



EVALUACIÓN Y NUEVAS TENDENCIAS DE ARQUITECTURA DE SOFTWARE (SI470)

Practica de Laboratorio

Ciclo 2019-2

Profesor:

Secciones: Todas

Creación de un Web Service REST usando Jersey

Requisitos:

- Eclipse
- Librería Jersey
- Apache Tomcat 7.0

Definir ubicación de Jersey

Para definir la ubicación de la librería Jaxrs, es necesario desempaquetar el archivo

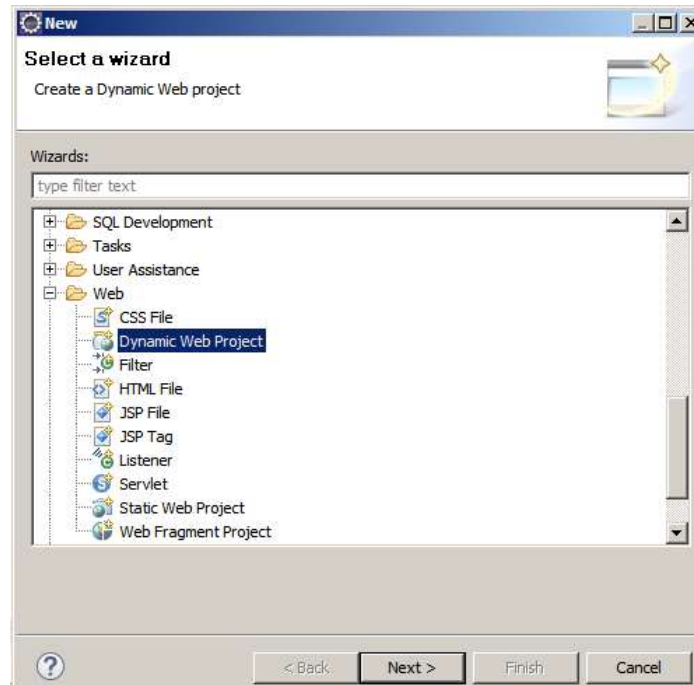
ideos	axis2-1.7.7-bin	4/17/2018 3:03 PM	Compressed (zippe...	22,479 KB
computer	dotNetFx40_Full_setup	4/13/2018 11:26 PM	Application	869 KB
Local Disk (C:)	eclipse-jee-mars-2-win32-x86_64	4/24/2018 5:37 PM	Compressed (zippe...	282,245 KB
app	eclipse-jee-oxygen-3-win32-x86_64	4/8/2018 11:20 PM	Compressed (zippe...	339,750 KB
data	Firefox Installer	4/13/2018 10:57 PM	Application	307 KB
PerfLogs	jaxrs-ri-2.25.1	4/17/2018 4:01 PM	Compressed (zippe...	4,848 KB
Program Files	jre-8u161-windows-x64	4/13/2018 11:10 PM	Application	69,657 KB
Program Files (x86)	putty	10/14/2013 7:17 PM	Application	484 KB

En el directorio c:\soft\

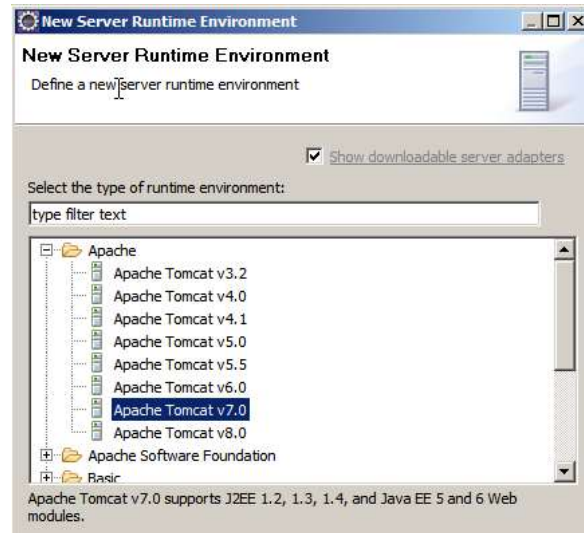
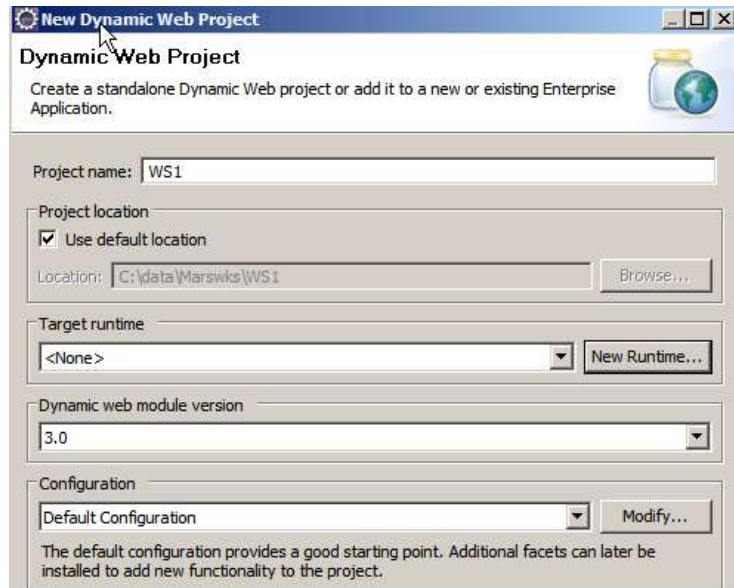
Computer > Local Disk (C:) > soft > jaxrs-ri				
Favorite	Include in library > Share with > New folder			
Name	Date modified	Type	Size	
api	4/22/2018 4:50 PM	File folder		
ext	4/22/2018 4:50 PM	File folder		
lib	4/22/2018 4:50 PM	File folder		
Jersey-LICENSE	4/22/2018 4:50 PM	Text Document		
third-party-license-readme	4/22/2018 4:50 PM	Text Document		

Crear un Proyecto Dynamic Web Project

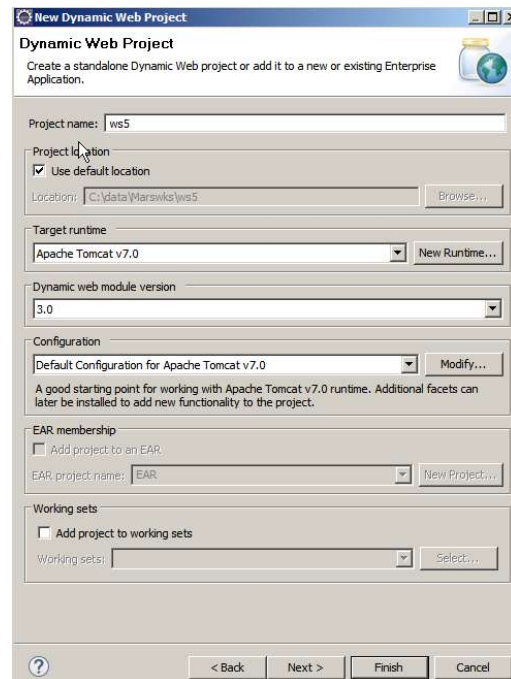
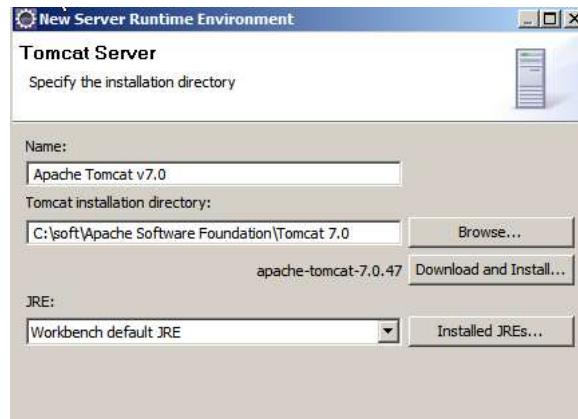
File/New /Other/Web/Dynamic Web Project



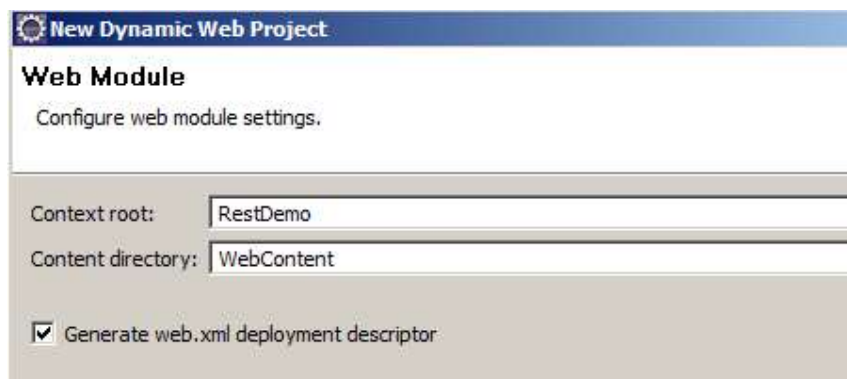
Dar nombre al proyecto, en la opción Target runtime seleccionar Apache Tomcat 7.0 si no está indicar el directorio de instalación



Indicar el directorio de Instalación de Apache Tomcat,

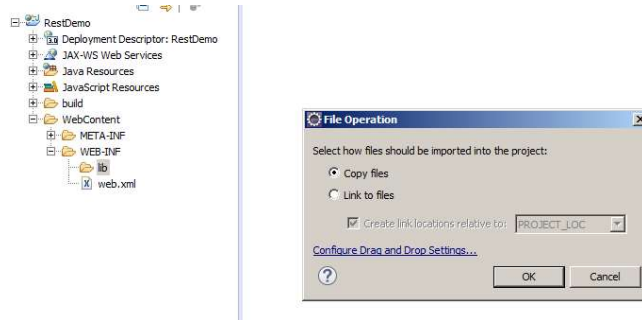


Asegurarse de marcar la opción Generate Web.xml deployment descriptor

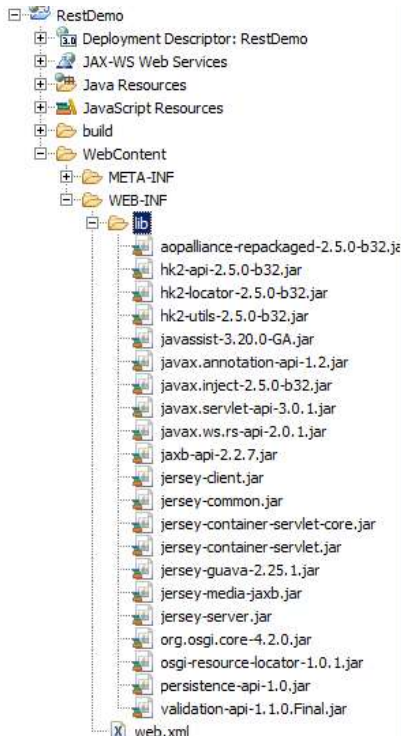


Una vez creado el proyecto, realizar lo siguiente:

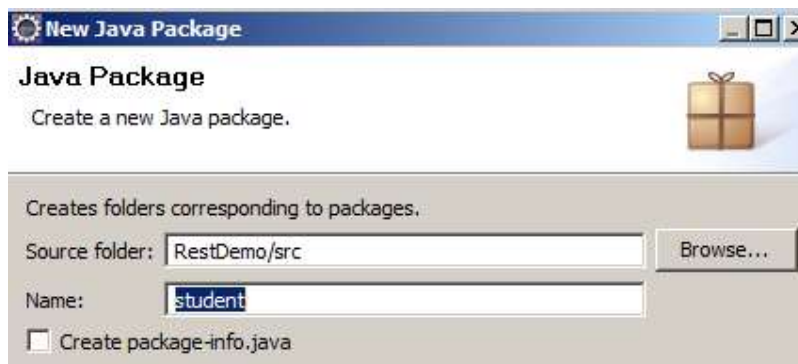
Copiar todas las librerías Jersey de los 3 directorios a Webcontent/WEN-INF/lib



Deben quedar copiadas todas las librerías en lib como lo muestra la siguiente figura



Crear un paquete llamado student en Javaresources/src dando click Derecho en src/New/Package



Crear una clase llamada Student

```
1 package student;
2 public class Student {
3     private String lastName;
4     private String firstName;
5     private int age;
6
7     public Student(String lastName, String firstName, int age) {
8         this.lastName = lastName;
9         this.firstName = firstName;
10        this.age = age;
11    }
12
13    public String getLastName() {
14        return lastName;
15    }
16
17    public void setLastName(String lastName) {
18        this.lastName = lastName;
19    }
20
21    public String getFirstName() {
22        return firstName;
23    }
24 }
```

Crear otra clase llamada Listado

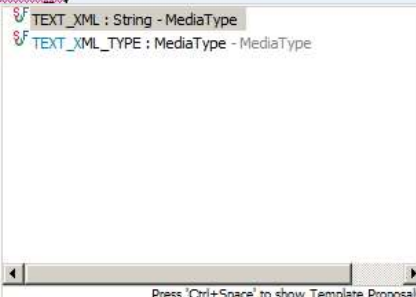
```
1 package student;
2 import java.util.ArrayList;
3 import java.util.List;
4 import java.util.Random;
5 public class Listado {
6     private static String [] lastNames={"Perez","Juarez","Burga","Penaranda"};
7     private static String [] firstNames ={"Luis","Maria","Jorge","Janet"};
8
9     public String listar(){
10        List<Student> studenList= new ArrayList<>();
11        Random random =new Random();
12
13        for (int i=0;i<=3;i++) {
14            String tempFirstName = firstNames [random.nextInt(firstNames.length)];
15            String tempLastName = lastNames [random.nextInt(lastNames.length)];
16
17            int age=18+random.nextInt(20);
18
19            Student tempStudent = new Student (tempLastName, tempFirstName, age);
20            studenList.add(tempStudent);
21        }
22        String cadena="";
23        for (Student temp : studenList) {
24            cadena=cadena+temp;
25        }
26        return cadena;
27    }
28
29 }
```

Creación del Web Service REST a través de la Clase Asistencia

```
1 package student;
2
3 public class Asistencia {
4     public Listado listado = new Listado();
5
6     public String mostrar()
7     {
8         String response=null;
9         return response;
10    }
11 }
12
13 }
14
```

Colocar la Anotaciones @PATH, @GET y @Produces, agregar los import correspondientes para cada anotación,

```
2
3 import javax.ws.rs.GET;
4 import javax.ws.rs.Path;
5 import javax.ws.rs.Produces;
6 import javax.ws.rs.core.MediaType;
7
8 //uso de anotaciones
9 @Path("/asistencia")
10 public class Asistencia {
11     public Listado listado = new Listado();
12
13     @GET
14     @Produces(MediaType.TEXT_XML)
15     public String mostrar()
16     {
17         String response=null;
18         return response;
19     }
20 }
21
22 }
23
```



Press 'Ctrl+Space' to show Template Proposals

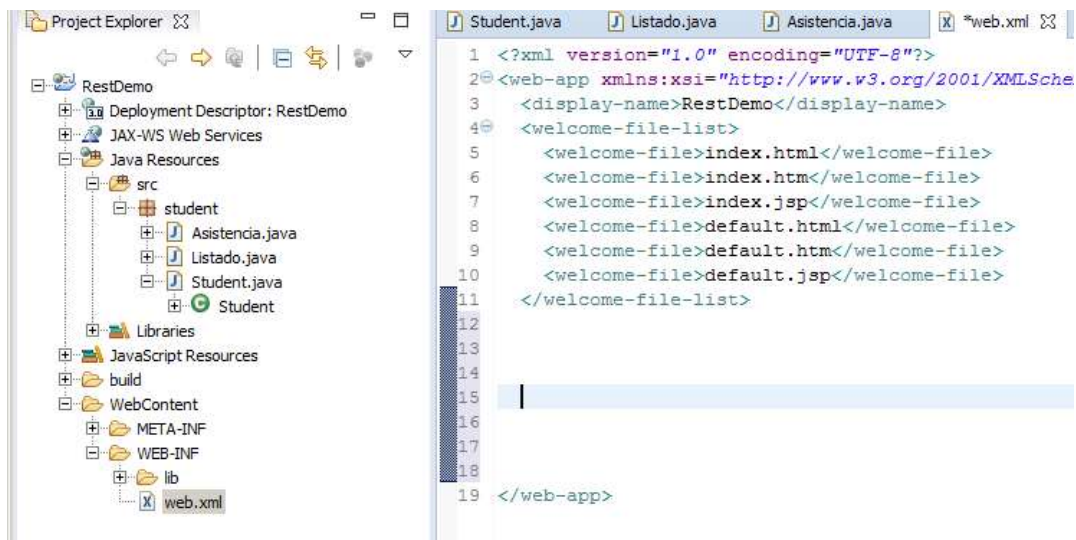
Código del método mostrarXML


```

1 package student;
2
3 import javax.ws.rs.GET;
4 import javax.ws.rs.Path;
5 import javax.ws.rs.Produces;
6 import javax.ws.rs.core.MediaType;
7
8 //uso de anotaciones
9 @Path("/asistencia")
10 public class Asistencia {
11     public Listado listado = new Listado();
12
13     @GET
14     @Produces(MediaType.TEXT_XML)
15     public String mostrarXML() {
16
17         String response = "<?xml version = '1.0'?>" +
18             "<lista>" + listado.listar() + "</lista>"; //se agrega el listado de alumnos
19
20         return response;
21     }
22 }
23
24 }
25

```

Ir al archivo web.xml en la carpeta WEB-INF, click en la parte inferior donde dice Source, web.xml es el archivo descriptor donde se registran las rutas y clases a exponer como webservices



Agregar la siguiente estructura xml

```

<servlet>
<servlet-name>JAVA WS</servlet-name>

```



```

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>student</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>JAVA WS</servlet-name>

<url-pattern>/rest/*</url-pattern>

</servlet-mapping>

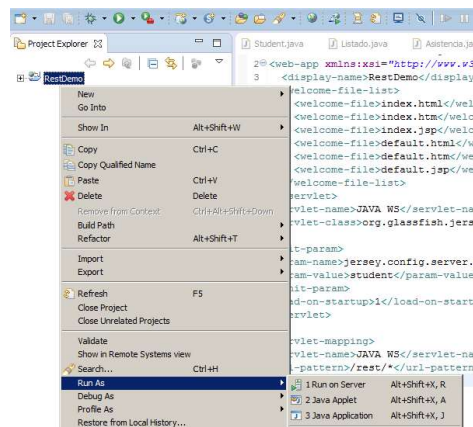
```

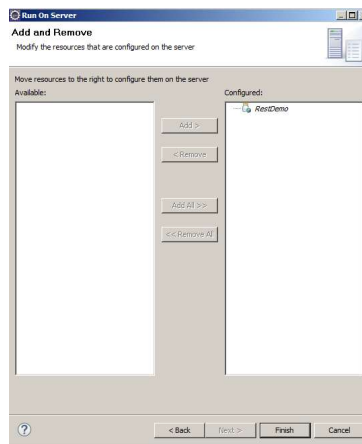
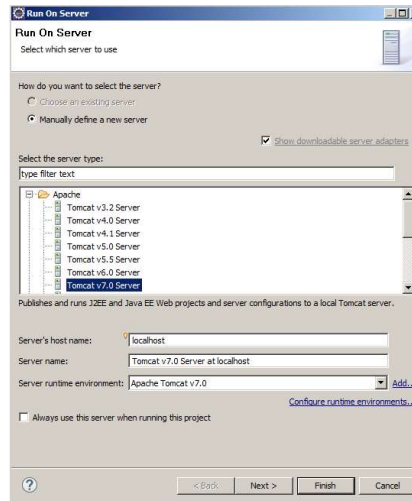
```

20 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/
3   <display-name>RestDemo</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12  <servlet>
13    <servlet-name>JAVA WS</servlet-name>
14    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
15
16    <init-param>
17      <param-name>jersey.config.server.provider.packages</param-name>
18      <param-value>student</param-value>
19    </init-param>
20    <load-on-startup>1</load-on-startup>
21  </servlet>
22
23  <servlet-mapping>
24    <servlet-name>JAVA WS</servlet-name>
25    <url-pattern>/rest/*</url-pattern>
26  </servlet-mapping>
27
28
29
30 </web-app>

```

Iniciar el proyecto en el servidor dando Click derecho en el proyecto/Run as/Run on Server





Asegurarse que el proyecto este seleccionado en Configured

Consultar esta dirección URL

<http://localhost:8080/RestDemo/rest/asistencia>



Agregar un método mas a la clase y ver el resultado en HTML, ver el resultado desde un navegador el cual por defecto siempre ve la información en HTML

```
package student;

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

//uso de anotaciones
@Path("/asistencia")
public class Asistencia {
    public Listado listado = new Listado();

    @GET
    @Produces(MediaType.TEXT_XML)
    public String mostrarXML() {

        String response="<?xml version = '1.0'?>" +
        "<lista>" + listado.listar() + "</lista>"; //se agrega el listado de alumnos
        return response;
    }

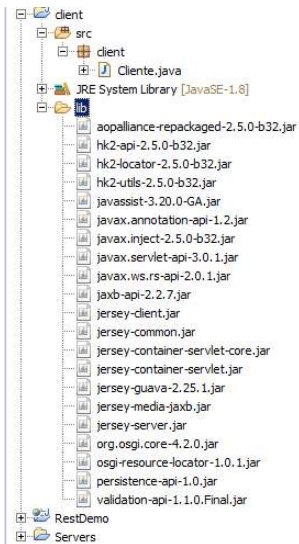
    @GET
    @Produces(MediaType.TEXT_HTML)
    public String mostrarHTML() {
        String response="<h1>" + listado.listar() + "</h1>"; //se agrega el listado de alumnos
        return response;
    }
}
```



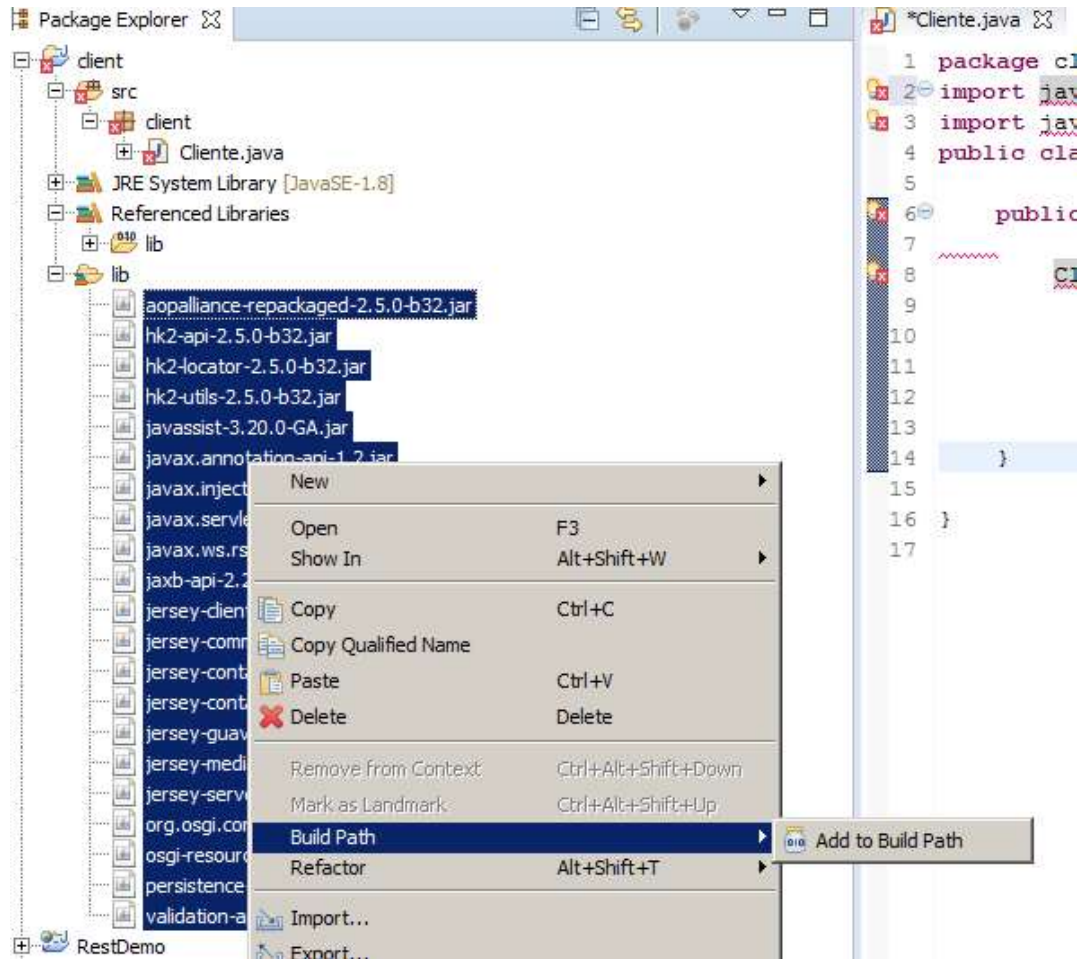
Creación de un cliente que consume el webservice

Crear un proyecto java llamado client y una clase llamada Cliente (con método main)

Crear un folder lib y volver a copiar las librerías jersey



Click derecho en el folder lib, marcar todas las librerias/Build Path/ Add to BuildPath



Crear clase consumidora del Webservice

Crear una clase consumidora en un proyecto java, denominar a la clase Cliente

```

1 package client;
2 import javax.ws.rs.client.Client;
3 import javax.ws.rs.client.ClientBuilder;
4 import javax.ws.rs.client.WebTarget;
5 import javax.ws.rs.core.MediaType;
6 public class Cliente {
7
8     public static void main(String[] args) {
9
10         Client consumidor=ClientBuilder.newClient();
11         WebTarget target=consumidor.target("http://localhost:8080/RestDemo/rest/asistencia");
12         System.out.println(
13             target.request(MediaType.APPLICATION_XML).get(String.class) );
14
15     }
16 }
17
18 }

```

Ejecutar la clase y verificar la respuesta en la consola