

SERVICIOS WEB RESTFULL CON WILDFLY - RESTEASY

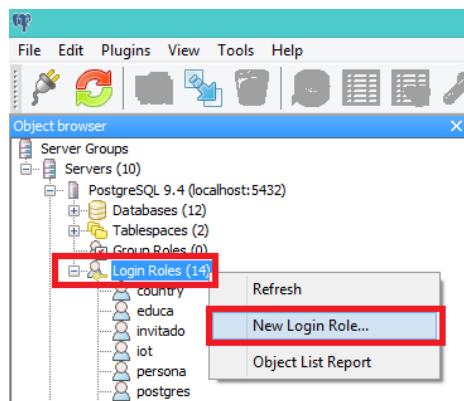
Profesor: Ing. Juan Antonio Castro Silva

Version: 1.1 (10-MAY-2016-10:31)

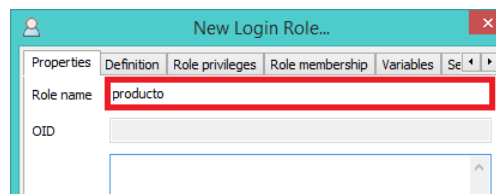
Este proyecto comprende cuatro grandes partes la creación de la base de datos, la aplicación web, las pruebas de los servicios web y la creación del cliente.

1. CREACION DE LA BASE DE DATOS

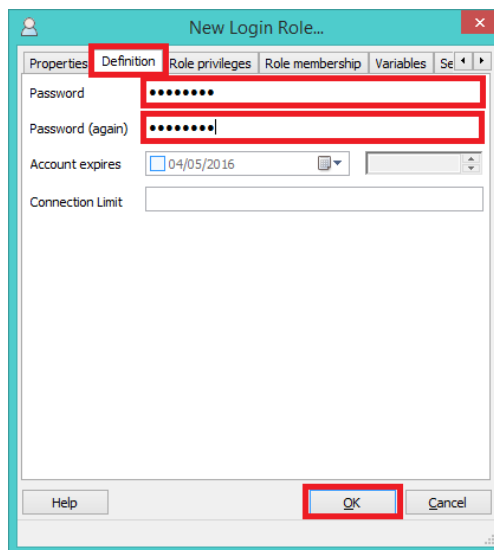
Cree un nuevo login role que tenga como nombre producto y clave producto. Para crear un nuevo login role haga click con el boton derecho del mouse en el menú [Login Roles] y seleccione [New Login Role...].



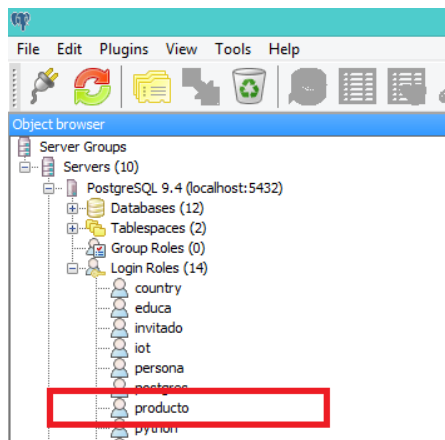
Digite el nombre del role [Role name], para este caso producto.



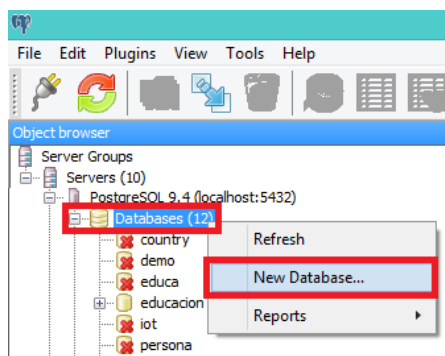
Seleccione la ficha [Definition], digite el password (producto) y confírmelo (producto), finalmente haga click en el boton [OK].



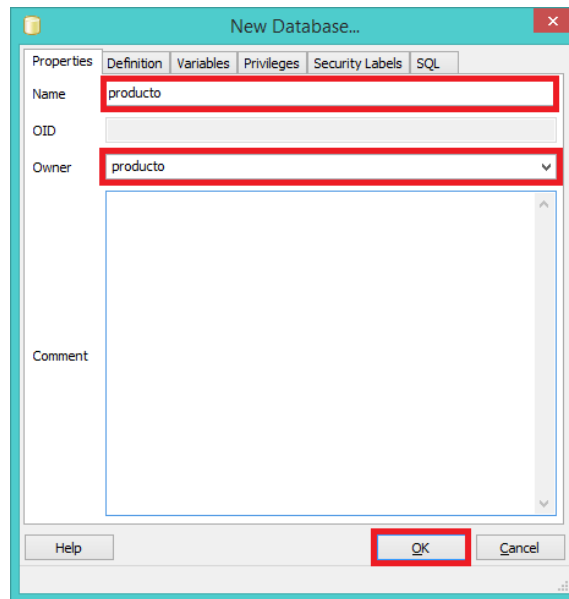
Como resultado debe aparecer creado el nuevo login role (producto):



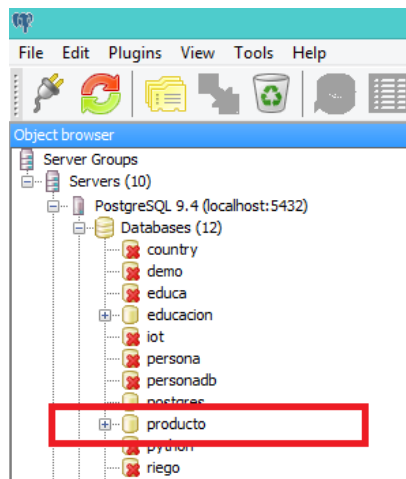
Para crear la base de datos, haga click con el boton derecho del mouse en el menú [Databases] y seleccione la opción [New Database].



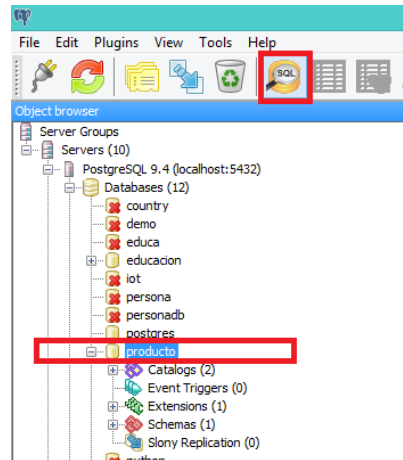
En la ventana de [New Database...], digite el nombre de la base de datos [Name] y seleccione el propietario [Owner], finalmente haga click en el boton [OK].



Como resultado debe aparecer creada la base de datos (producto):



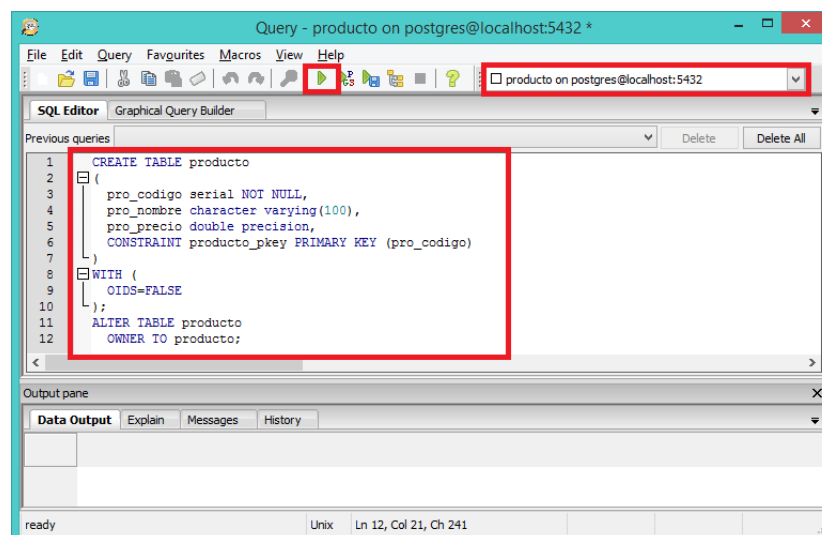
Para crear la tabla (producto), seleccione la base de datos producto y haga click en el boton [SQL] que se encuentra arriba en la barra de herramientas.



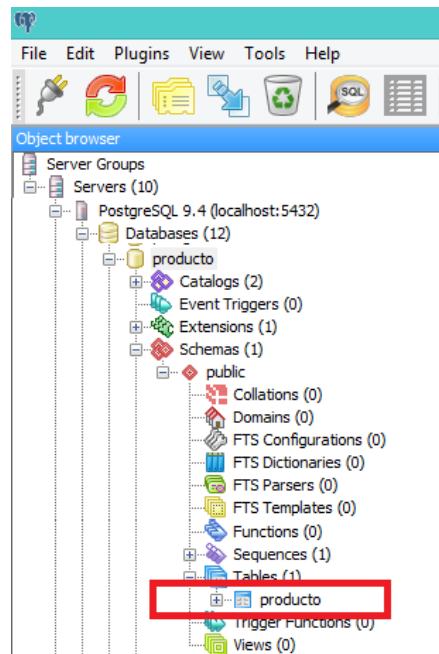
En la ventana de consulta [Query], pegue la sentencia SQL para crear la tabla producto.

```
CREATE TABLE producto
(
  pro_codigo serial NOT NULL,
  pro_nombre character varying(100),
  pro_precio double precision,
  CONSTRAINT producto_pkey PRIMARY KEY (pro_codigo)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE producto
  OWNER TO producto;
```

Haga click en el boton de [Execute query] que se encuentra arriba en la barra de herramientas, para ejecutar la sentencia SQL.



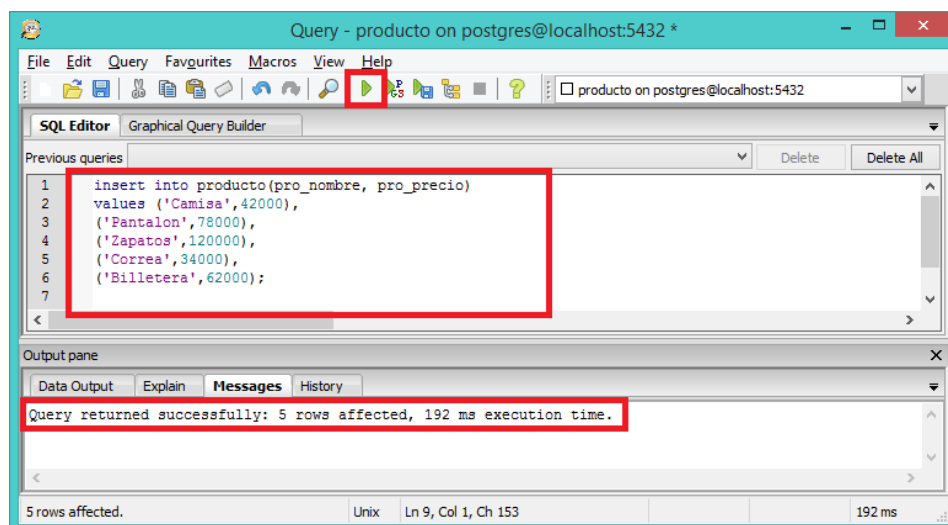
Como resultado debe haber creado la tabla producto.



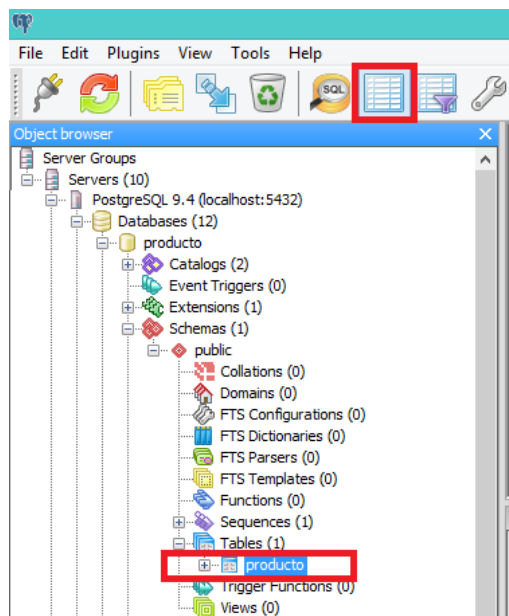
Para insertar los registros en la tabla (producto), seleccione la base de datos producto y haga click en el boton [SQL] que se encuentra arriba en la barra de herramientas. La sentencia SQL insert permite adicionar registros a la base de datos, péguela en la ventana de [Query].

```
insert into producto(pro_nombre, pro_precio)
values ('Camisa',42000),
('Pantalon',78000),
('Zapatos',120000),
('Correa',34000),
('Billetera',62000);
```

Haga click en el boton de [Execute query], para insertar los registros en la tabla.

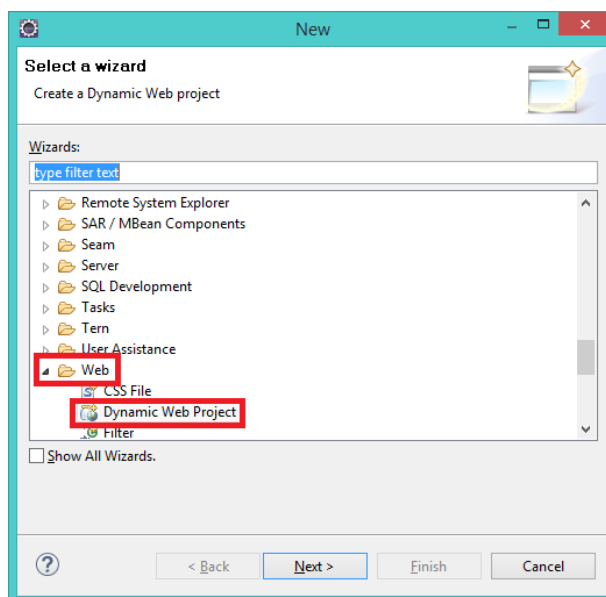


Para visualizar los datos, seleccione la tabla (producto) y haga click en el boton [Ver datos] que se encuentra arriba en la barra de herramientas.

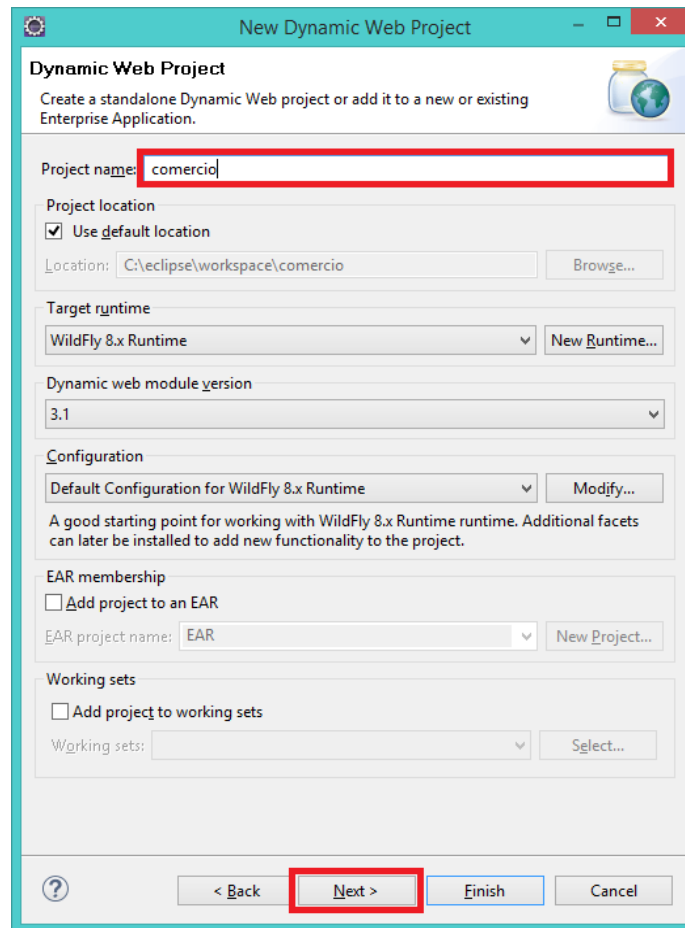


Deben aparecer los siguientes registros en la tabla producto:

	pro_codigo [PK] serial	pro_nombre character varying(100)	pro_precio double precision
1	1	Camisa	42000
2	2	Pantalon	78000
3	3	Zapatos	120000
4	4	Correa	34000
5	5	Billetera	62000
*			



En la ventana de [New Dynamic Web Project] digite el nombre del proyecto (comercio) y haga click en el boton de [Next].



New 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: [Browse...](#)

Target runtime

[New Runtime...](#)

Dynamic web module version

Configuration

[Modify...](#)

A good starting point for working with WildFly 8.x Runtime runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name: [New Project...](#)

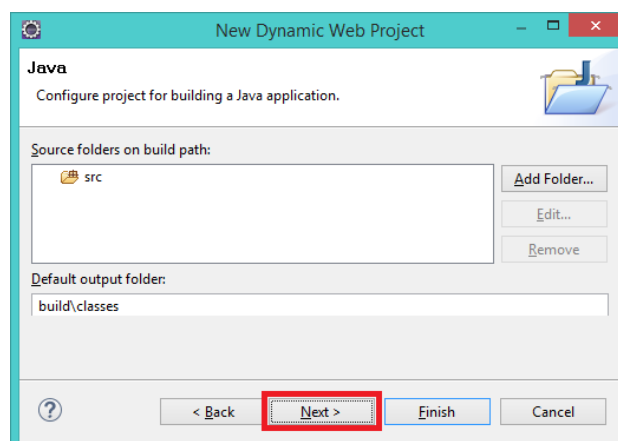
Working sets

☐ Add project to working sets

Working sets: [Select...](#)

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

Sin realizar cambios, haga click en el boton de [Next].



New Dynamic Web Project

Java

Configure project for building a Java application.

Source folders on build path:

[Add Folder...](#)

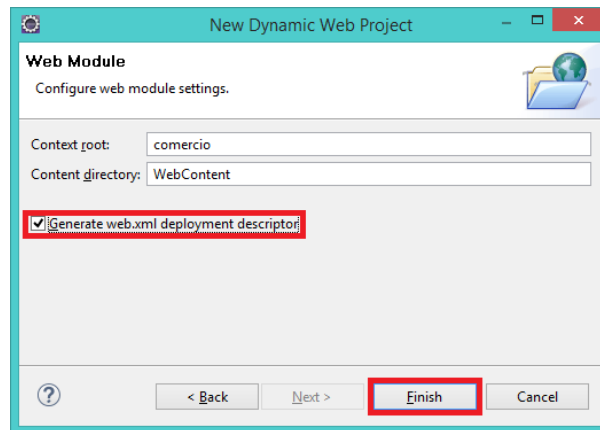
[Edit...](#)

[Remove](#)

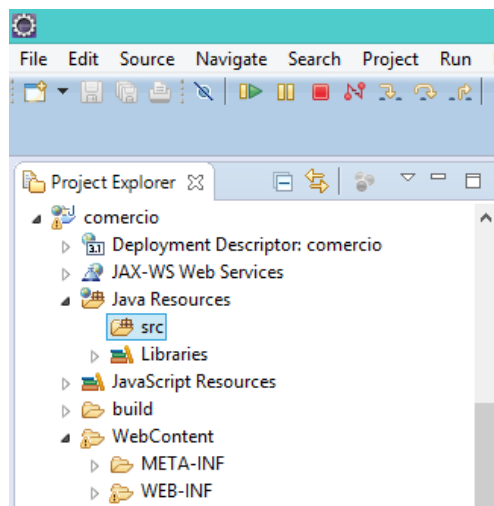
Default output folder:

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

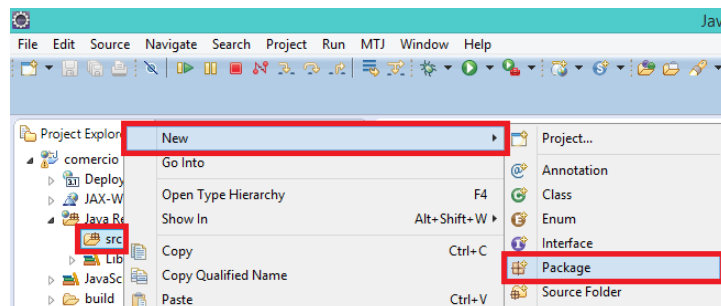
Seleccione el cajón de chequeo [Generate web.xml deployment descriptor] y haga click en el boton de [Finish]



Como resultado se ha creado el proyecto (comercio) en eclipse.



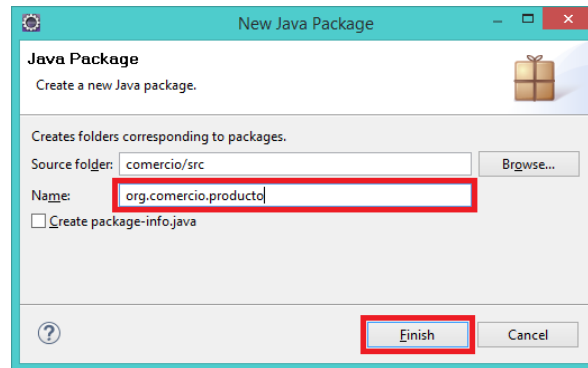
Para crear un paquete haga click en la carpeta [src], [New] y finalmente en [Package].



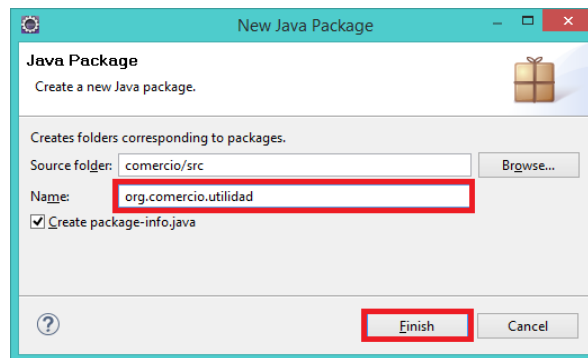
Para este proyecto se deben crear dos paquetes:

- org.comercio.producto
- org.comercio.utilidad

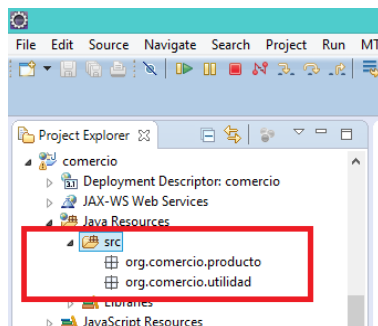
Digite el nombre del paquete en el cajón de texto de nombre [Name] (org.comercio.producto) y haga click en el boton de [Finish]



Repita el procedimiento para el otro paquete (org.comercio.utilidad).



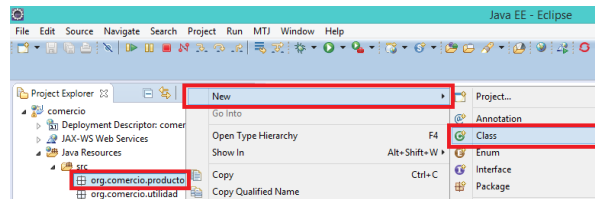
Al final debe tener los dos paquetes creados dentro de la carpeta [src].



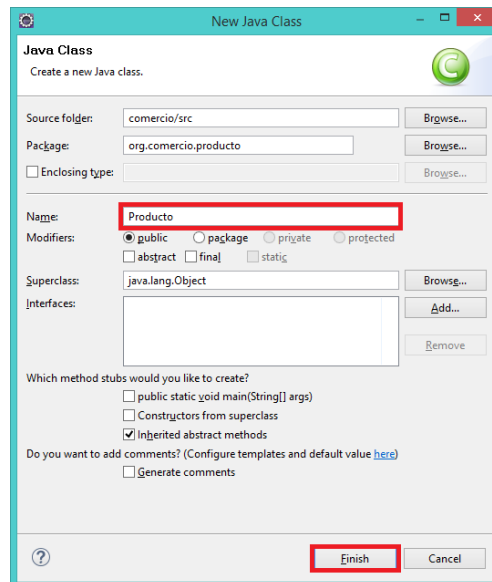
Dentro del paquete `org.comercio.producto` debe crear tres clases:

- Producto
- ProductoLista
- ProductoServicio

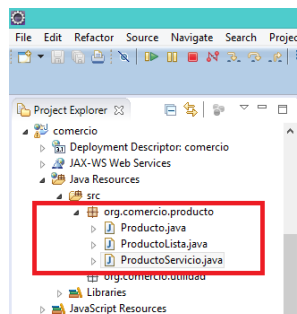
Para crear una clase haga click con el boton derecho del mouse en el paquete `[org.comercio.producto]`, seleccione `[New]` y finalmente haga click en `[Class]`.



En la ventana de `[New Java Class]` digite el nombre `[Name]` de la clase (`Producto`) y haga click en el boton `[Finish]`.



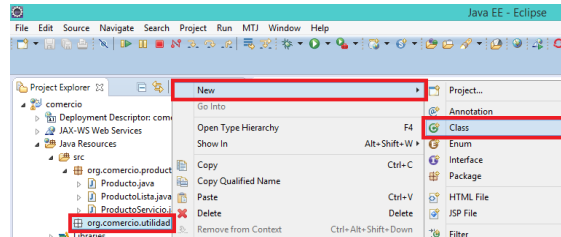
Repita esta operación con las otras dos clases (`ProductoLista` y `ProductoServicio`), al finalizar debe tener creadas las tres clases.



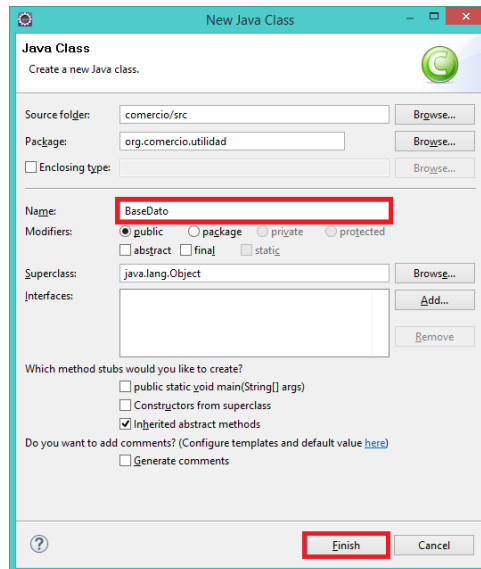
Dentro del paquete (org.comercio.utilidad) debe crear dos clases:

- BaseDato
- WebConfig

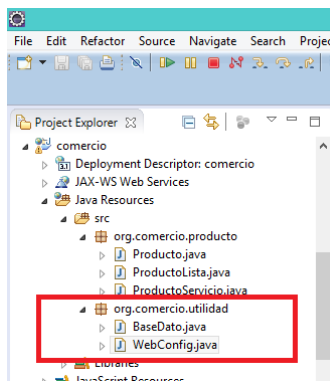
Para crear las clases haga click en el paquete [org.comercio.utilidad], [New] y finalmente en [Class].



En la ventana de [New Java Class] digite el nombre [Name] de la clase (BaseDato) y haga click en el boton de [Finish].



Al finalizar debe tener las dos clases (BaseDato y WebConfig) creadas dentro del paquete [org.comercio.utilidad].



En la clase WebConfig digite el siguiente contenido.

```
package org.comercio.utilidad;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("/servicios")
public class WebConfig extends Application {
    // No methods defined inside
}
```

El contenido de la clase BaseDato es el siguiente:

```
package org.comercio.utilidad;

import java.sql.Connection;
import java.sql.DriverManager;

public class BaseDato {

    public Connection getConexion(){
        Connection conexion1 = null;

        String driver = "org.postgresql.Driver";
        String url = "jdbc:postgresql://localhost:5432/producto";
        String user = "producto";
        String password = "producto";

        try{
            Class.forName(driver);
            conexion1 = DriverManager.getConnection(url, user, password);
        }
        catch(Exception e){
            System.out.println("Error: " + e.toString());
        }

        return conexion1;
    }
}
```

El contenido de la clase Producto es el siguiente:

```
package org.comercio.producto;

public class Producto {
    int codigo;
    String nombre;
    double precio;

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
}
```

```

    public String getNombre() {
        return nombre;
    }

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

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }
}

```

El contenido de la clase ProductoLista es el siguiente:

```

package org.comercio.producto;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="listing")
public class ProductoLista {
    private List<Producto> items;

    public ProductoLista(){
    }

    public ProductoLista(List<Producto> items){
        this.items = items;
    }

    @XmlElement(name="data")
    public List<Producto> getItems(){
        return items;
    }
}

```

El contenido de la clase ProductoServicio es el siguiente:

```
package org.comercio.producto;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.DELETE;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;

import org.comercio.utilidad.BaseDato;

@Path("/producto")
public class ProductoServicio {

    @POST
    @Path("/adicionar")
    @Consumes({"application/json"})
    @Produces("application/json")
    public Response adicionar(Producto producto) {
        BaseDato basedato = new BaseDato();
        Connection conexion1 = null;
        PreparedStatement sentenciaPreparada1 = null;
        String sql = "";
        String mensaje = "";
        int insertados = 0;

        try{
            conexion1 = basedato.getConexion();

            sql = "INSERT INTO producto (pro_nombre, pro_precio)";
            sql = sql + " VALUES (?,?)";

            sentenciaPreparada1 = conexion1.prepareStatement(sql);
            sentenciaPreparada1.setString(1, producto.getNombre());
            sentenciaPreparada1.setDouble(2, producto.getPrecio());
            insertados = sentenciaPreparada1.executeUpdate();
        }
        catch(Exception e){
            System.out.println("Error: " + e.toString());
        }

        if(insertados > 0){
            mensaje = "{\"mensaje\":\"Adicionar OK\"}";
            return Response.status(200).entity(mensaje).build();
        }
        else{
            mensaje = "{\"mensaje\":\"Error al adicionar\"}";
            return Response.status(400).entity(mensaje).build();
        }
    }
}
```

```

@PUT
@Path("/modificar/{codigo}")
@Consumes({"application/json"})
@Produces("application/json")
public Response modificar(Producto producto,
    @PathParam(value = "codigo")int codigo) {
    BaseDato basedato = new BaseDato();
    Connection conexion1 = null;
    PreparedStatement sentenciaPreparada1 = null;
    String sql = "";
    String mensaje = "";
    int modificados = 0;

    try{
        conexion1 = basedato.getConexion();

        sql = "UPDATE producto set pro_nombre=?, pro_precio=?";
        sql = sql + " WHERE pro_codigo=?";

        sentenciaPreparada1 = conexion1.prepareStatement(sql);
        sentenciaPreparada1.setString(1, producto.getNombre());
        sentenciaPreparada1.setDouble(2, producto.getPrecio());
        sentenciaPreparada1.setIn(3, codigo);
        modificados = sentenciaPreparada1.executeUpdate();
    }
    catch(Exception e){
        System.out.println("Error: " + e.toString());
    }

    //return Response.ok().entity(mensaje).build();
    if(modificados > 0){
        mensaje = "{\"mensaje\":\"Modificar OK\"}";
        return Response.status(200).entity(mensaje).build();
    }
    else{
        mensaje = "{\"mensaje\":\"Error al modificar\"}";
        return Response.status(400).entity(mensaje).build();
    }
}

@DELETE
@Path("/eliminar/{codigo}")
@Consumes({"application/json"})
@Produces("application/json")
public Response adicionar(@PathParam(value = "codigo") int codigo) {
    BaseDato basedato = new BaseDato();
    Connection conexion1 = null;
    PreparedStatement sentenciaPreparada1 = null;
    String sql = "";
    String mensaje = "";
    int eliminados = 0;

    try{
        conexion1 = basedato.getConexion();

        sql = "DELETE FROM producto WHERE pro_codigo=?";

        sentenciaPreparada1 = conexion1.prepareStatement(sql);
        sentenciaPreparada1.setInt(1, codigo);
        eliminados = sentenciaPreparada1.executeUpdate();
    }
}

```



```

        catch(Exception e){
            System.out.println("Error: " + e.toString());
        }

        if(eliminados > 0){
            mensaje = "{\"mensaje\":\"Eliminar OK\"}";
            return Response.status(200).entity(mensaje).build();
        }
        else{
            mensaje = "{\"mensaje\":\"Error al eliminar\"}";
            return Response.status(400).entity(mensaje).build();
        }
    }

    @GET
    @Path("/listar")
    @Produces("application/json")
    // @Produces("application/xml")
    public ProductoLista getProductos() {
        ArrayList lista = new ArrayList();

        BaseDato basedato = new BaseDato();
        Connection conexion1 = null;
        Statement sentencial = null;
        ResultSet rs1 = null;
        String sql = "";

        try{
            conexion1 = basedato.getConexion();
            sentencial = conexion1.createStatement();

            sql = "select * from producto";

            rs1 = sentencial.executeQuery(sql);

            while(rs1.next()){
                int codigo = rs1.getInt("pro_codigo");
                String nombre = rs1.getString("pro_nombre");
                double precio = rs1.getDouble("pro_precio");

                Producto producto = new Producto();

                producto.setCodigo(codigo);
                producto.setNombre(nombre);
                producto.setPrecio(precio);

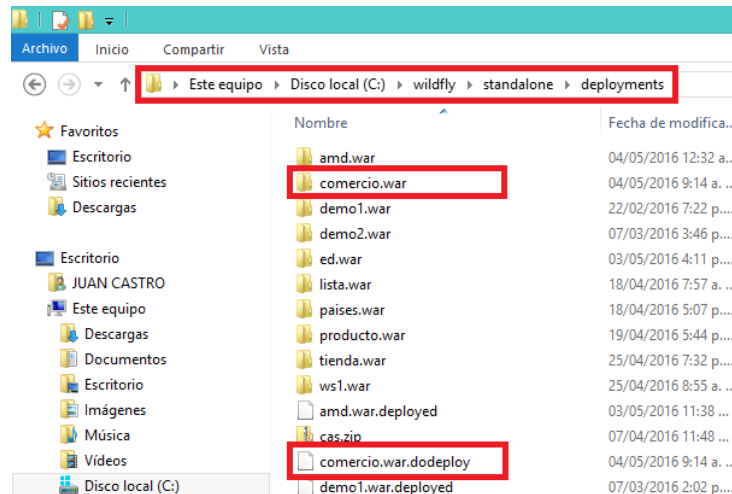
                lista.add(producto);
            }
        }
        catch(Exception e){
            System.out.println("Error: " + e.toString());
        }

        return new ProductoLista(lista);
    }
}

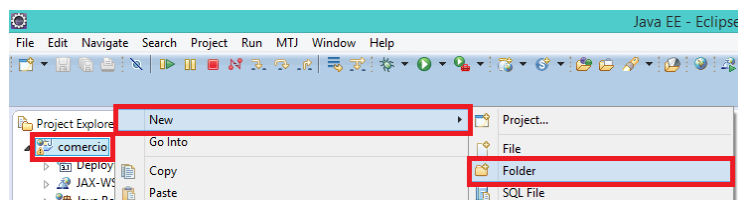
```

Configurar la aplicación Web para programar desde eclipse.

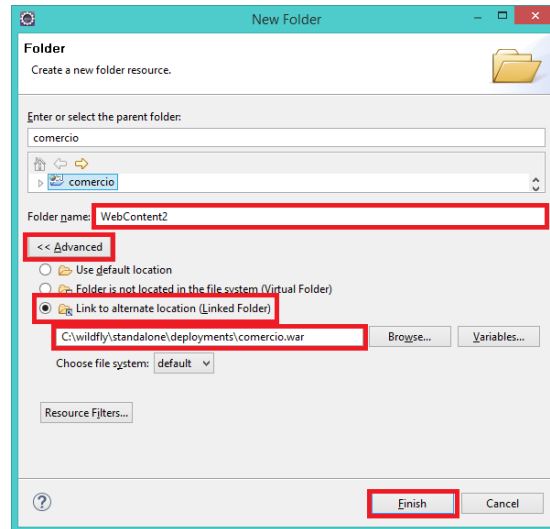
Abra el explorador de archivos y dentro de la carpeta de publicaciones del wildfly (wildfly/standalone/deployments/), debe crear un folder (carpeta) para almacenar los contenidos de la aplicación web (comercio.war) y el archivo de publicación (comercio.war.dodeploy) para que el servidor de aplicaciones (wildfly) la publique.



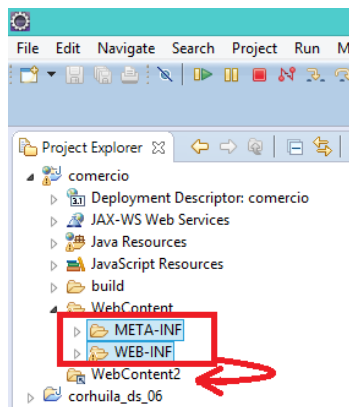
Para configurar la aplicación en eclipse, haga click con el boton derecho del mouse sobre el proyecto [comercio], seleccione la opción [New] y finalmente haga click en [Folder].



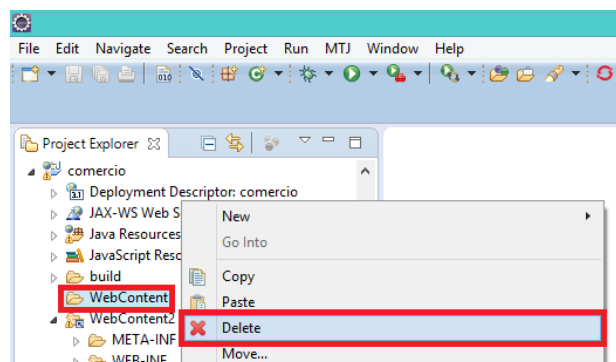
En la ventana de [New folder] digite el nombre del folder [Folder name] (WebContent2), seleccione el boton [Advanced], escoja la opción [Link to alternate location (Linked Folder)], seleccione la ruta donde está la carpeta donde se alojará la aplicación web (wildfly\standalone\deployments\comercio.war) y finalmente haga click en el boton [Finish].



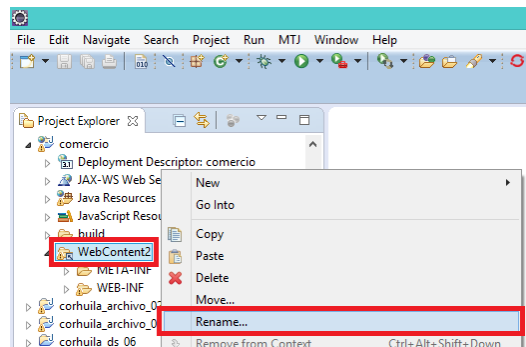
Mueva todo el contenido (archivos y folders) que están dentro de la carpeta WebContent al nuevo folder WebContent2.



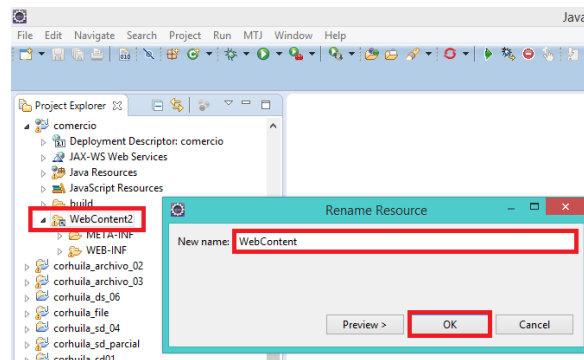
Verifique que la carpeta WebContent este vacía y que su contenido este ahora en WebContent2. Haga click con el boton derecho del mouse sobre el folder WebContent y seleccione la opción de borrar [Delete].



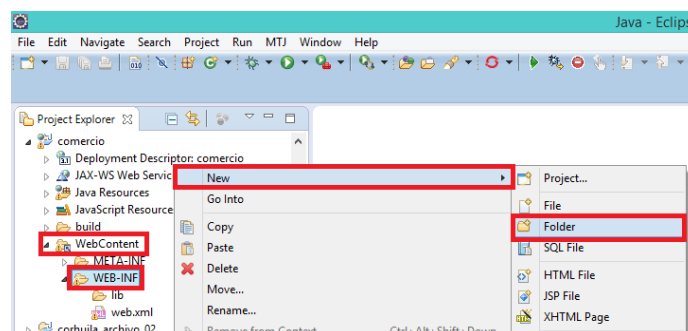
Renombre la carpeta WebContent2 como WebContent, haga click con el boton derecho del mouse sobre el folder WebContent2, seleccione la opción renombrar [Rename].



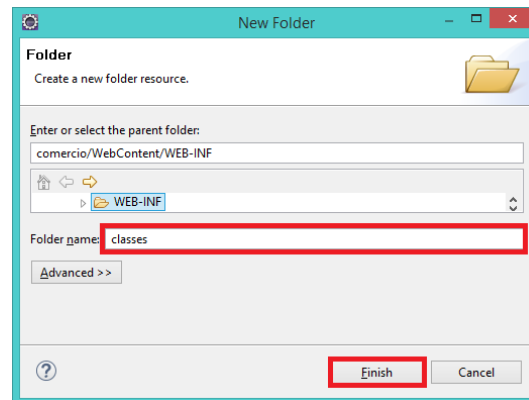
Digite el nuevo nombre [New name] (WebContent) y haga click en el boton [OK].



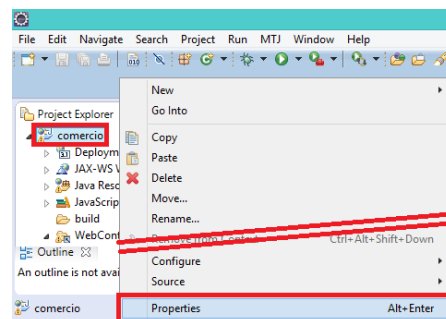
Configure eclipse para que al compilar las clases (archivos .java) grabe los resultados (archivos .class) en la carpeta de publicaciones (WEB-INF/classes). Para crear la carpeta de salida (WEB-INF/classes), haga click con el boton derecho del mouse sobre la carpeta WEB-INF que está dentro de (WebContent), seleccione [New] y finalmente haga click en [Folder].



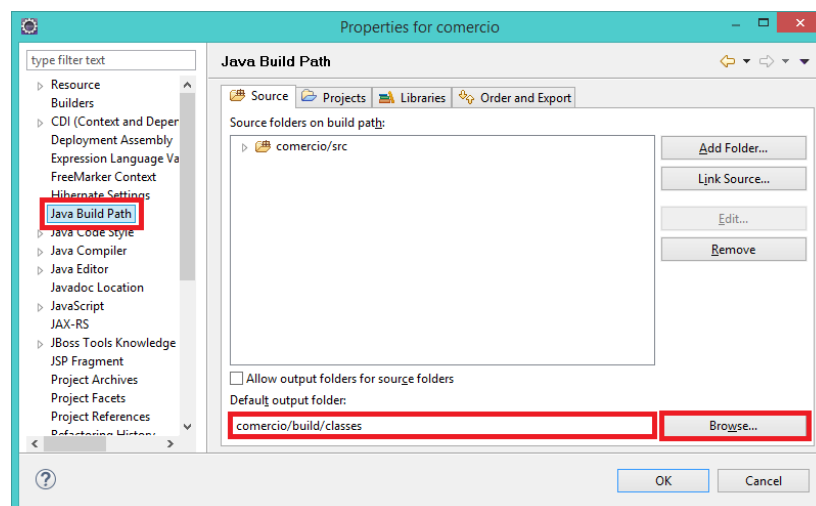
En la ventana [New Folder] digite el nombre de la nueva carpeta (classes) y haga click en el boton [Finish].



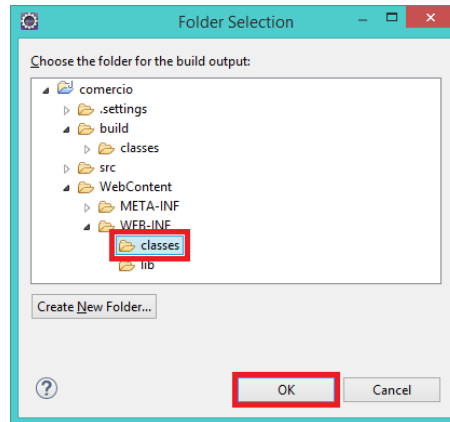
Para cambiar la carpeta de salida (donde se graban los archivos compilados .class), haga click con el boton derecho del mouse sobre el proyecto (comercio) y seleccione la opción propiedades [Properties].



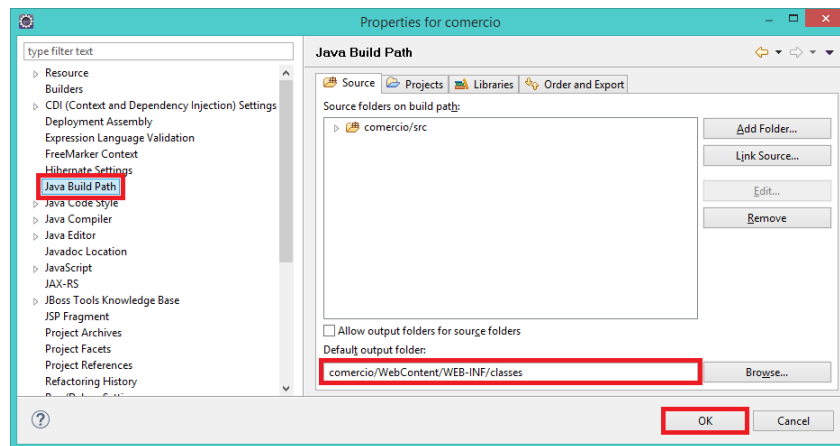
En la ventana de propiedades se puede apreciar que la carpeta de salida [output folder] está apuntando al folder build de eclipse y se debe cambiar a la carpeta de salida del wildfly, haga click en el boton [Browse...].



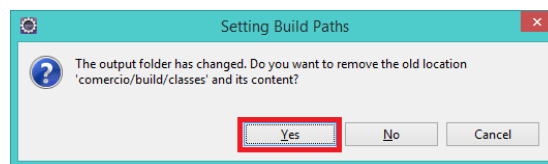
En la ventana de selección del folder [Folder Selection], elija la carpeta classes que está dentro del folder WEB-INF y haga click en el boton [OK].



Verifique que la carpeta de salida ha cambiado a WebContent/WEB-INF/classes y haga click en el boton [OK].

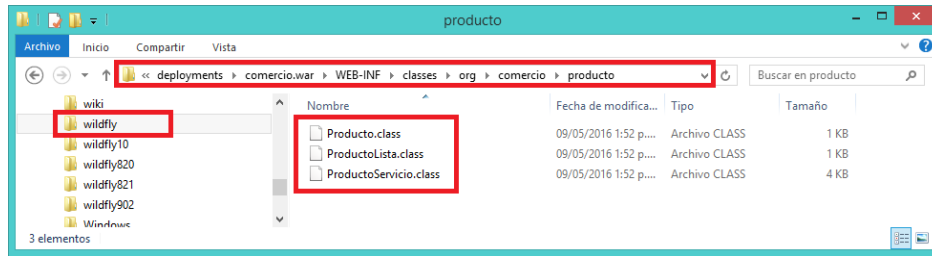


Para confirmar que desea hacer el cambio de la carpeta de salida y mover los contenidos, haga click en el boton [Yes].

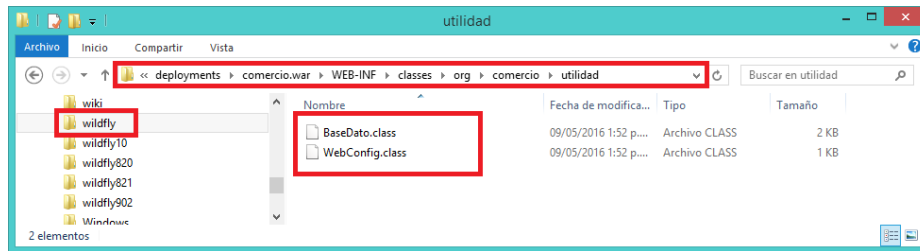


Para verificar que los archivos compilados (.class) estén en la carpeta de salida, abra el explorador de archivos y navegue hasta las carpetas que representan los dos paquetes creados (org.comercio.producto).

(/wildfly/standalone/deployments/comercio.war/WEB-INF/classes /org/comercio/producto).

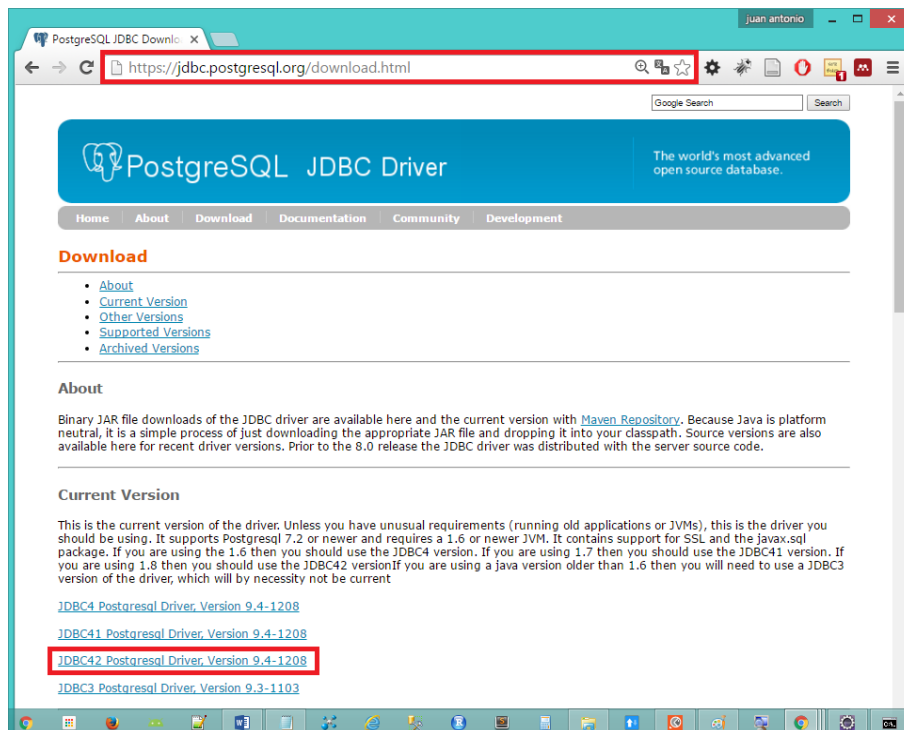


(/wildfly/standalone/deployments/comercio.war/WEB-INF/classes/org/comercio/utilidad).



Instalación del Driver JDBC

Para acceder a la base de datos (postgresql) debe instalar el driver JDBC en la carpeta lib que está dentro del folder comercio.war/WEB-INF/, el driver lo puede descargar del sitio web (<https://jdbc.postgresql.org/download.html>). En esta página web se advierte que si se utiliza JDK 1.7 se debe usar el Driver versión JDBC41, y se usa el JDK 1.8 se debe utilizar el Driver versión JDBC42.

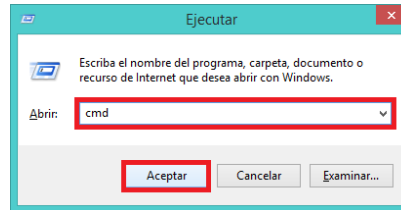


Copie el Driver JDBC en la carpeta comercio.war/WEB-INF/lib.

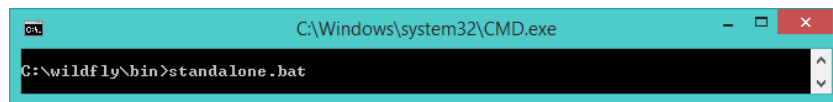


3. PRUEBA DE LOS SERVICIOS WEB

Abra una ventana de comandos (cmd para Windows o un terminal para Linux).



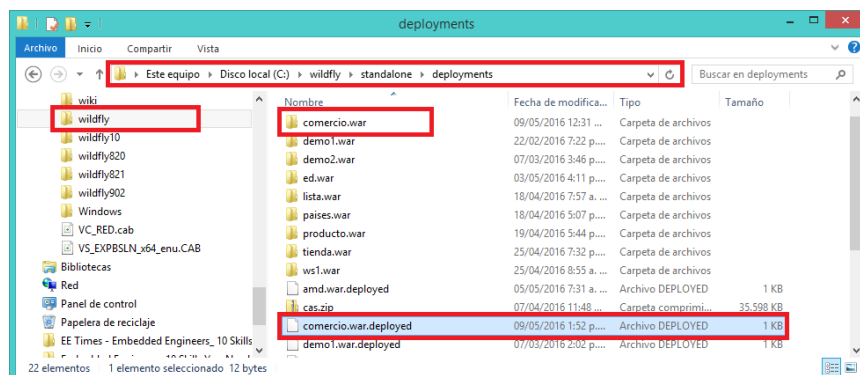
Para ejecutar (poner a correr) el servidor de aplicaciones, ingrese a la carpeta donde está instalado el wildfly, luego al folder bin donde están los archivos ejecutables (wildfly/bin) y ejecute el comando standalone.bat (./standalone.sh para Linux).



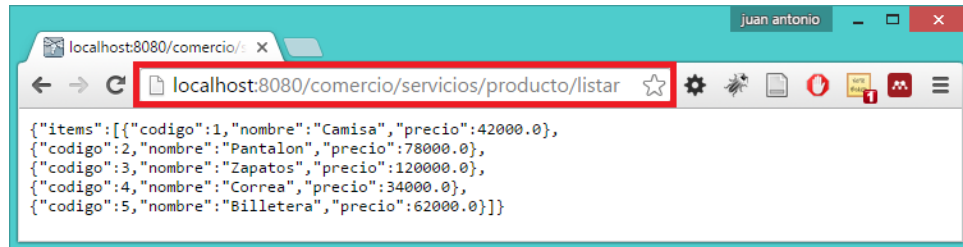
Para probar que el servidor de aplicaciones está corriendo, abra un navegador y digite la dirección (url) del servidor (http://localhost:8080/).



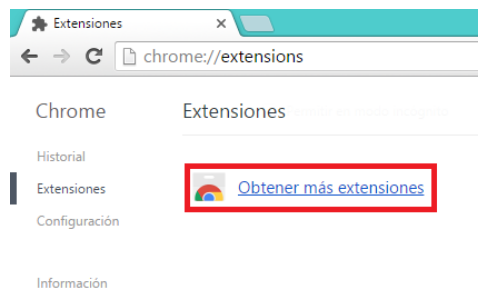
Para probar que la aplicación web esta publicada abra el explorador de archivos y navegue hasta la carpeta de publicaciones (/wildfly/standalone/deployments) y el archivo de publicación de la aplicación debe haber cambiado la extensión de dodeploy a deployed (comercio.war.deployed).



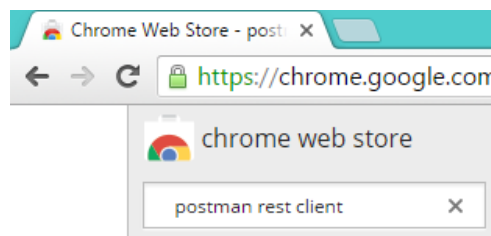
En el navegador digite la url del servicio web que permite listar todos los registros de la base de datos a través del método HTTP GET (<http://localhost:8080/comercio/servicios/producto/listar>).



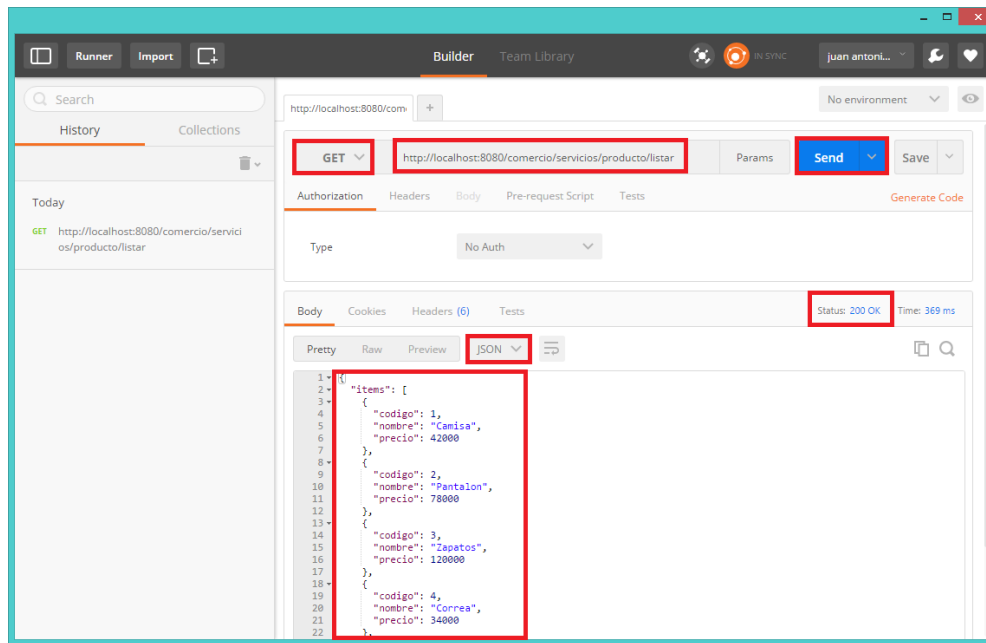
La prueba de los servicios web la podemos hacer de diversas formas, para este proyecto se utilizará la extensión de Chrome llamada Postman. Para instalar la extensión, abra el administrador de extensiones (<chrome://extensions>) y haga click en el enlace [Obtener más extensiones].



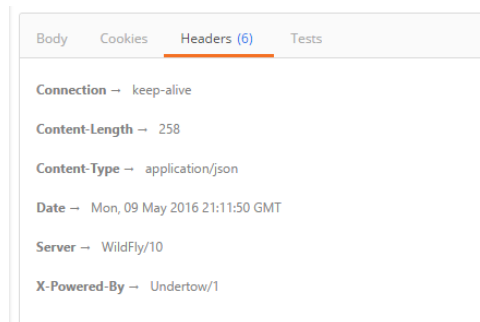
Haga la búsqueda de la extensión (postman rest client) y siga las instrucciones de instalación.



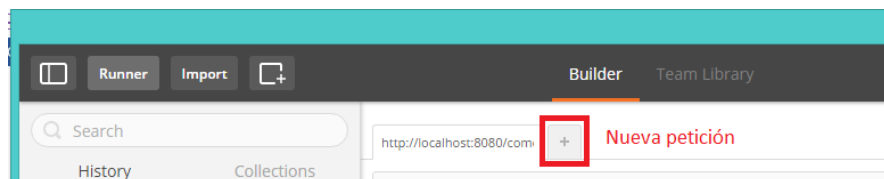
Abra el Postman y pruebe los servicios web, seleccione el método HTTP (GET), digite la url (<http://localhost:8080/comercio/servicios/producto/listar>) y finalmente haga click en el boton [Send]. En la sección de resultados se puede observar el estado de la petición (Status 200 OK), el formato de salida (JSON) y los datos enviados por el servidor de aplicaciones.



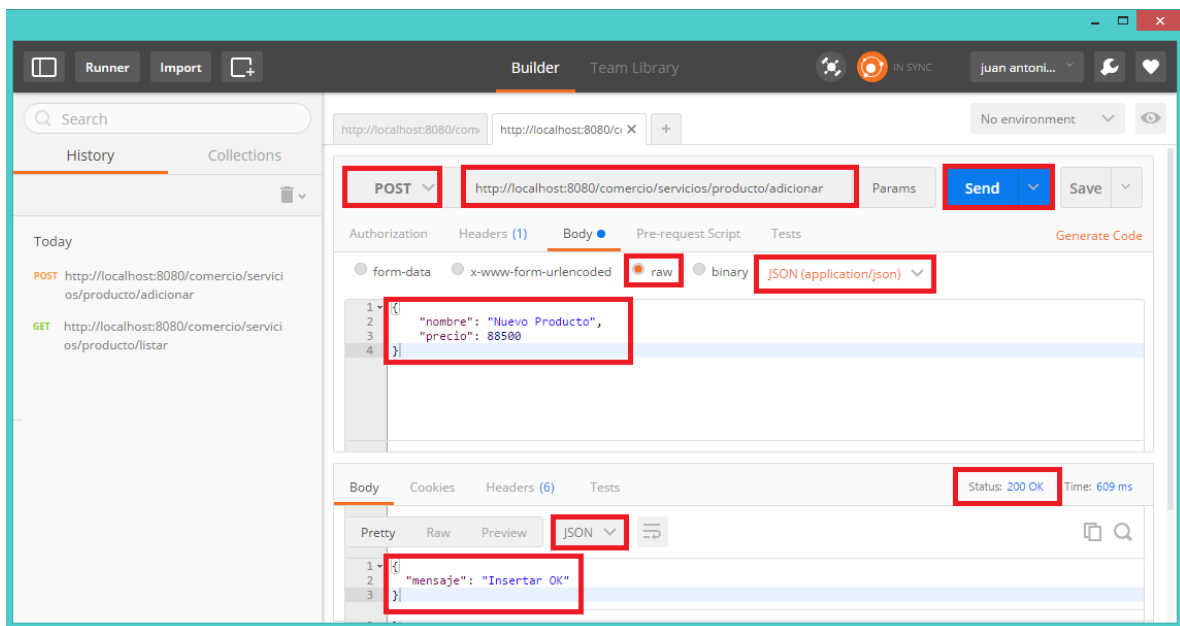
Para ver los encabezados de la respuesta [Headers] haga click en la ficha [Headers] en la sección de respuesta, aquí podemos ver la longitud del contenido (Content-Length → 258), el tipo de contenido (Content-Type → application/json), el nombre del servidor (Server → WildFly/10), etc.



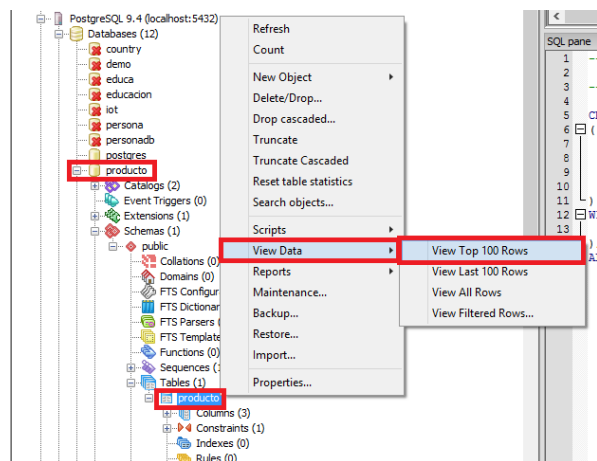
Para crear otra petición haga click en el boton de nueva petición [+].



Pruebe el servicio web para adicionar un producto, digite la url de la nueva petición (<http://localhost:8080/comercio/servicios/producto/adicionar>), seleccione el método HTTP (POST), haga click en la ficha Body, en la opción [raw] y seleccione el tipo de contenido [JSON (application/json)], digite el contenido a enviar, es decir el objeto JSON `{"nombre":"Nuevo Producto","precio":88500}` y finalmente haga click en el boton [Send]. Observe la respuesta del servidor [Status 200 OK], tipo de contenido [JSON] y el contenido de la respuesta `{"mensaje":"Insertar OK"}`.



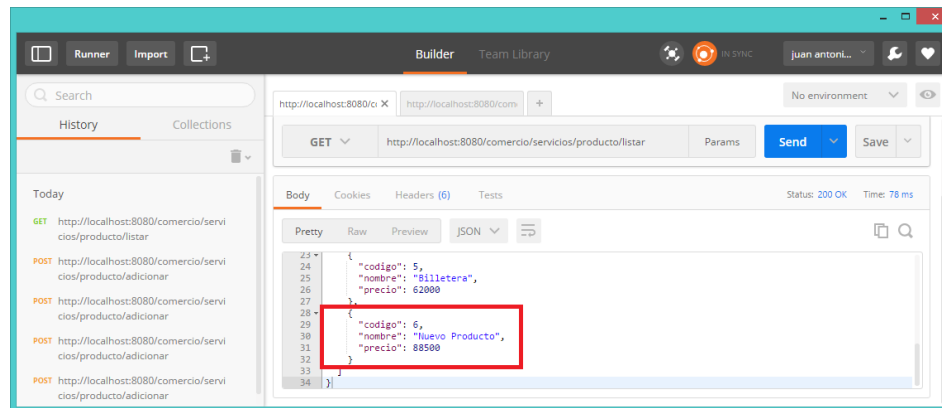
Para comprobar que se está grabando en la base de datos, en el administrador de la base de datos (pgAdmin3), haga click con el boton derecho del mouse en la tabla producto, seleccione el menú Ver datos [View Data] y la opción Ver las primeras 100 filas [View Top 100 Rows].



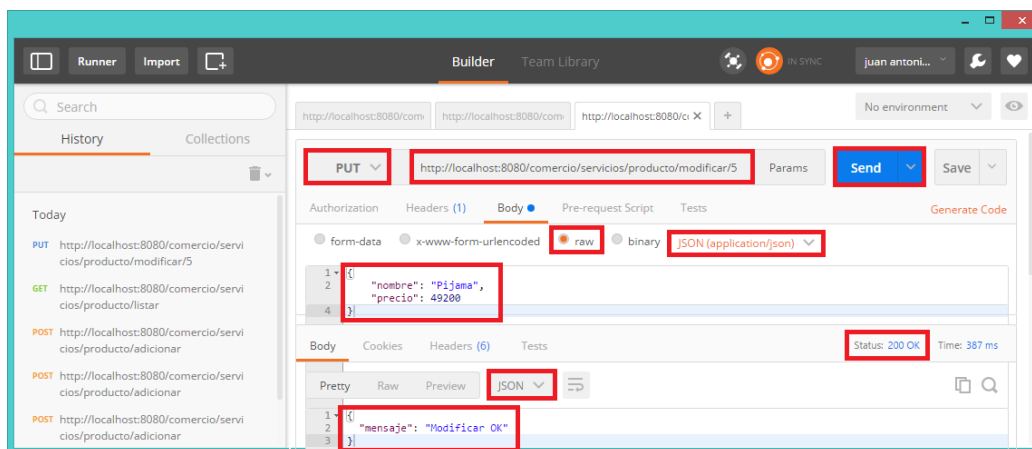
En la ventana de datos se puede ver el nuevo registro insertado.

	pro_codigo [PK] serial	pro_nombre character varying(100)	pro_precio double precision
1	1	Camisa	42000
2	2	Pantalon	78000
3	3	Zapatos	120000
4	4	Correa	34000
5	5	Billetera	62000
6	6	Nuevo Producto	88500
*			

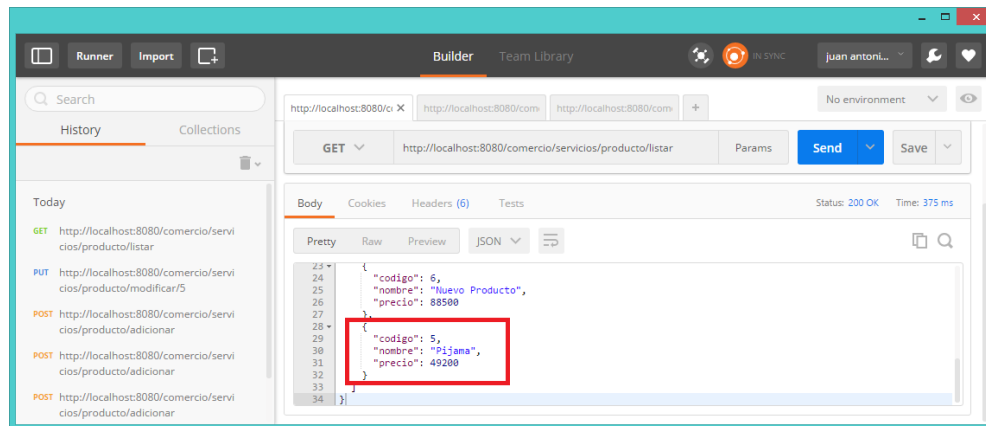
Otra forma de comprobar los resultados de las operaciones es volver a listar los datos, llamando de nuevo al servicio web de producto/listar.



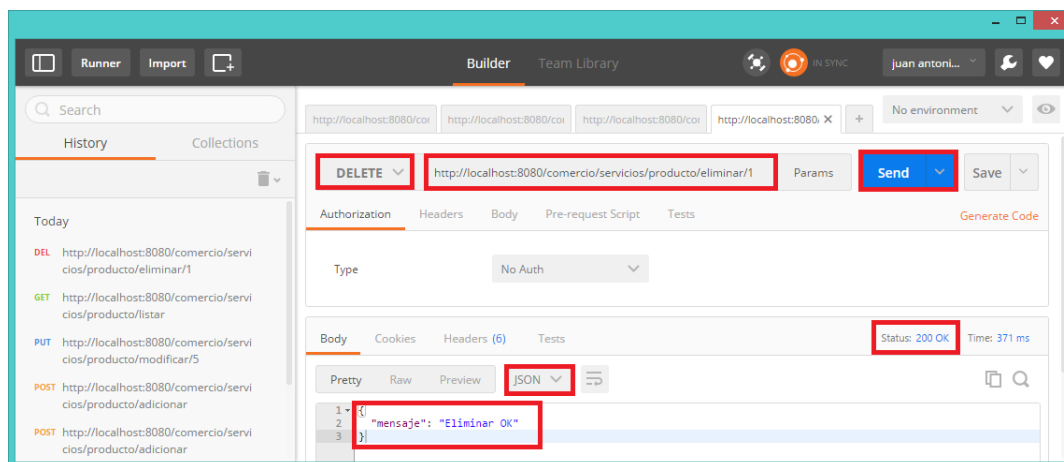
Pruebe el servicio web para modificar un producto, digite la url de la nueva petición (<http://localhost:8080/comercio/servicios/producto/modificar/5>), el 5 corresponde al código del producto que se desea modificar, seleccione el método HTTP (PUT), haga click en la ficha Body, en la opción [raw] y seleccione el tipo de contenido [JSON (application/json)], digite el contenido a enviar, es decir el objeto JSON { "nombre": "Pijama", "precio": 49200 } y finalmente haga click en el botón [Send]. Observe la respuesta del servidor [Status 200 OK], tipo de contenido [JSON] y el contenido de la respuesta { "mensaje": "Modificar OK" }.



Compruebe que el registro ha sido modificado, ejecute de nuevo el servicio web de listar (producto/listar).



Pruebe el servicio web para eliminar un producto, digite la url de la nueva petición (`http://localhost:8080/comercio/servicios/producto/eliminar/1`), el 1 corresponde al código del producto que se desea eliminar, seleccione el método HTTP (DELETE) y finalmente haga click en el botón [Send]. Observe la respuesta del servidor [Status 200 OK], tipo de contenido [JSON] y el contenido de la respuesta `{"mensaje": "Eliminar OK"}`.



4. CREACION DEL CLIENTE REST CON JAVASCRIPT

La librería RestEasy permite crear de una manera fácil el cliente rest en JavaScript para los servicios web que se han creado. Adicione el código para registrar el Servlet JSAPI en el archivo de configuración de la aplicación (WEB-INF/web.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>comercio</display-name>

  <servlet>
    <servlet-name>RETEasy-JSAPI</servlet-name>
    <servlet-class>org.jboss.resteasy.jsapi.JSAPIServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>RETEasy-JSAPI</servlet-name>
    <url-pattern>/rest-api.js</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Cree una carpeta js dentro del folder WebContent y adicione en ella un archivo JavaScript (producto.js) con el siguiente código:

```
$(function() {
  var dialog, form;

  var table = $('#example').DataTable({
    "ajax" : "servicios/producto/listar",
    "columns" : [ {
      "data" : "codigo"
    }, {
      "data" : "nombre"
    }, {
      "data" : "precio"
    } ]
  });

  $('#example tbody').on('click', 'tr', function() {
    if ($(this).hasClass('selected')) {
      $(this).removeClass('selected');
    } else {
      table.$('tr.selected').removeClass('selected');
      $(this).addClass('selected');
    }
  });
});
```

```

    }
});

function editar(accion) {
    var codigo = "0";
    var nombre = "";
    var precio = "";

    if (accion != "Adicionar") {
        codigo = table.row('.selected').data().codigo;
        nombre = table.row('.selected').data().nombre;
        precio = table.row('.selected').data().precio;
    }

    document.getElementById("accion").value = accion;
    document.getElementById("codigo").value = codigo;
    document.getElementById("nombre").value = nombre;
    document.getElementById("precio").value = precio;
    dialog.dialog("open");
}

function ejecutar() {
    var accion = document.getElementById("accion").value;

    $("#boton").html('<span class="ui-button-text">' + accion + '</span>');

    if (accion == "Adicionar") {
        adicionarProducto();
    }
    if (accion == "Modificar") {
        modificarProducto();
    }
    if (accion == "Eliminar") {
        eliminarProducto();
    }
}

function adicionarProducto() {
    // Collect input from html page
    var nombre = document.getElementById("nombre").value;
    var precio = document.getElementById("precio").value;

    var r = new REST.Request();
    r.setURI(REST.apiUrl + "/producto/adicionar");
    r.setMethod("POST");
    r.setContentType("application/json");
    r.setEntity({
        nombre : nombre,
        precio : precio
    });
    r.execute(function(status, request, entity) {
        mostrarRespuesta(entity);
    });
}

function modificarProducto() {
    // Collect input from html page
    var codigo = document.getElementById("codigo").value;
    var nombre = document.getElementById("nombre").value;
    var precio = document.getElementById("precio").value;

```



```

        var r = new REST.Request();
        r.setURI(REST.apiUrl + "/producto/modificar/" + codigo);
        r.setMethod("PUT");
        r.setContentType("application/json");
        r.setEntity({
            nombre : nombre,
            precio : precio
        });
        r.execute(function(status, request, entity) {
            mostrarRespuesta(entity);
        });
    }

function eliminarProducto() {
    // Collect input from html page
    var codigo = document.getElementById("codigo").value;

    var r = new REST.Request();
    r.setURI(REST.apiUrl + "/producto/eliminar/" + codigo);
    r.setMethod("DELETE");
    r.execute(function(status, request, entity) {
        mostrarRespuesta(entity);
    });
}

function mostrarRespuesta(entity){
    table.ajax.reload();
    dialog.dialog("close");
    document.getElementById("dialogo-mensaje").innerHTML = "<p>"
        + entity.mensaje + "</p>";
    $("#dialogo-mensaje").dialog({
        modal : true,
        buttons : {
            Ok : function() {
                $(this).dialog("close");
            }
        }
    });
}

dialog = $("#dialog-form").dialog({
    autoOpen : false,
    height : 360,
    width : 640,
    modal : true,
    buttons : {
        "Ejecutar": {
            id: 'boton',
            text: 'Ejecutar',
            click: function () {
                ejecutar();
            }
        },
        Cancel : function() {
            dialog.dialog("close");
        }
    },
    close : function() {
        form[0].reset();
    }
});

```

```

form = dialog.find("form").on("submit", function(event) {
    event.preventDefault();
    ejecutar();
});
$("#adicionar").button().on("click", function() {
    editar('Adicionar');
});
$("#modificar").button().on("click", function() {
    editar('Modificar');
});
$("#eliminar").button().on("click", function() {
    editar('Eliminar');
});
});

```

Cree una carpeta css dentro del folder WebContent y adicione en ella el archivo css (estilo.css) con el siguiente contenido.

```

label, input {
    display: block;
}

input.text {
    margin-bottom: 12px;
    width: 95%;
    padding: .4em;
}

fieldset {
    padding: 0;
    border: 0;
    margin-top: 25px;
}

h1 {
    font-size: 2.2em;
    margin: .6em 0;
}

```

Dentro de la carpeta WebContent cree el archivo index.jsp con el siguiente contenido.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<!-- This will include the whole javascript file including ajax handling -->
<script lang="javascript" src="rest-api.js"></script>
<link rel="stylesheet"
    href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
<script type="text/javascript"
    src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
<script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
<script type="text/javascript"
    src="https://cdn.datatables.net/1.10.11/js/jquery.dataTables.min.js"></script>
<link rel="stylesheet"
    href="https://cdn.datatables.net/1.10.11/css/jquery.dataTables.min.css">
<link rel="stylesheet"
    href="css/estilo.css">
<script type="text/javascript" src="js/producto.js"></script>
</head>
<body>
    <h1>GESTION DE PRODUCTOS</h1>
    <div style="float: right;">
        <input type="button" value="Adicionar" id="adicionar" />
        <input type="button" value="Modificar" id="modificar" />
        <input type="button" value="Eliminar" id="eliminar" />
    </div>

    <table id="example" class="display" cellpadding="0" width="100%">
        <thead>
            <tr>
                <th>Codigo</th>
                <th>Nombre</th>
                <th>Precio</th>
            </tr>
        </thead>
        <tfoot>
            <tr>
                <th>Codigo</th>
                <th>Nombre</th>
                <th>Precio</th>
            </tr>
        </tfoot>
    </table>

    <div id="dialogo-mensaje" title="Gestión de Productos">
    </div>

    <div id="dialog-form" title="Forma Producto">
        <p class="validateTips">Todos los campos son requeridos.</p>
        <form>
            <fieldset>
                <input type="hidden" name="accion" id="accion" value="" />
                <input type="hidden" name="codigo" id="codigo" value="" />
                <label for="nombre">Nombre</label> <input type="text" name="nombre">
            </fieldset>
        </form>
    </div>
</body>
</html>
```

```

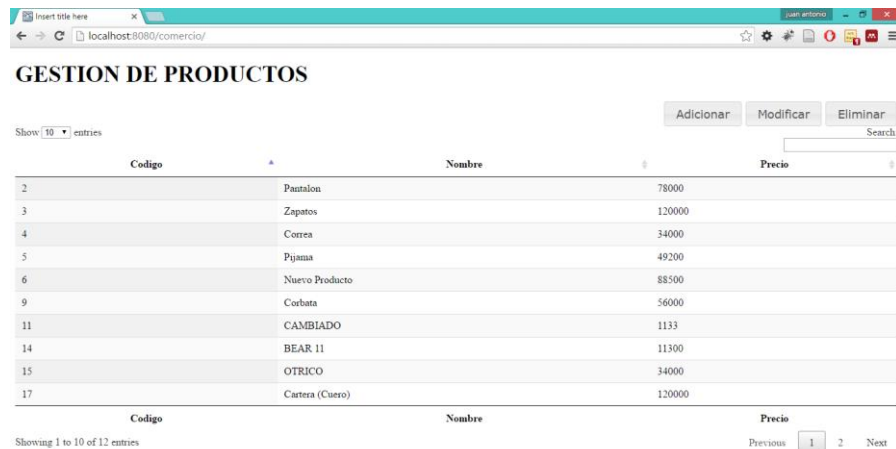
        id="nombre" value="" class="text ui-widget-content ui-corner-all" />
<label for="precio">Precio</label> <input type="text" name="precio"
id="precio" value="jane@smith.com"
class="text ui-widget-content ui-corner-all" />

<!--
    Allow form submission with keyboard without duplicating
    The dialog button
-->
<input type="submit" tabindex="-1"
style="position: absolute; top: -1000px">
</fieldset>
</form>
</div>

</body>
</html>

```

La aplicación web creada permite la gestión de los productos (CRUD, Create, Read, Update, Delete) empleando los servicios web tipo RESTful.

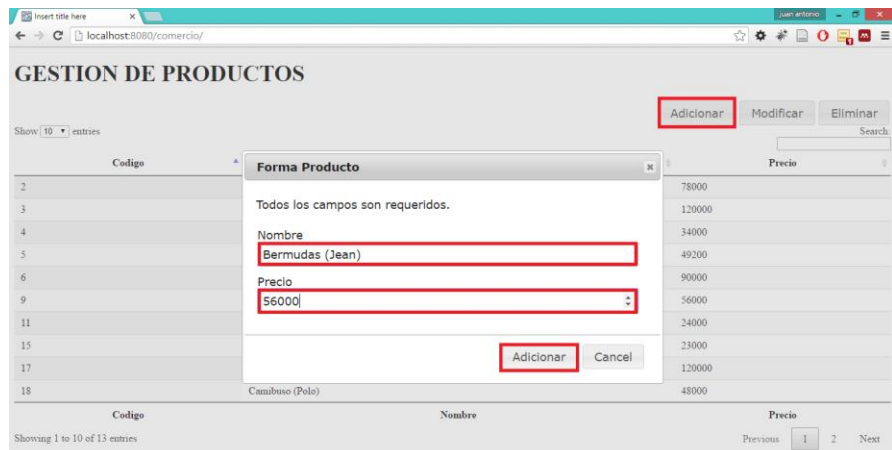


The screenshot shows a web browser window with the URL 'localhost:8080/comercio/'. The page title is 'GESTION DE PRODUCTOS'. Below the title is a toolbar with three buttons: 'Adicionar', 'Modificar', and 'Eliminar'. A search bar is also present. Below the toolbar is a table with 12 entries. The table has four columns: 'Codigo', 'Nombre', 'Precio', and an empty column. The data rows are as follows:

Codigo	Nombre	Precio	
2	Pantalón	78000	
3	Zapatos	120000	
4	Correa	34000	
5	Pijama	49200	
6	Nuevo Producto	88500	
9	Corbata	56000	
11	CAMBIADO	1133	
14	BEAR 11	11300	
15	OTRICO	34000	
17	Cartera (Cuero)	120000	

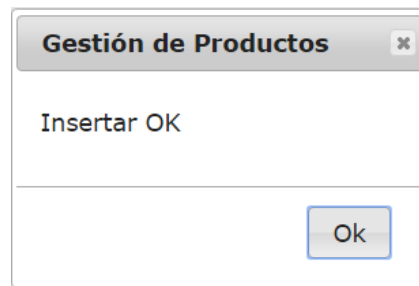
At the bottom of the table, it says 'Showing 1 to 10 of 12 entries'. There are also 'Previous', '1', '2', and 'Next' links for pagination.

Para adicionar un registro haga click en el boton [Adicionar] arriba en la barra de herramientas, se abrirá un cajón de dialogo, digite los datos solicitados (Nombre, Precio) y haga click en el boton [Adicionar].

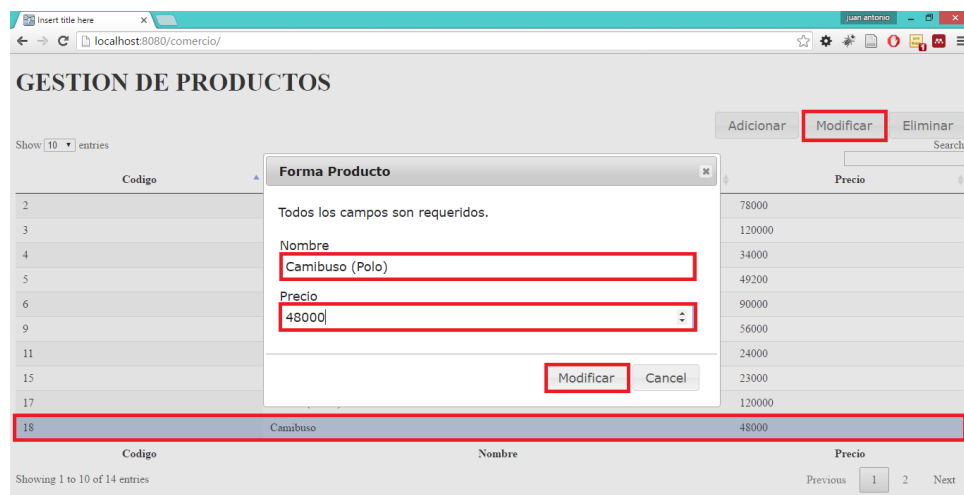


The screenshot shows the same web browser window as before, but with the 'Forma Producto' dialog box open. The dialog box has a title bar 'Forma Producto' and a close button. Inside the dialog, there is a message 'Todos los campos son requeridos.' Below this message are two input fields: 'Nombre' and 'Precio'. The 'Nombre' field contains the text 'Bermudas (Jean)' and the 'Precio' field contains the value '56000'. At the bottom of the dialog, there are two buttons: 'Adicionar' and 'Cancel'. The background table is partially visible, showing the same data as before.

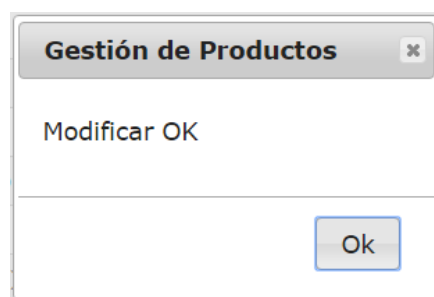
El servidor enviará un mensaje de respuesta indicado el estado de la petición.



Para modificar un registro, seleccione la fila (registro) que desea modificar, haga click en el boton [Modificar], los datos actuales del registro seleccionado se cargaran en el formulario, cambie los datos y finalmente haga click en el boton [Ejecutar].



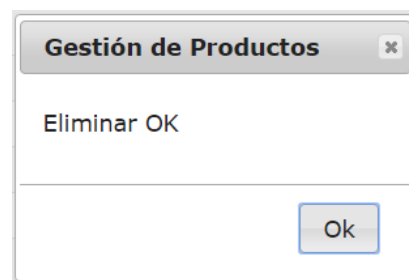
El servidor de aplicaciones enviara un mensaje con la respuesta a la petición de modificación del registro.



Para eliminar un registro, seleccione el registro que desea eliminar, haga click en el boton [Eliminar] y finalmente haga click en el boton [Ejecutar].



El servidor de aplicaciones enviara un mensaje con la respuesta a la petición de modificación del registro.



Si quiere cambiar el estilo de la página reemplace en el archivo index.jsp la línea donde se define el llamado a la hoja de estilo.

```
<link rel="stylesheet"
      href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
```

JQuery tiene una gran cantidad de temas y estilos que puede utilizar, por ejemplo pruebe con este tema.

```
<link rel="stylesheet"
      href="http://code.jquery.com/ui/1.11.4/themes/ui-lightness/jquery-ui.css">
```

Obtendrá como resultado una apariencia diferente.

The screenshot displays a web application for product management. The main heading is 'GESTION DE PRODUCTOS'. Below the heading, there are buttons for 'Adicionar', 'Modificar', and 'Eliminar', along with a search bar. A modal dialog titled 'Forma Producto' is open, showing a message 'Todos los campos son requeridos.' and input fields for 'Nombre' (containing 'Billetera (Cuero)') and 'Precio' (containing '90000'). The background shows a table of products with columns 'Codigo', 'Nombre', and 'Precio'.

Codigo	Nombre	Precio
13		23000
11		24000
22		25000
4		34000
9		45000
5		49200
24		56000
2		78000
21		80000
6	Billetera (Cuero)	90000

Showing 1 to 10 of 13 entries

Previous 1 2 Next