

Ejemplo REST

A-Crear Dominio

1-Crear un proyecto del tipo Java Application llamado Dominio en la carpeta C:\EjercicioREST\.

2-Crear paquete uy.edu.ort.dominio

3-Crear la clase PersonaDTO en el paquete uy.edu.ort.dominio

4-Codigo de la clase PersonaDTO

```
@XmlRootElement
public class PersonaDTO {
    private int id;
    private String nombre;
    private String apellido;
    private String direccion;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }
}
```

B-Creación de un módulo EJB para Negocio

En este módulo EJB se encapsula la lógica de negocio de nuestra aplicación. Este módulo contiene los Session Bean en donde se desarrolla la lógica de negocio. Se utiliza el servidor GlassFish 4.0 y la versión JEE 6

1-Crear Modulo EJB

- 1.1-Crear un Enterprise Application con el nombre AppGestionPersona
- 1.2-Crear un módulo EJB con el nombre AppGestionPersonaNegocio-ejb
- 1.3-Ingresa la ubicación del proyecto en la carpeta C:\EjercicioREST\
- 1.4-Asociarlo al Enterprise Application: AppGestionPersona
- 1.5-Agregar en la librerías el proyecto Dominio.

2-Crear Session Beans

2.1-Creamos el paquete uy.edu.ort.gestion.persona.negocio

2.2- Creamos una Session Beans PersonaNegocioSB de tipo Stateless e interface Local en el paquete uy.edu.ort.gestion.persona.negocio

2.3 Código de la interface Local

```
@Local
public interface PersonaNegocioSBLocal {
    public void alta(PersonaDTO personaDTO);
    public void eliminar (PersonaDTO personaDTO);
    public void modificar(PersonaDTO personaDTO);
    public List<PersonaDTO> listaPersona();
}
```

2.4 Código de Session Beans

```
@Stateless
public class PersonaNegocioSB implements PersonaNegocioSBLocal {
```

```
    @Override
    public void alta(PersonaDTO persona) {
        System.out.println("Se dio de alta la persona"+persona. getNombre());
    }
```

```
    @Override
    public void eliminar(PersonaDTO persona) {
        System.out.println("Se dio de baja la persona"+persona.getId());
    }
```

@Override

```
public void modificar(PersonaDTO persona) {  
    System.out.println("Se modifico la persona"+persona.getId());  
}
```

@Override

```
public List<PersonaDTO> listaPersona() {  
    PersonaDTO personaDTO = new PersonaDTO();  
    personaDTO.setApellido("Perez");  
    personaDTO.setDireccion("Lejos");  
    personaDTO.setNombre("Juan");  
    personaDTO.setId(1);  
    List<PersonaDTO> listaPersona = new ArrayList();  
    listaPersona.add(personaDTO);  
    return listaPersona;  
}
```

```
}
```

```
}
```

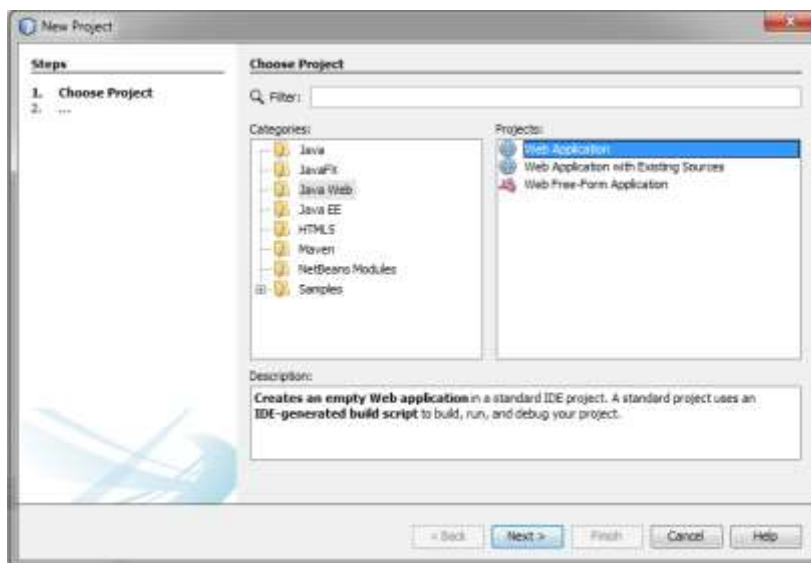
C-Crear el Web Services REST

En este módulo Web se encapsula los servicios Web con REST que son expuestos para ser consumidos por los clientes.

Este módulo contiene los Session Bean en donde se desarrolla los servicios Web con REST.

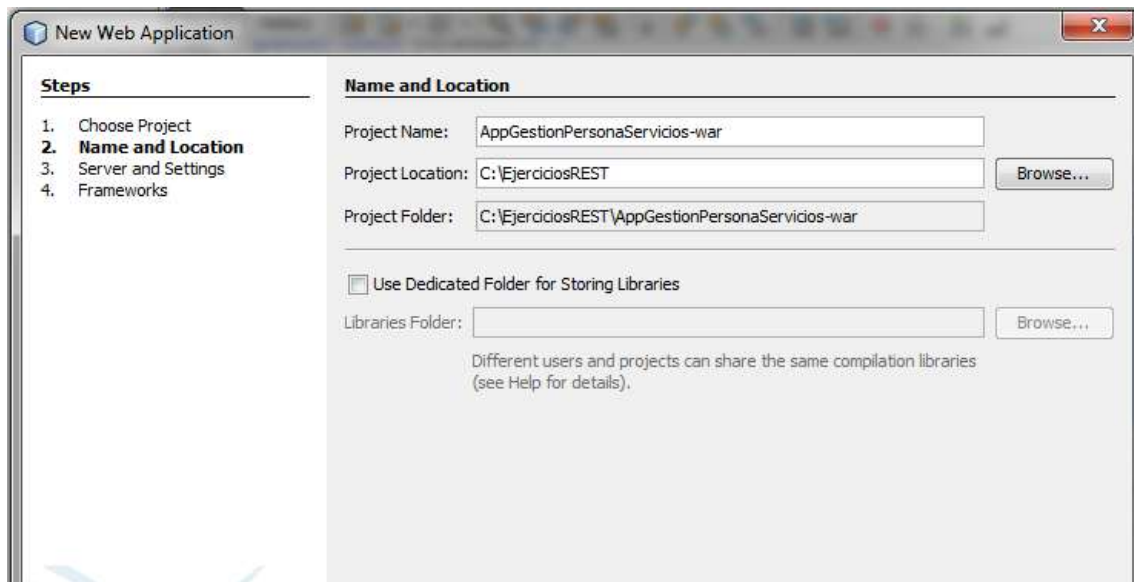
1-Crear un Módulo Web

1.1-En la IDE de NetBeans, crear un proyecto seleccionando de la categoría Java Web la opción Web Application

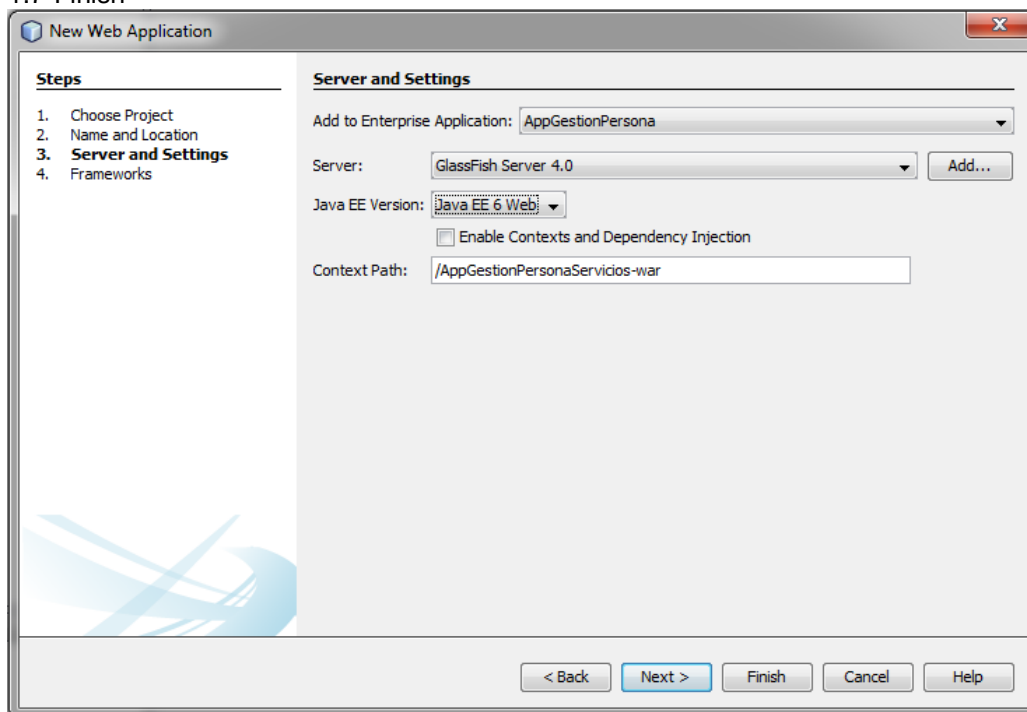


1.2-Ingresar el nombre del proyecto AppGestionPersonaServicios-war

1.3-Ingresar la ubicación del proyecto en la carpeta C:\EjercicioREST\



- 1.4-Add to Enterprise Application: AppGestionPersona
- 1.5-Seleccionamos el servidor Glassfish
- 1.6-Seleccionamos la versión de JEE 6
- 1.7-Finish



- 1.8-Agregar en la librería el módulo EJB (proyecto) que contiene la lógica de negocio (AppGestionPersonaNegocio-ejb) y el Dominio.

2-Crear un servicio REST

2.1-Creamos el paquete uy.edu.ort.gestion.persona.servicios

2.2-Creamos una Session Beans PersonaServicioSB de tipo Stateless sin Interface en el paquete uy.edu.ort.gestion.persona.servicios.

2.3-Codigo de PersonaServicioSB

@Stateless

@Path("/personas")

public class PersonaServicioSB {

 @EJB

 private PersonaNegocioSBLocal bLocal;

 @POST

 @Consumes("application/json")

 @Produces("application/json")

 public PersonaDTO alta(PersonaDTO persona) {

 bLocal.alta(persona);

 return persona;

 }

@PUT

@Path("/{id}")

@Consumes("application/json")

```
public Response modificar(@PathParam("id") Integer id, PersonaDTO persona) {
```

```
    Response response;
```

```
    try {
```

```
        PersonaDTO personaDTO = new PersonaDTO();
```

```
        personaDTO.setId(id);
```

```
        bLocal.modificar(persona);
```

```
        response = Response.ok(persona).build();
```

```
    } catch (Exception ex) {
```

```
        response = Response.serverError().entity(ex).build();
```

```
    }
```

```
    return response;
```

```
}
```

@DELETE

@Path("/{id}")

```
public Response eliminar(@PathParam("id") Integer id) {
```

```
    Response response;
```

```
    try {
```

```
        PersonaDTO personaDTO = new PersonaDTO();
```

```
        personaDTO.setId(id);
```

```
        bLocal.eliminar(personaDTO);
```

```
        response = Response.ok(id).build();
```

```
    } catch (Exception ex) {
```

```
        response = Response.serverError().entity(ex).build();
```

```
    }
```

```
    return response;
```

```
}
```

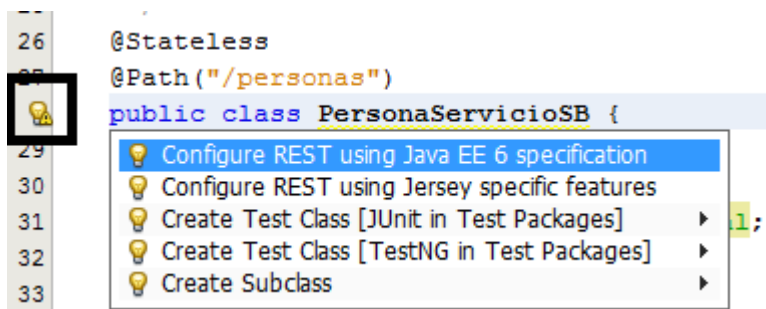
@GET

@Produces("application/json")

```
public List<PersonaDTO> listaPersona() {  
    return bLocal.listaPersona();  
}
```

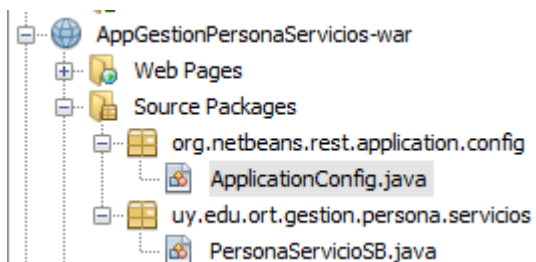
}2.4-Crear el servicio

2.4.1- Clickear en la lamparita como se muestra en la siguiente imagen



2.4.2-Selección Configure REST using Java EE 6 specification

2.4.3-Se generara una clase llamada ApplicationConfig



D-Build y Deploy

- 1.Hacer build del proyecto AppGestionPersona
- 2.Hacer deploy del proyecto AppGestionPersona

```
INFO: app.gestion.persona.entidades.Persona actually got transformed
INFO: EclipseLink, version: Eclipse Persistence Services - 2.5.0.v20130507-0faad2b
INFO: File:/C:/Ejercicios/JEE/AppGestionPersona/dist/qrDeploy/AppGestionPersona/AppGestionPersonaPersistencia-ejb.jar/ GestionPersonaPU login successful
INFO: EJBS181:Portable JNDI names for EJB PersonaRegistroES: [java:global/AppGestionPersona/AppGestionPersonaPersistencia-ejb/PersonaRegistroES, java:global/AppGestionPersona/AppGestionPersonaPersistencia-ejb/PersonaRegistroES]
INFO: EJBS181:Portable JNDI names for EJB PersonaPersistenciaES: [java:global/AppGestionPersona/AppGestionPersonaPersistencia-ejb/PersonaPersistenciaES, java:global/AppGestionPersona/AppGestionPersonaPersistencia-ejb/PersonaPersistenciaES]
INFO: EJBS181:Portable JNDI names for EJB PersonaServicioES: [java:global/AppGestionPersona/AppGestionPersonaServicios-war/PersonaServicioES, java:global/AppGestionPersona/AppGestionPersonaServicios-war/PersonaServicioES]
INFO: Registering the Jersey servlet application, named org.netbeans.csf.application.config.ApplicationConfig, at the servlet mapping /webresources/*, with the Application class org.netbeans.csf.application.config.ApplicationConfig
INFO: Loading application [AppGestionPersonaAppGestionPersonaServicios-war.war] at [AppGestionPersonaServicios-war]
INFO: AppGestionPersona se ha desplegado correctamente en 0.366 milisegundos.
```

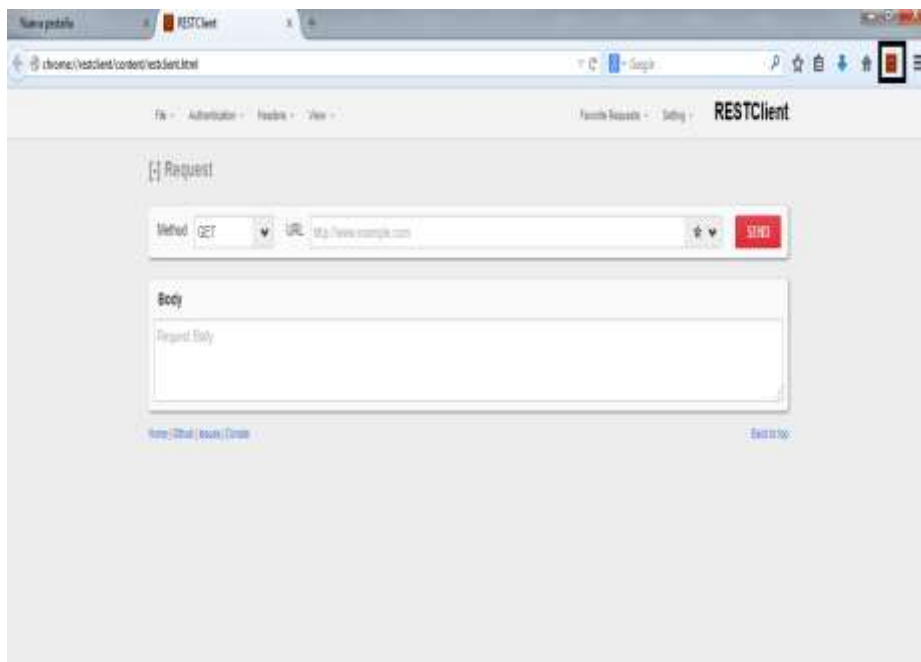

E-Crear Cliente REST

Para los clientes hay dos opciones para probar los servicios REST una mediante una aplicación Java y otra mediante un complemento de un browser

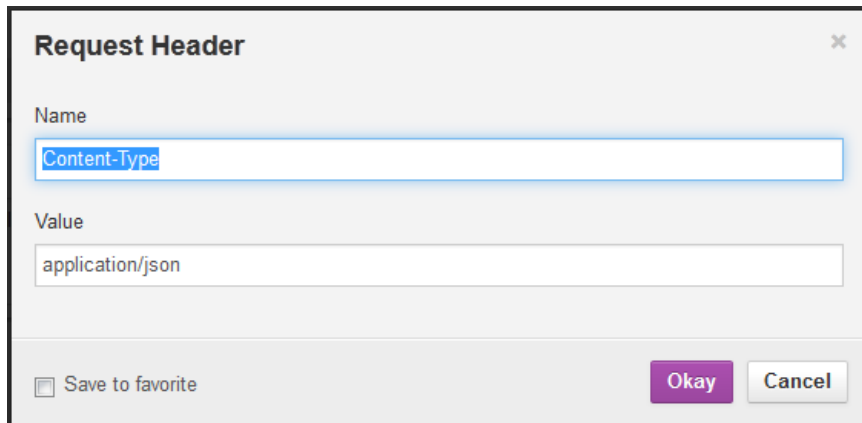
1-Mediante el Browser

Para realizar pruebas de REST utilizamos un complemento de Browser. Para este caso utilizamos el Browser FireFox.

1-Descargamos el complemento <https://addons.mozilla.org/es/firefox/addon/restclient/>



2-En la opción Headers->Custom Header agregar los datos que se muestran en la pantalla siguiente



A dialog box titled "Request Header" with a close button (X) in the top right corner. It contains two input fields: "Name" with the text "Content-Type" and "Value" with the text "application/json". At the bottom, there is a checkbox labeled "Save to favorite" and two buttons: "Okay" and "Cancel".

Name: Content-Type

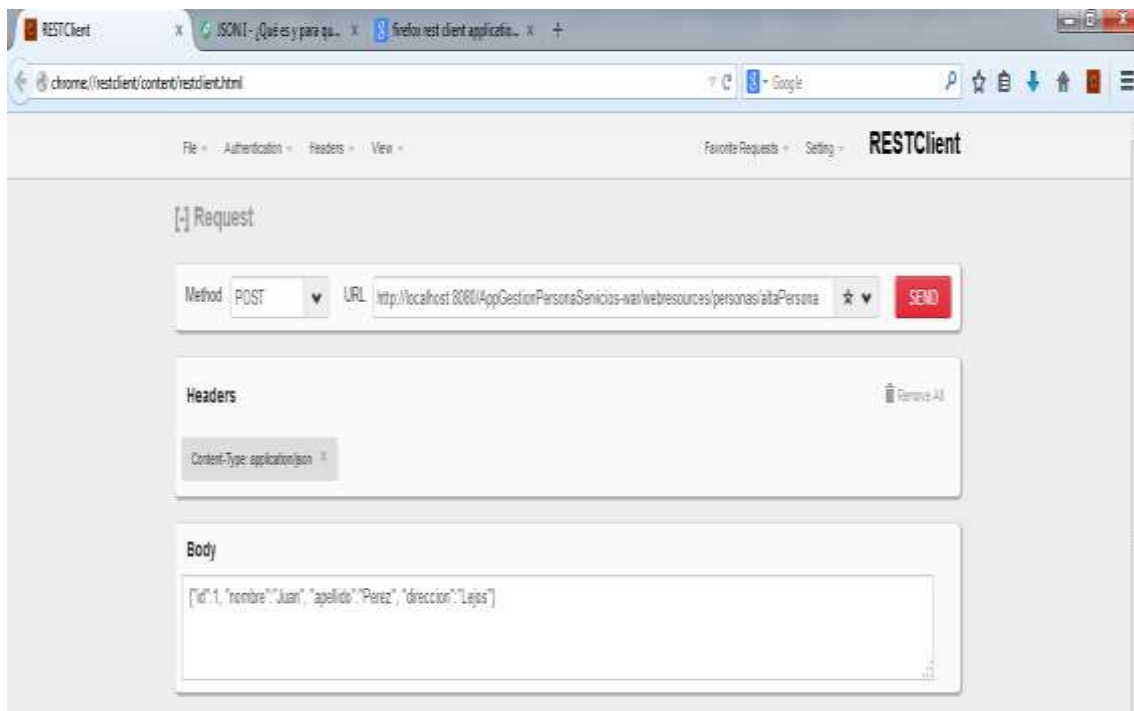
Value: application/json

3- En la pantalla principal ingresar los siguientes datos

Method: POST

URL: <http://localhost:8080/AppGestionPersonaServicios-war/webresources/personas/>

Body: {"id":1, "nombre":"Juan", "apellido":"Perez", "direccion":"Lejos"} (Formato JSON)



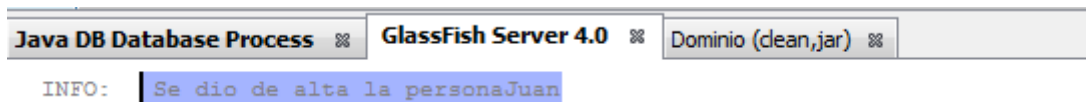
A screenshot of the RESTClient application running in a web browser. The interface shows a "Request" section with a "Method" dropdown set to "POST" and a "URL" field containing "http://localhost:8080/AppGestionPersonaServicios-war/webresources/personas/altaPersona". A red "SEND" button is next to the URL. Below the URL, there is a "Headers" section with a "Content-Type: application/json" header. At the bottom, there is a "Body" section with a text area containing the JSON string: {"id":1, "nombre":"Juan", "apellido":"Perez", "direccion":"Lejos"}. The browser's address bar shows the RESTClient URL, and the top of the browser window has several tabs open.

4-Realizamos SEND

5-Response



Verificar en el servidor si salio el mensaje



6-Operaciones

6.1-Lista

Method: GET

URL: <http://localhost:8080/AppGestionPersonaServicios-war/webresources/personas/>

6.2-Modificar

Method: PUT

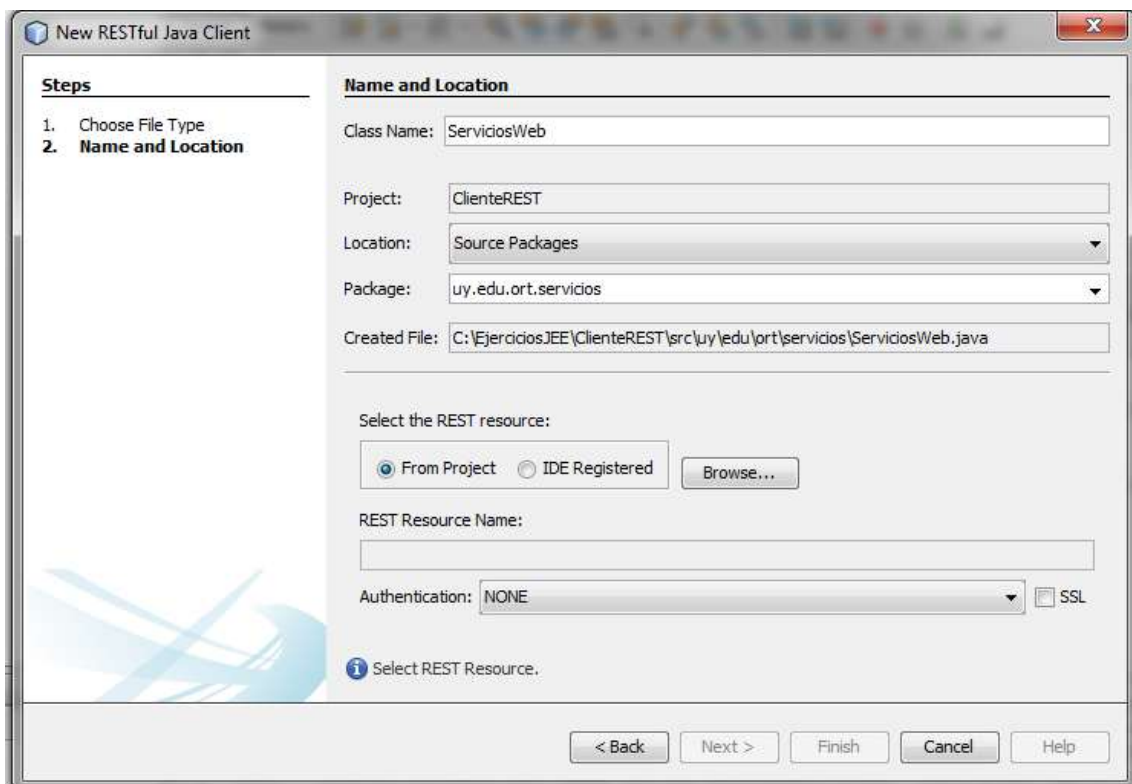
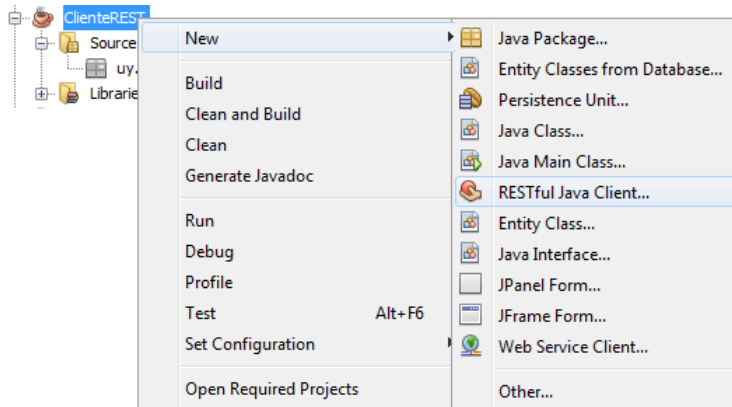
URL: <http://localhost:8080/AppGestionPersonaServicios-war/webresources/personas/1>

2-Mediante una aplicación Java

1-Crear un proyecto del tipo Java Application llamado ClienteREST en la carpeta C:\EjercicioREST\.

2-Crear paquete uy.edu.ort.cliente

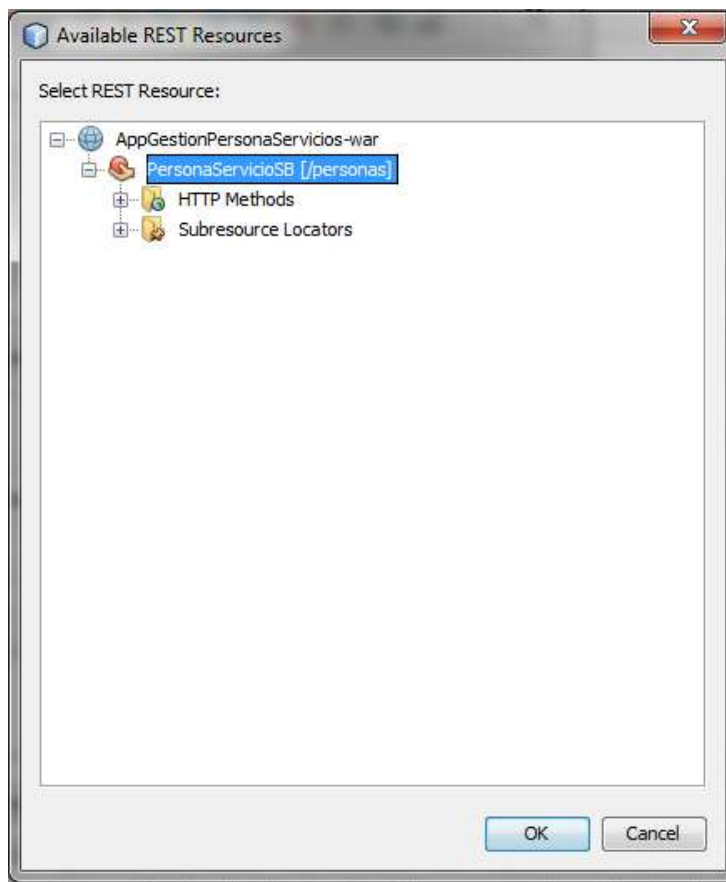
3-Clikeamos sobre el proyecto y seleccionamos RESTful Java Client



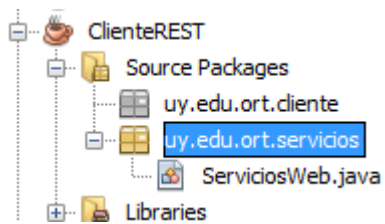
Class Name: ServicioWeb

Package: uy.edu.ort.servicios

4- Clikeamos sobre Browser y seleccionamos el proyecto Web que contiene los servicios



5-Ok y Finish



6- Agregamos el proyecto Dominio

7- Creamos un clase en el paquete uy.edu.ort.cliente con el nombre ClienteREST

7.1- Código de ClienteREST

```
public class ClienteREST {  
    public static void main(String[] args) {  
        ServicioWeb serviciosWeb = new ServicioWeb();  
        PersonaDTO persona = new PersonaDTO();  
        persona.setNombre("Juan");  
        persona.setApellido("Mart");  
        persona.setDireccion("Lejos");  
        persona.setId(1);  
        serviciosWeb.alta(persona);  
    }  
}
```

8-Modificamos el constructor de la clase ServicioWeb

```
public ServicioWeb () {  
    ClientConfig cc = new ClientConfig().register(new JacksonFeature());  
    client = javax.ws.rs.client.ClientBuilder.newClient(cc);  
    webTarget = client.target(BASE_URI).path("personas");  
}
```