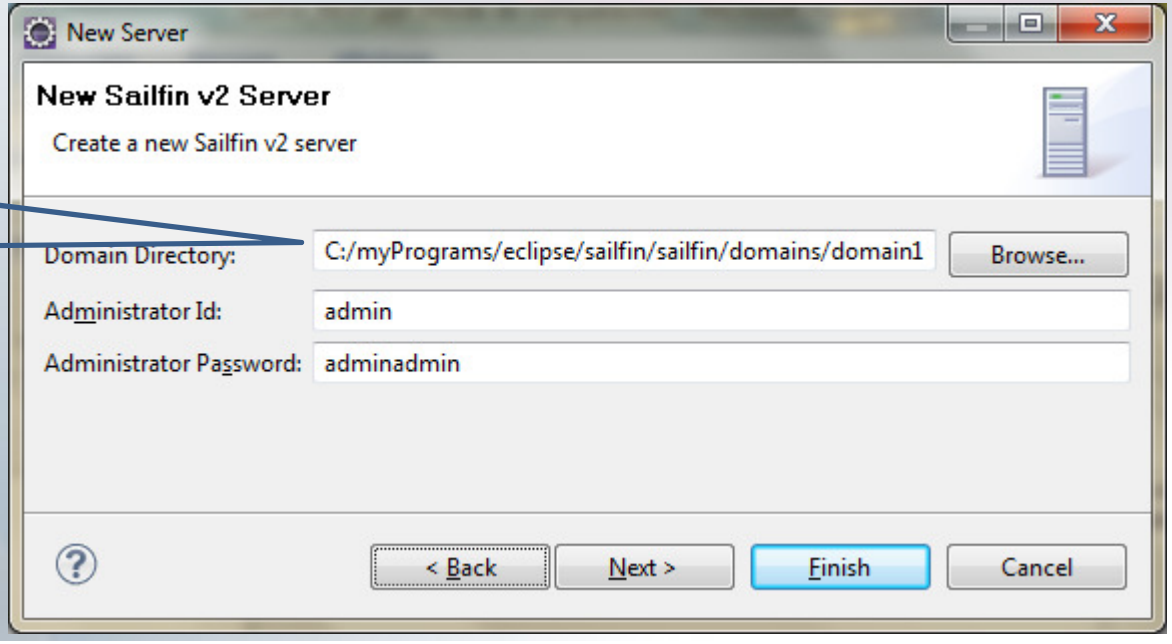


Configuration

Click ‘Next’

Configure your server and click ‘Finish’





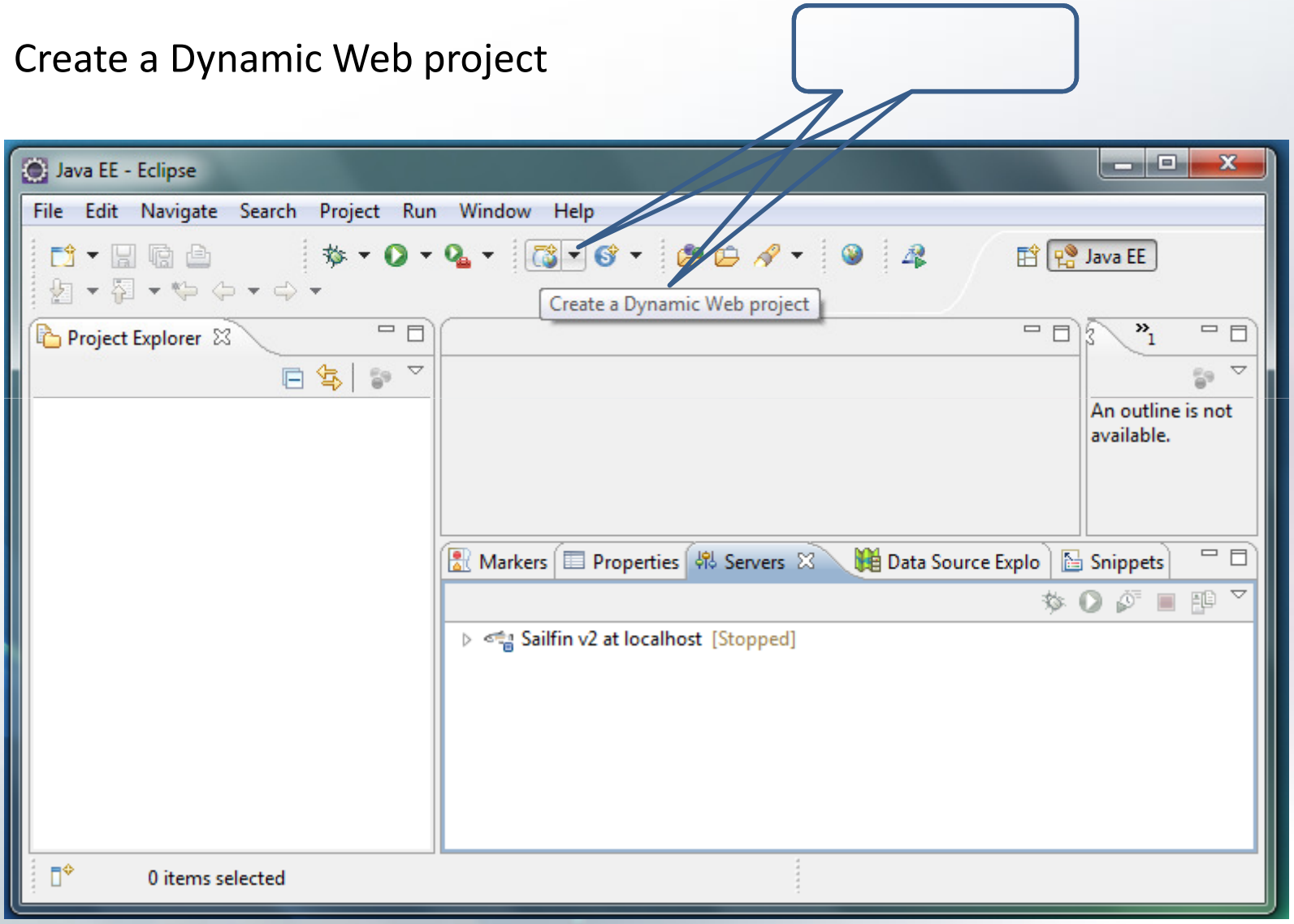
Create your first SIP Servlet

Create your first SIP Servlet

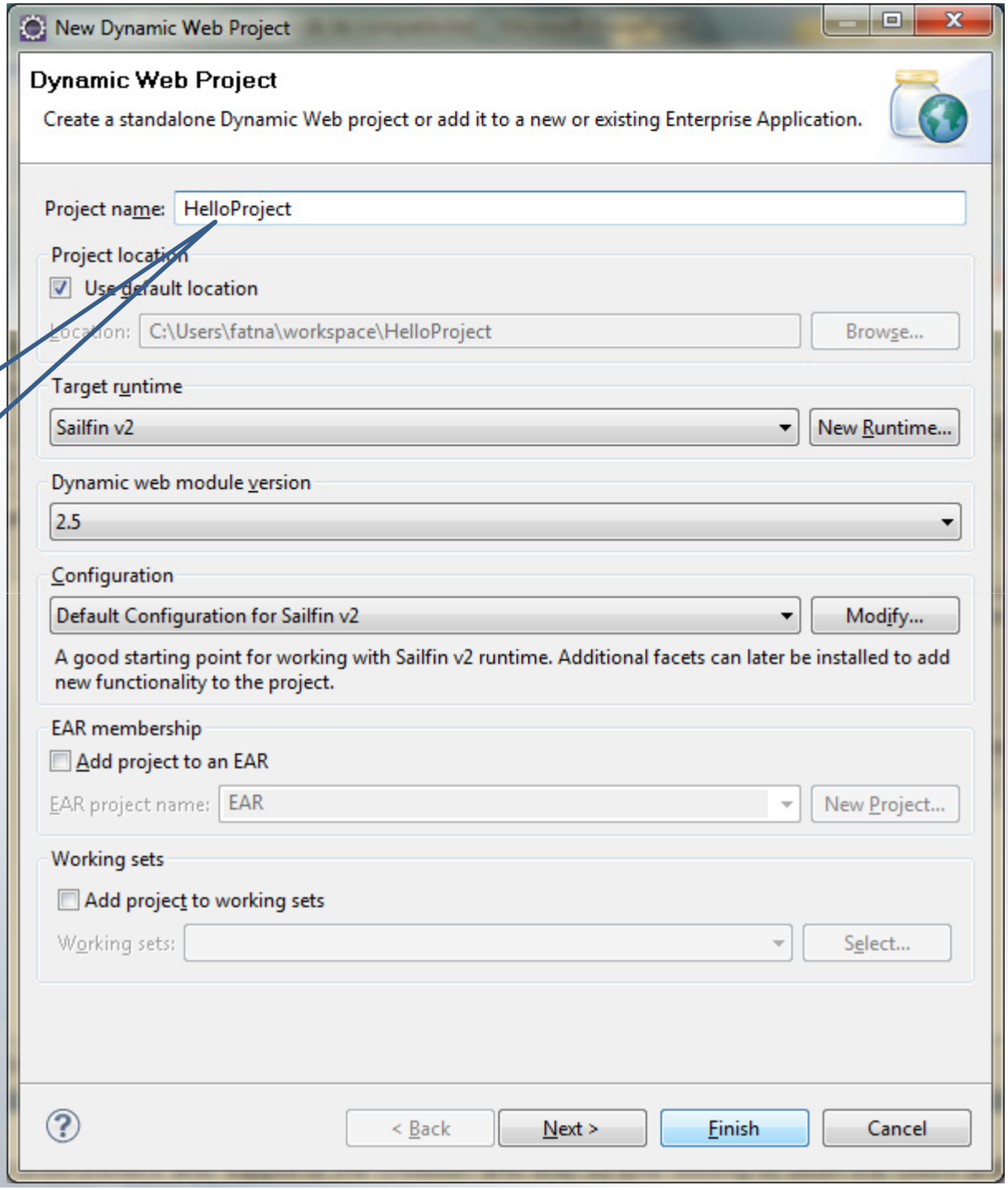
- Example: a simple “Hello World” SIP Servlet
  - Create a SIP Servlet
    - Create a Dynamic Web project
    - Create a package
    - Create a SIP Servlet
    - Add the Servlet methods (e.g. doInvite())
  - Start SailFin AS
  - Deploy and test the Servlet
    - Run the SIP Servlet on the sailfin server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

Create your first SIP Servlet

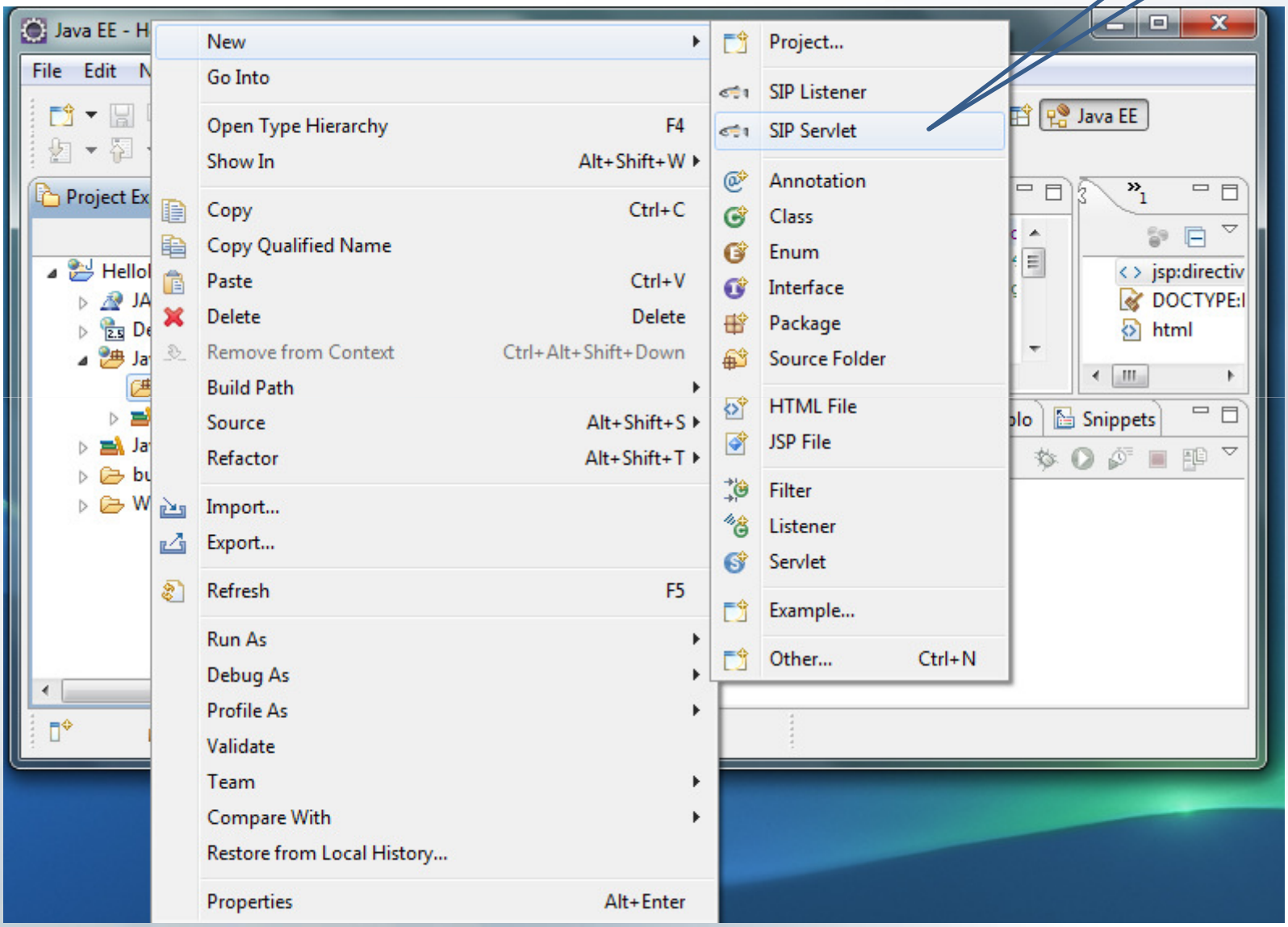
- Create a Dynamic Web project



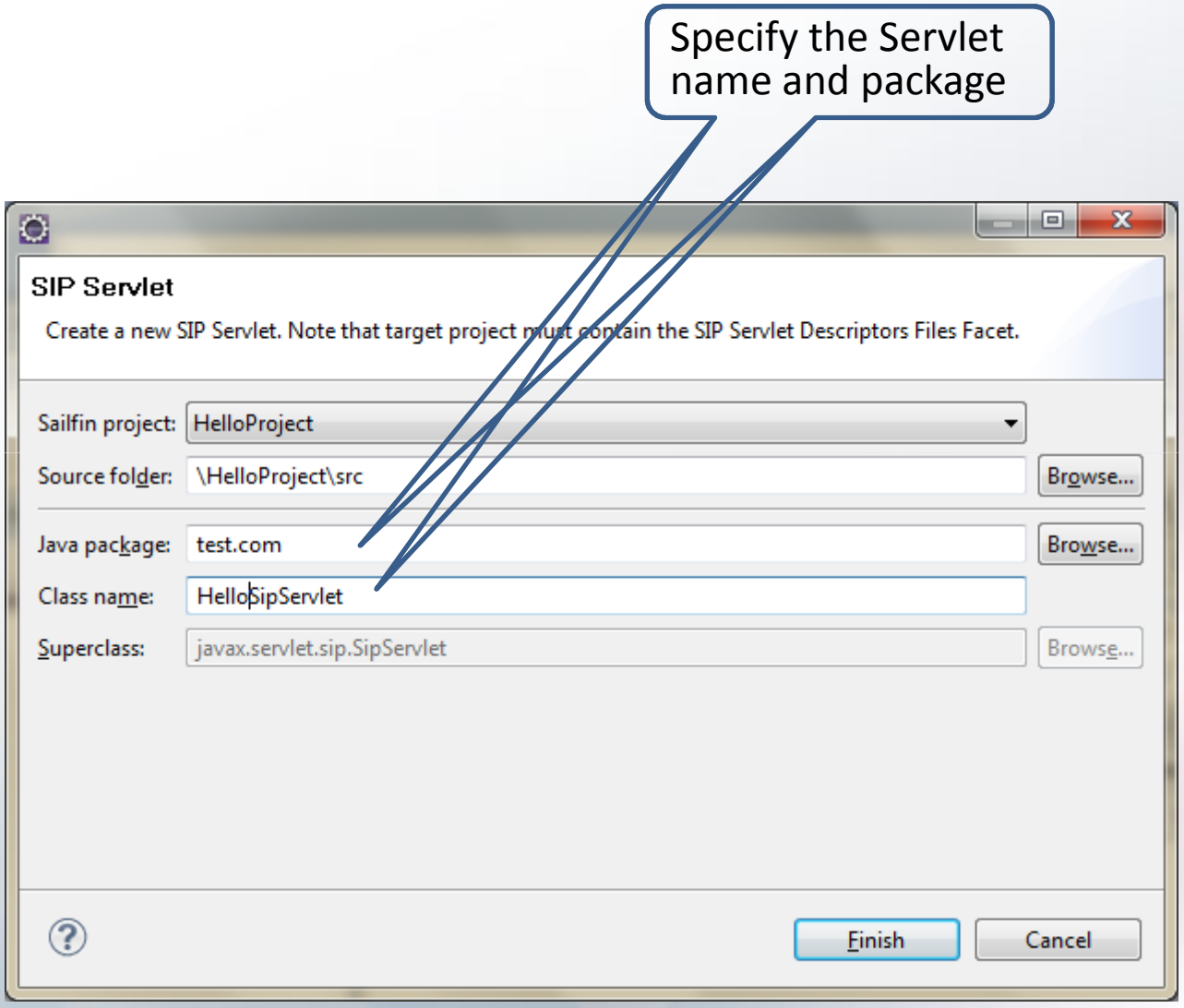
Name your project and click ‘Finish’



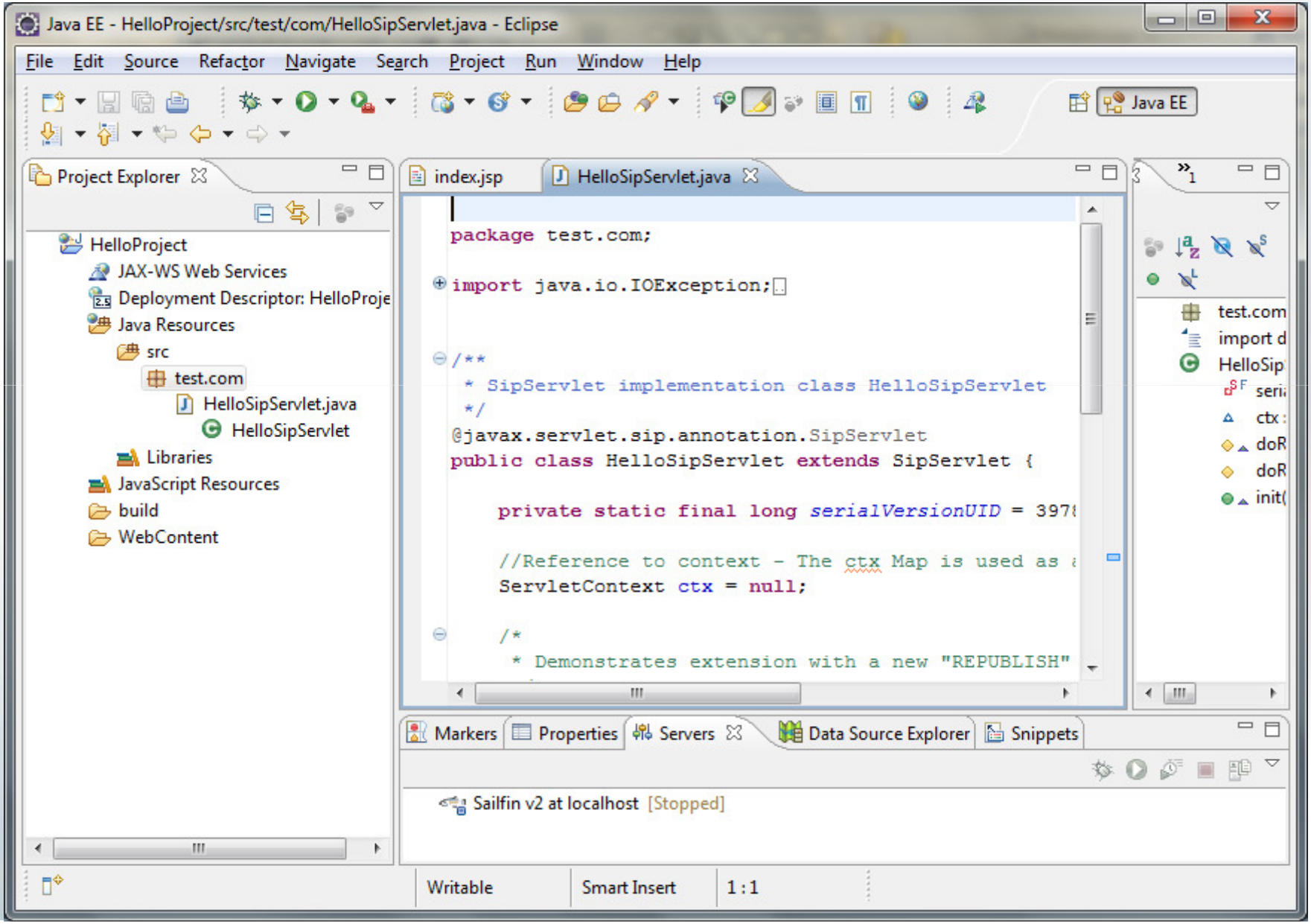
Create your first SIP Servlet



Create your first SIP Servlet



Create your first SIP Servlet

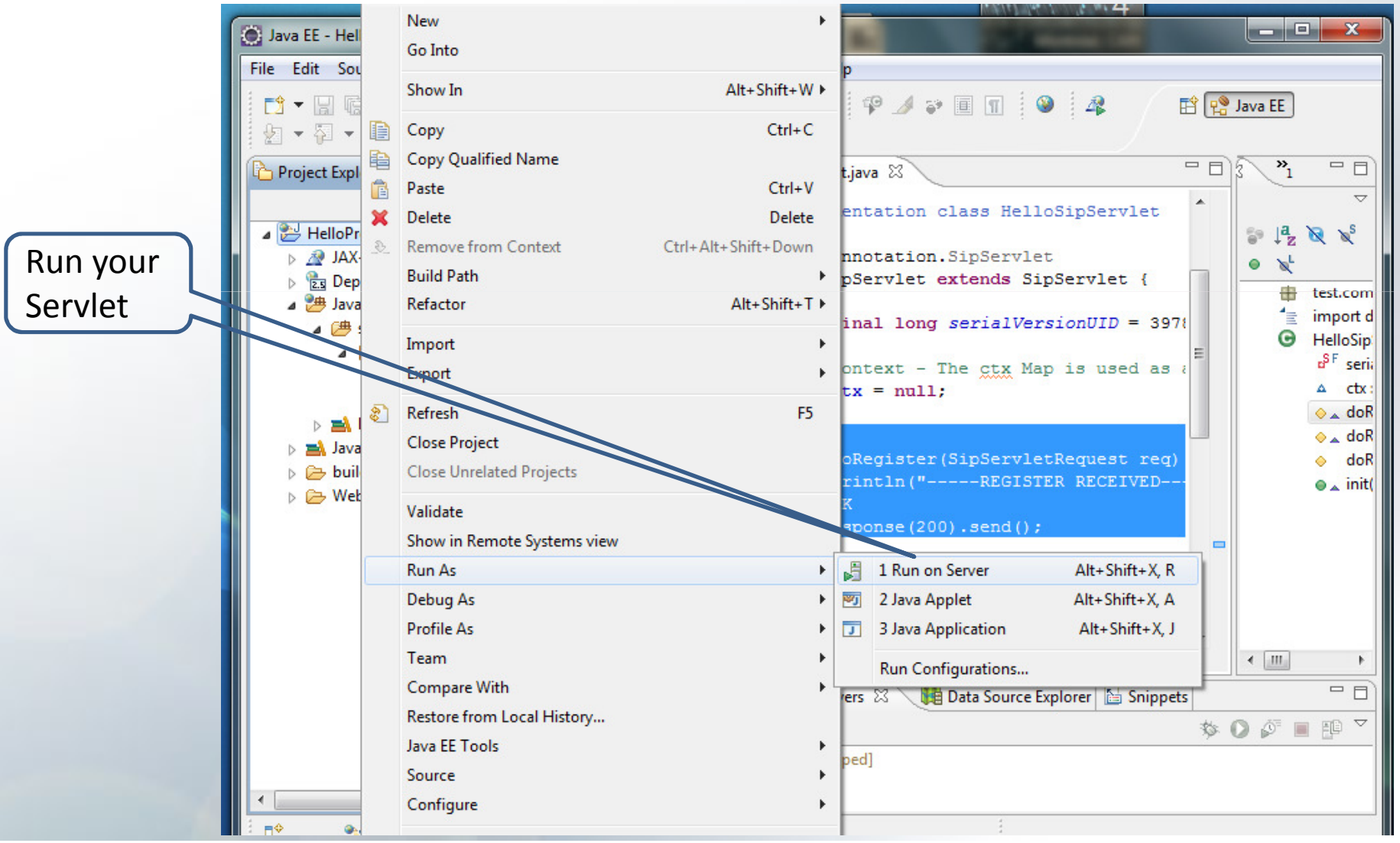


Create your first SIP Servlet

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

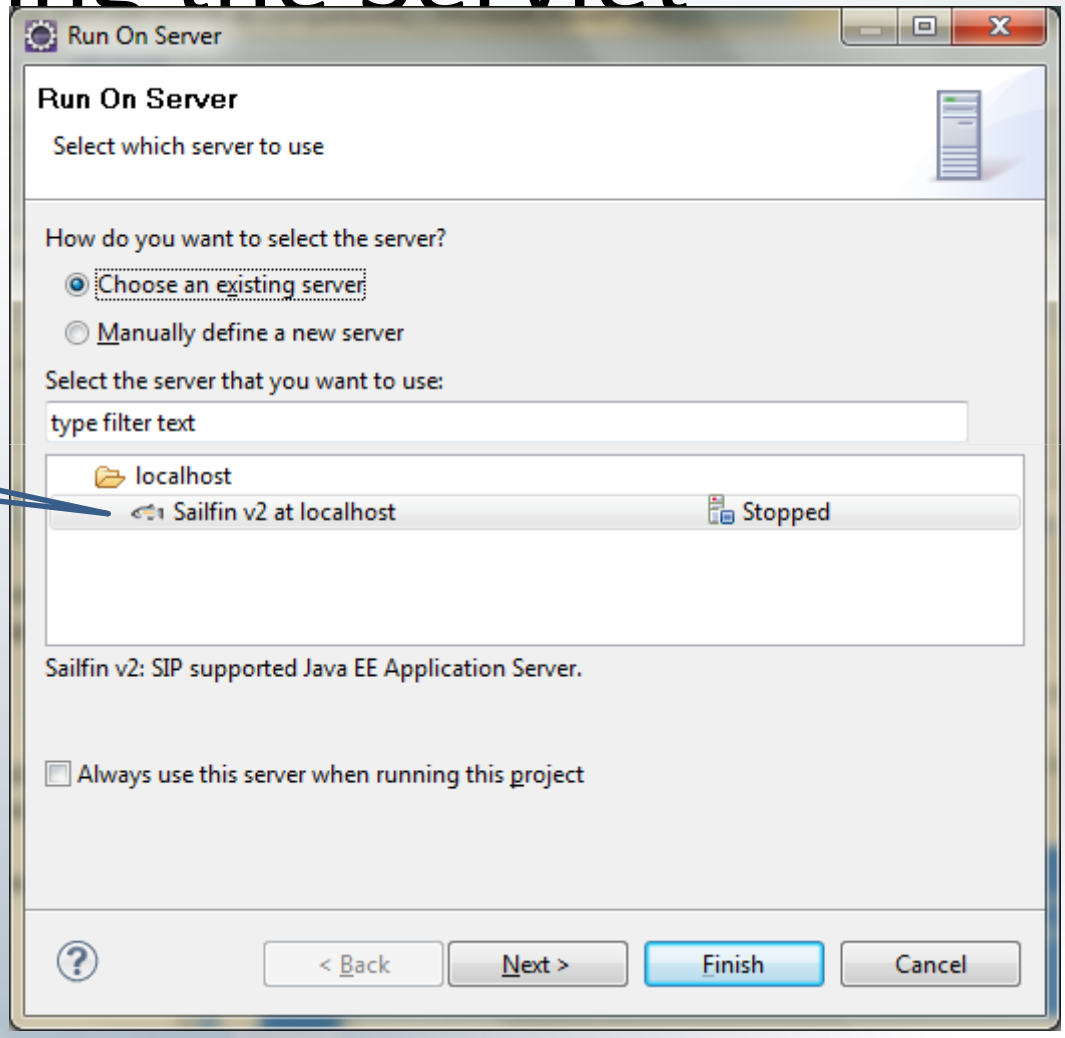
```
@Override
 protected void doRegister(SipServletRequest req) throws
                                    ServletException, IOException
{
        System.out.println("-----REGISTER RECEIVED-----");
        //send 200 OK
        req.createResponse(200).send();
}
```

Create your first SIP Servlet: Testing the Servlet

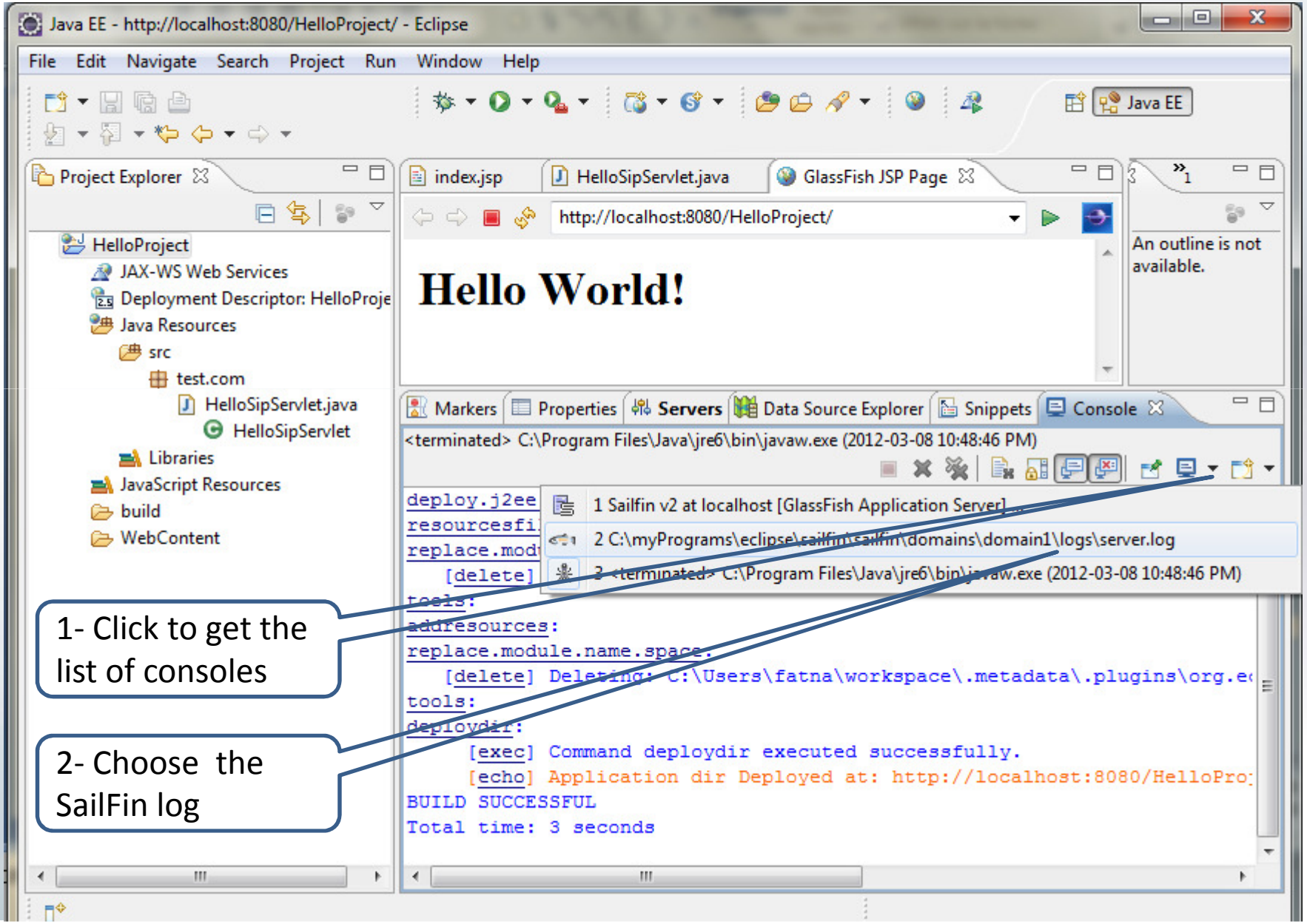


Create your first SIP Servlet: Testing the Servlet

Choose ‘SailFin V2’ and click ‘Finish’

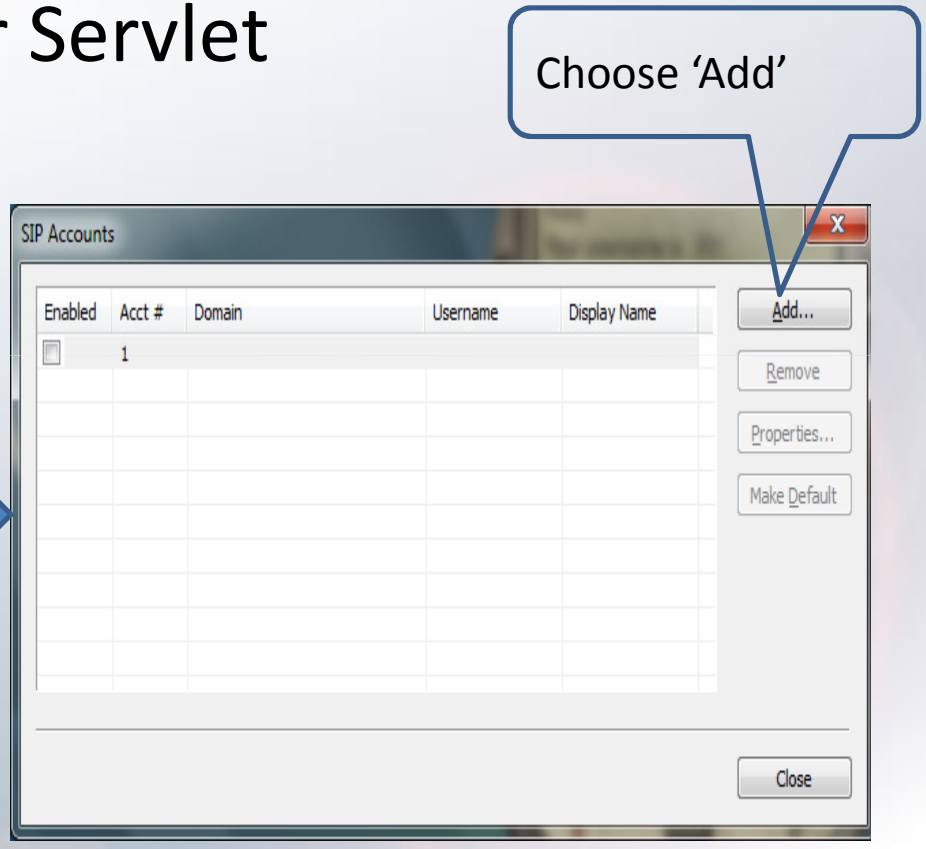


Create your first SIP Servlet:

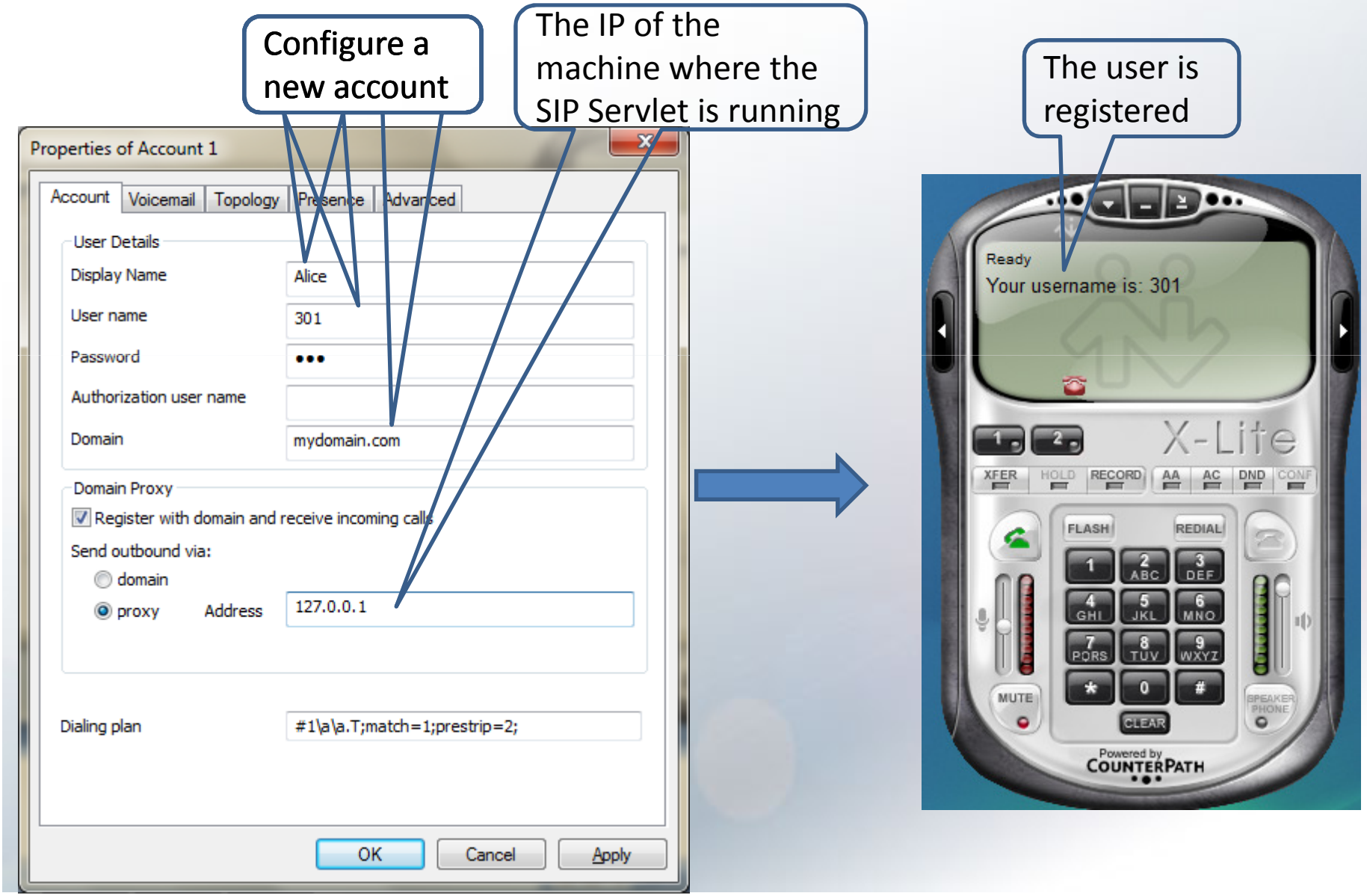


Create your first SIP Servlet

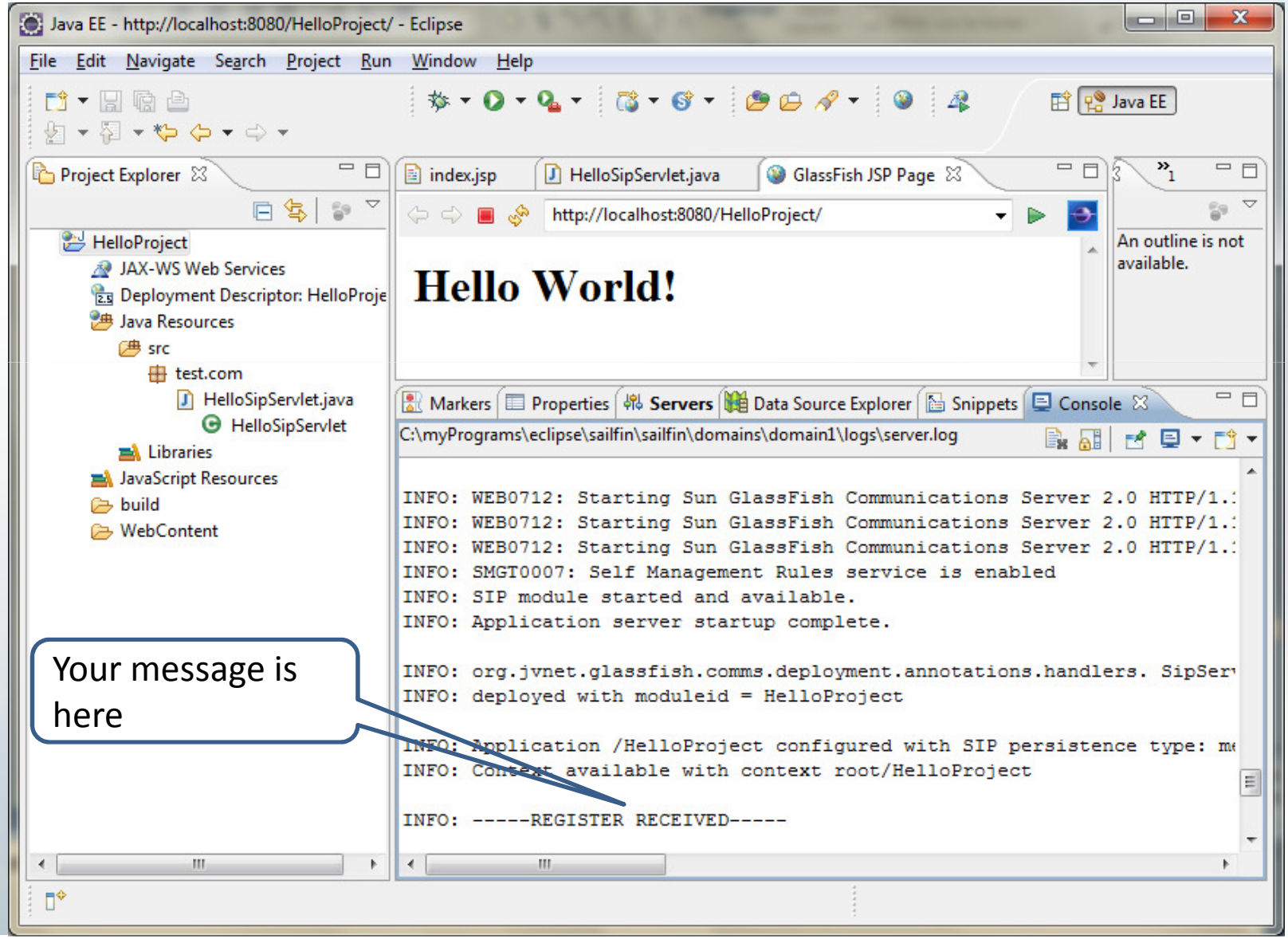
- Run X-Lite to test your Servlet



Create your first SIP Servlet



Create your first SIP Servlet



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 RESFul 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>
```

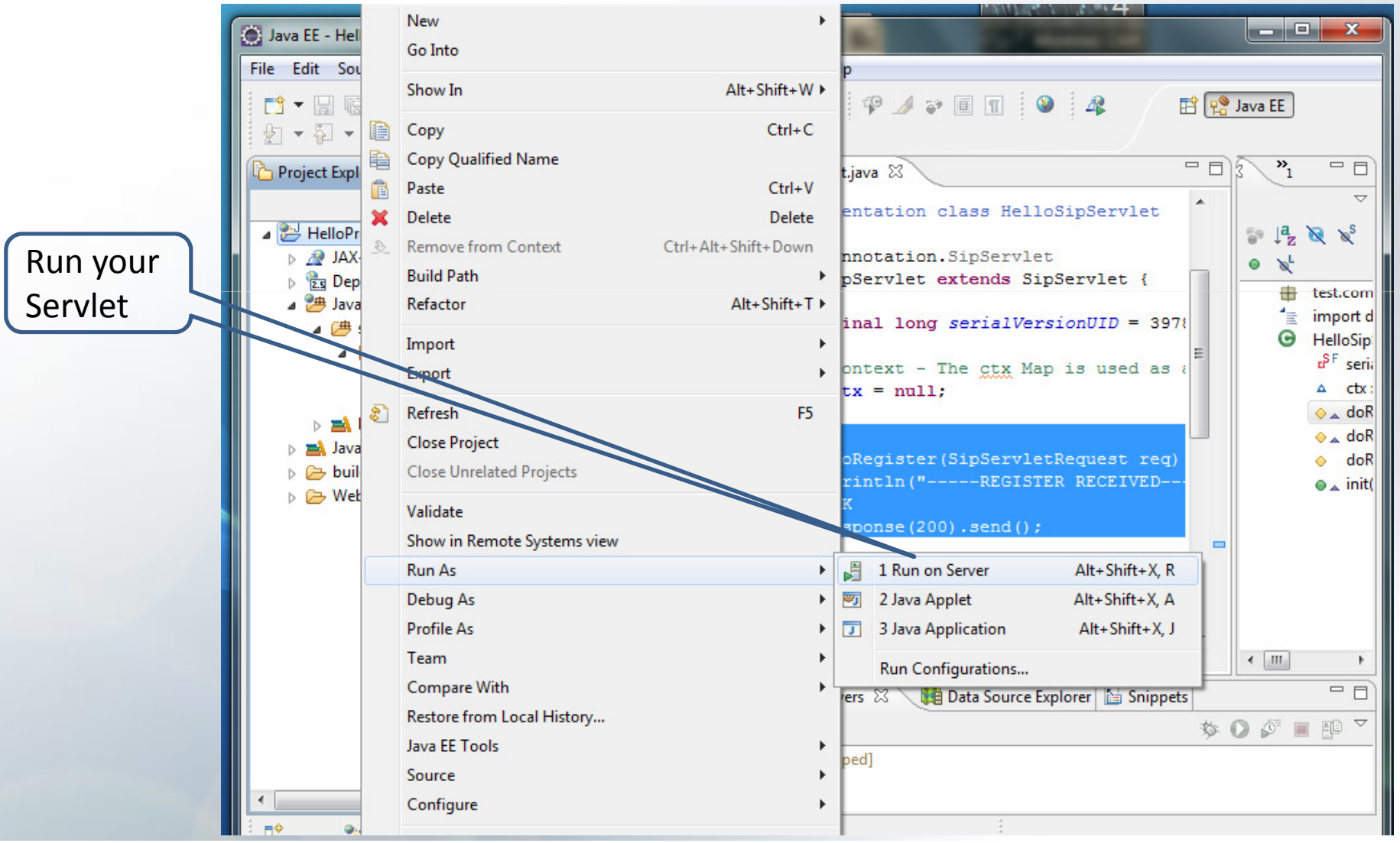
Create your first RESTful service

- 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)

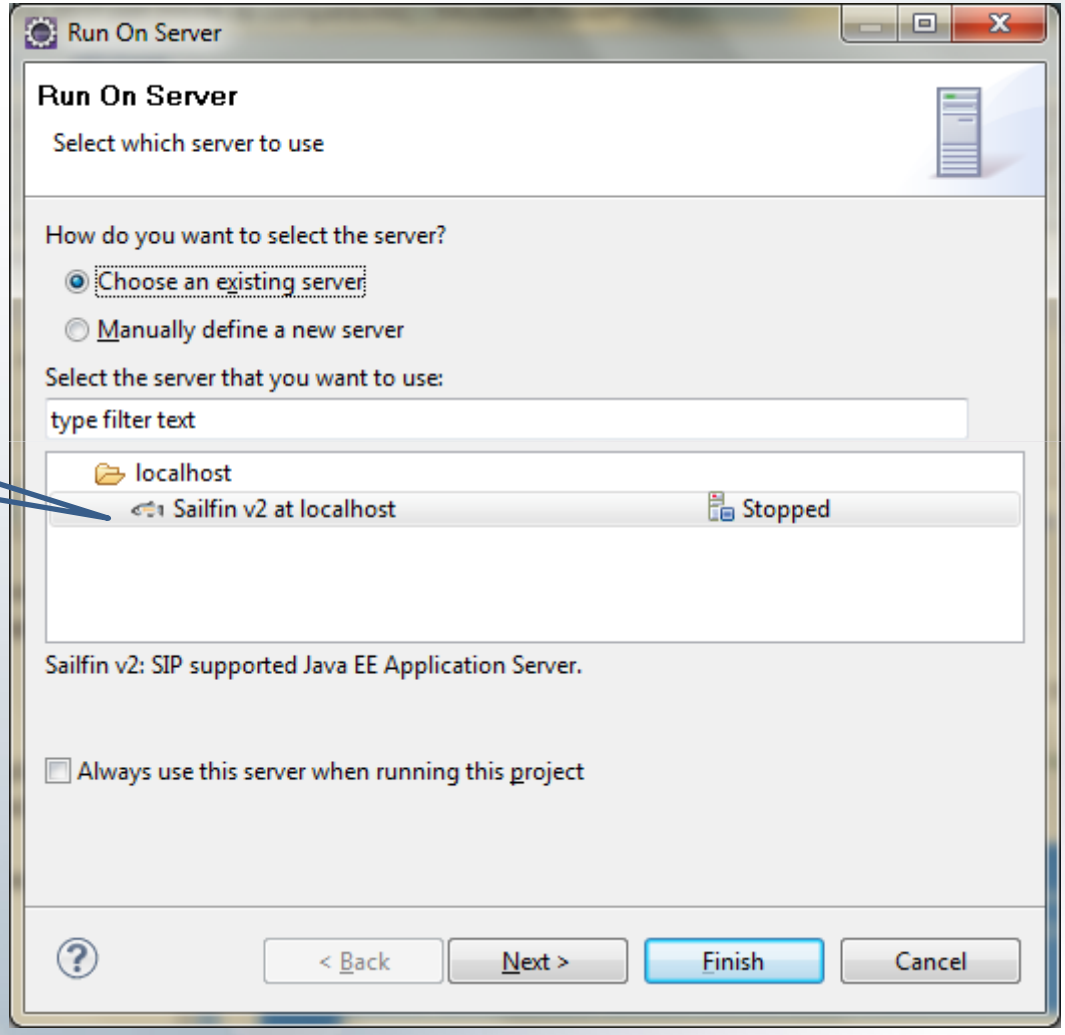
      and
      - the "hello" @Path annotation from your class file.
    - You should get the message “HTML: Hello Jersey“ (The browser requests the HTML representation of your resource)

Create your first RESTful service



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