

CURSO STRUTS FRAMEWORK

INTEGRACIÓN DE TECNOLOGÍAS

STRUTS + EJB+ JPA

CRUD



Por el experto: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

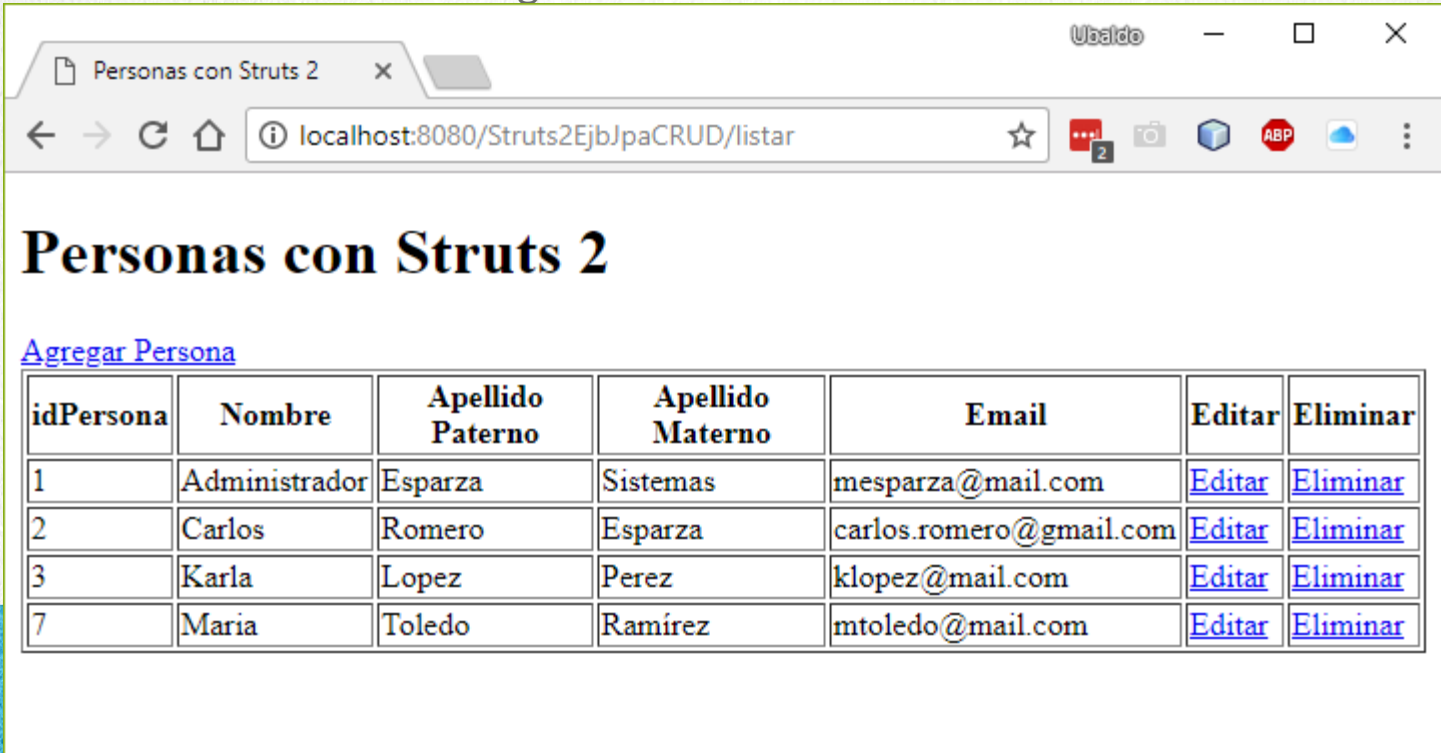


CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

Crear una aplicación para poner en práctica la integración de los frameworks Struts + EJB + JPA aplicando las operaciones CRUD (Cread-Read-Update-Delete) sobre la tabla de persona. Al finalizar deberemos observar lo siguiente:



The screenshot shows a web browser window with the title "Personas con Struts 2". The address bar displays "localhost:8080/Struts2EjbJpaCRUD/listar". The page content includes a link "Agregar Persona" and a table with the following data:

idPersona	Nombre	Apellido Paterno	Apellido Materno	Email	Editar	Eliminar
1	Administrador	Esparza	Sistemas	mesparza@mail.com	Editar	Eliminar
2	Carlos	Romero	Esparza	carlos.romero@gmail.com	Editar	Eliminar
3	Karla	Lopez	Perez	klopez@mail.com	Editar	Eliminar
7	Maria	Toledo	Ramírez	mtoledo@mail.com	Editar	Eliminar

REQUERIMIENTO DEL EJERCICIO

Mostrar el listado de personas utilizando Struts + EJB + JPA. Para ello necesitamos integrar las 3 tecnologías descritas.



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

1. COPIAR PROYECTO

Vamos a utilizar Maven para crear el proyecto Java Web. El proyecto se llamará StrutsEjbJpaCRUD. Este proyecto integrará las 3 tecnologías: Struts + EJB + JPA.

La tecnología que usaremos para la capa de datos será JPA. En la capa de Negocio utilizaremos EJB, y en la capa de presentación utilizaremos Struts 2.

Vamos a apoyarnos del proyecto StrutsEjbJpaBasico para realizar este proyecto, solo agregaremos los cambios necesarios en la capa de presentación que es donde básicamente están los cambios en la vista.

Empecemos con nuestro ejercicio:



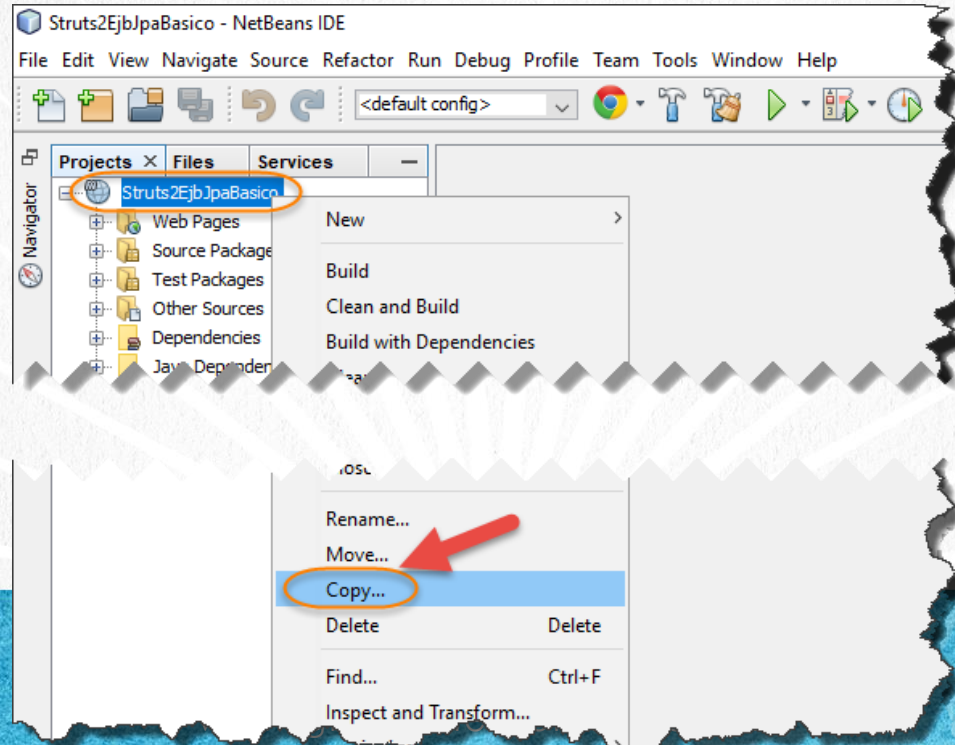
Experiencia y Conocimiento para tu vida

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

1. COPIAR PROYECTO

Vamos a partir del ejercicio StrutsEjbJpaBasico para crear nuestro proyecto ya que muchas de los archivos son idénticos, y solo modificaremos los archivos que sean necesarios. Copiamos el proyecto StrutsEjbJpaBasico y lo renombramos a StrutsEjbJpaCRUD:

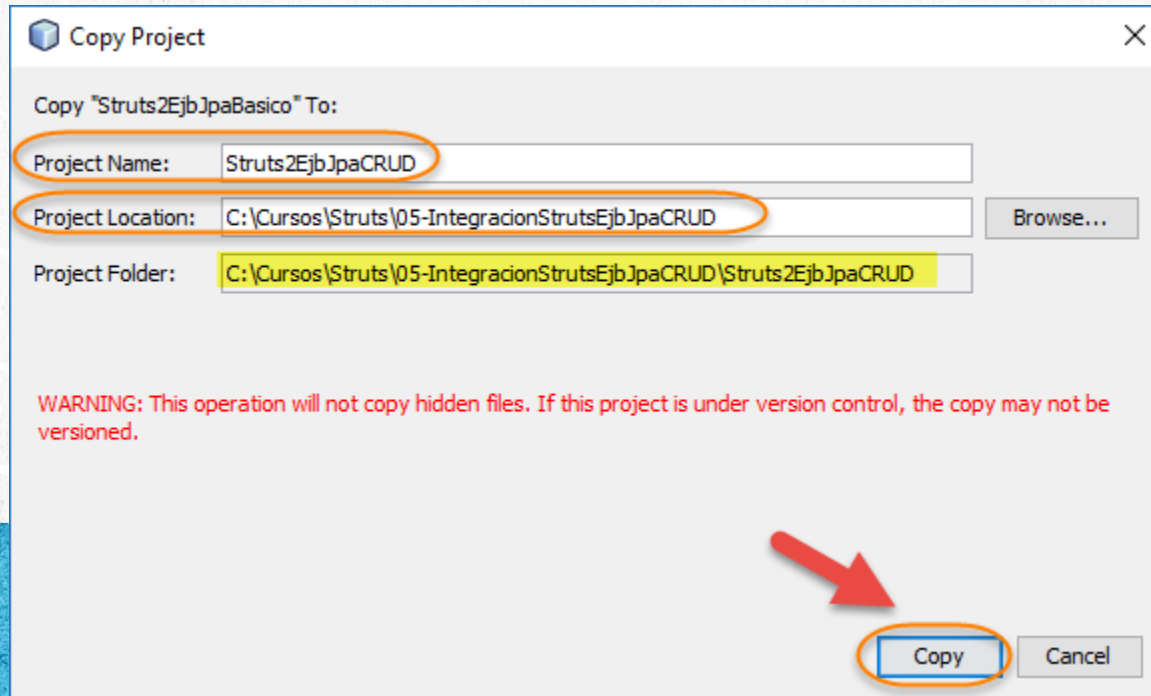


1. COPIAR PROYECTO

Renombramos el proyecto a StrutsEjbJpaCRUD y la ruta puede cambiar, nosotros vamos a utilizar la siguiente ruta:

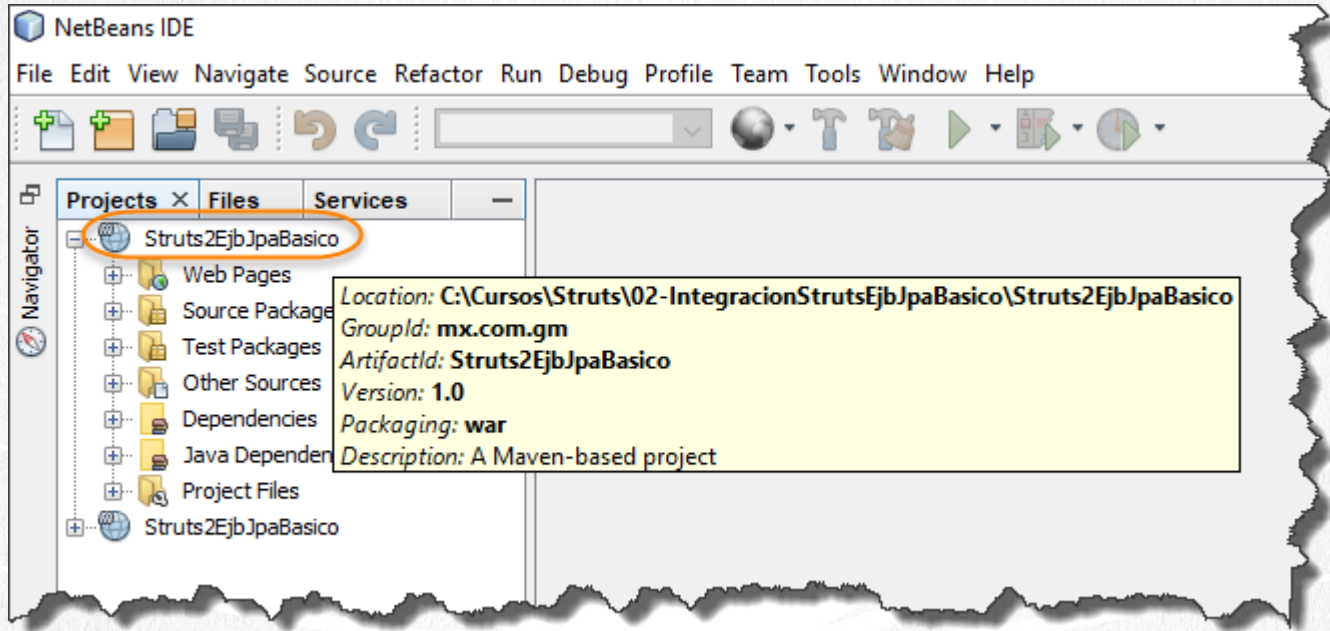
Nombre del Proyecto: StrutsEjbJpaCRUD

Project Location: C:\Cursos\Struts\05-IntegracionStrutsEjbJpaCRUD\StrutsEjbJpaCRUD



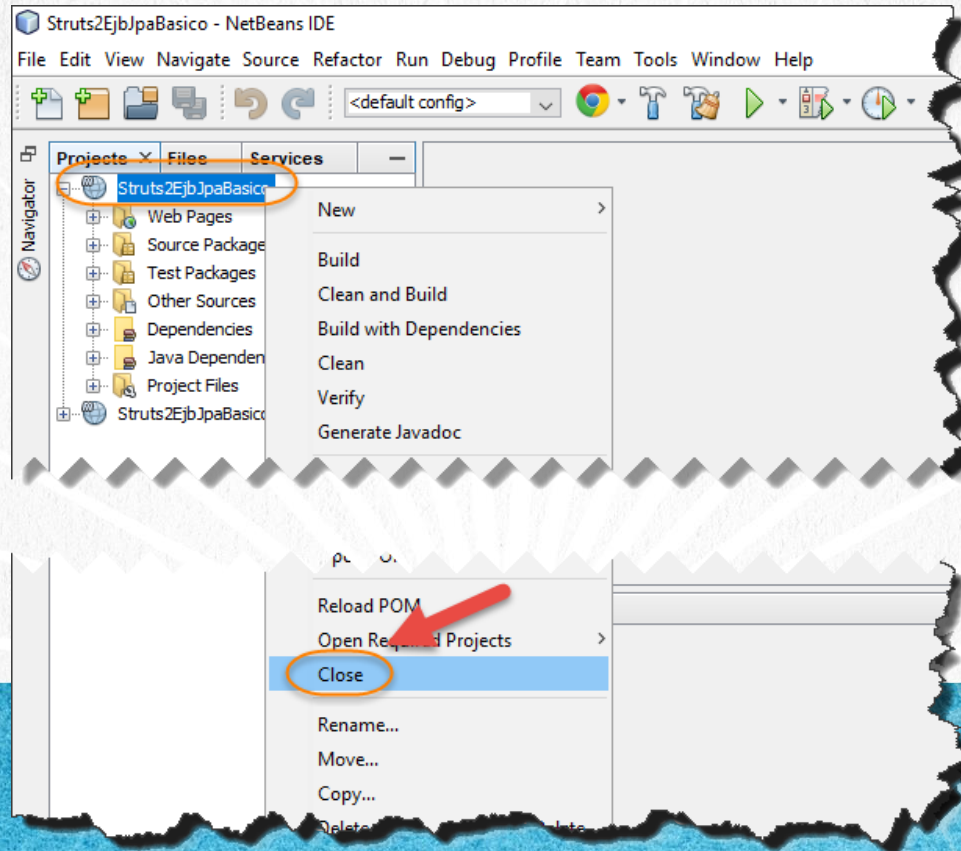
2. CERRAR PROYECTO

- Identificamos el proyecto que deseamos cerrar:



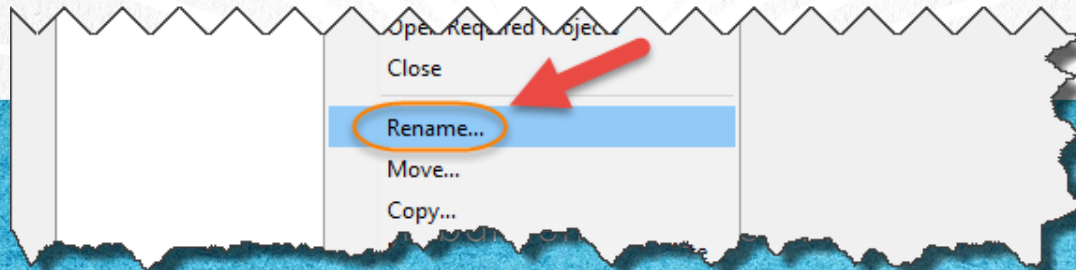
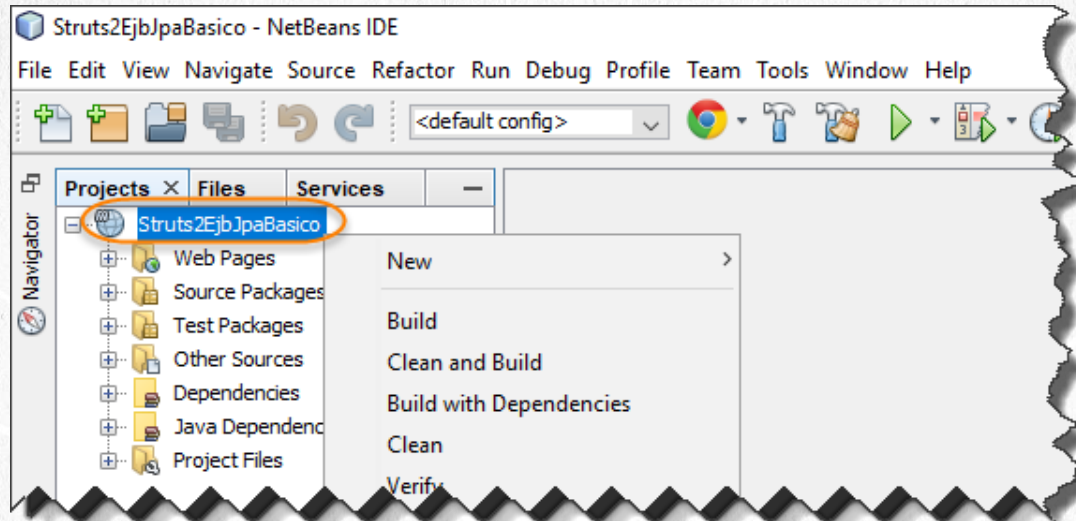
2. CERRAR PROYECTO

- Y Cerramos el proyecto que ya no vamos a utilizar:



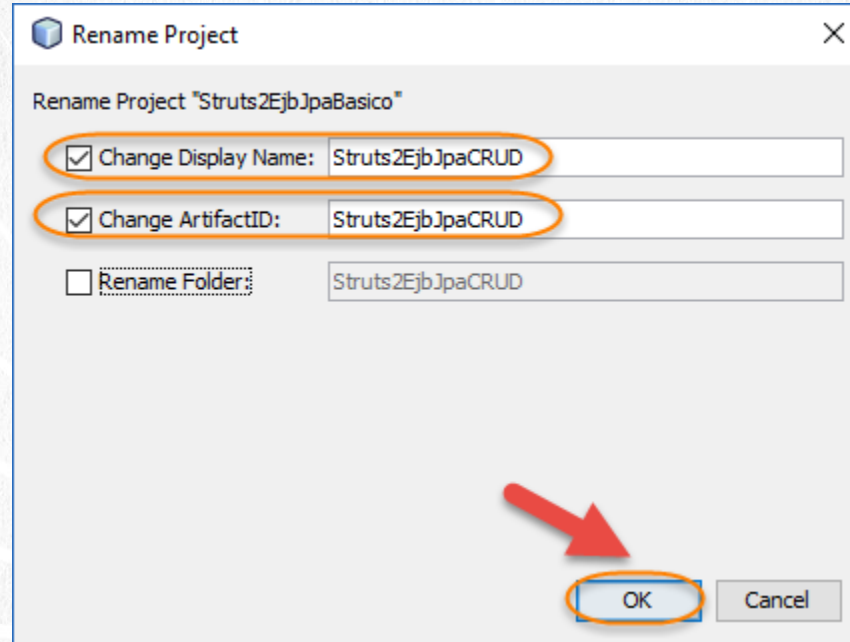
3. RENOMBAMOS EL PROYECTO

- Finalmente renombramos el proyecto como sigue:



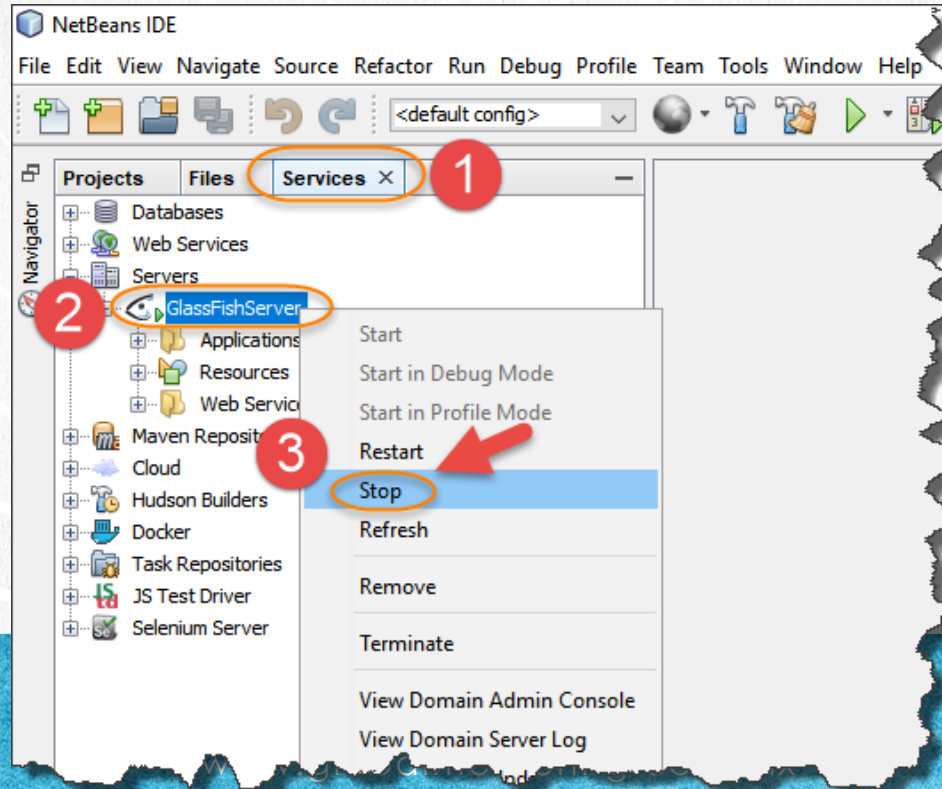
3. RENOMBAMOS EL PROYECTO

- Finalmente renombramos el proyecto como sigue:



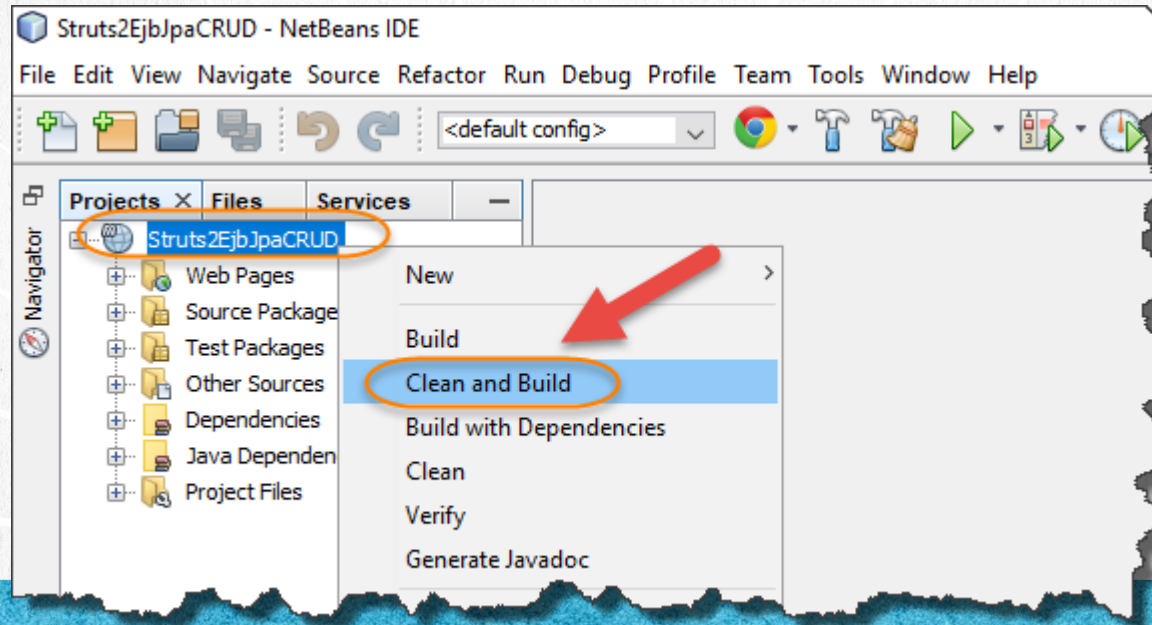
4. DETENEMOS GLASSFISH SI ESTUVIERA INICIADO

- Antes de hacer Clean & Build del proyecto para que descargue las librerías si fuera necesario, verificamos que el servidor de Glassfish no esté iniciado ya que puede haber problemas para hacer el proceso de Clean & build si el servidor está iniciado.



5. HACEMOS CLEAN & BUILD

• Para que se descarguen las nuevas librerías si fuera necesario, hacemos Clean & Build al proyecto. Si por alguna razón este proceso falla, se debe desactivar cualquier software como antivirus, Windows defender o firewall durante este proceso para que no se impida la descarga de archivos .jar de Java. Una vez terminado se pueden volver a activar estos servicios. Este proceso puede demorar varios minutos dependiendo de su velocidad de internet:

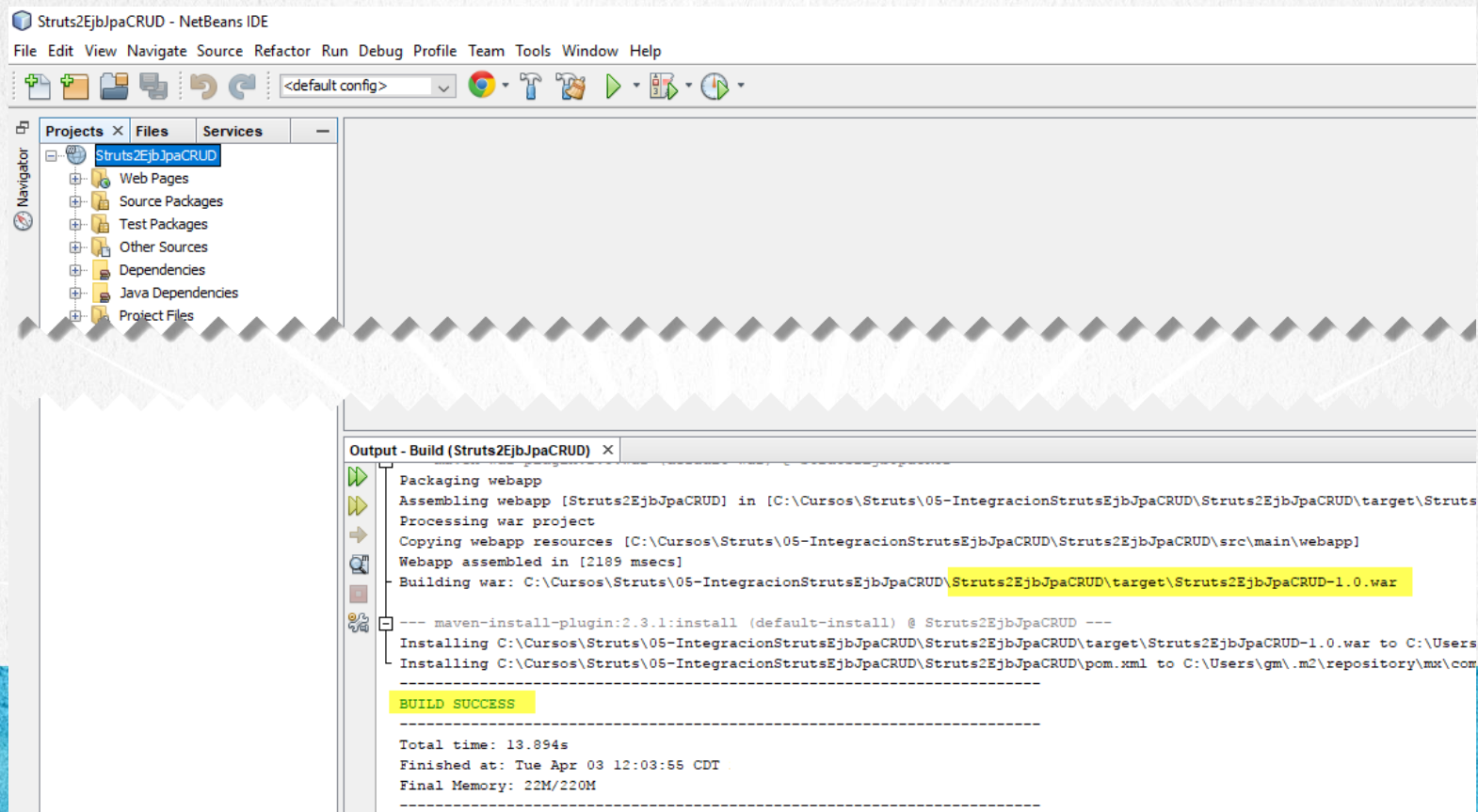


CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

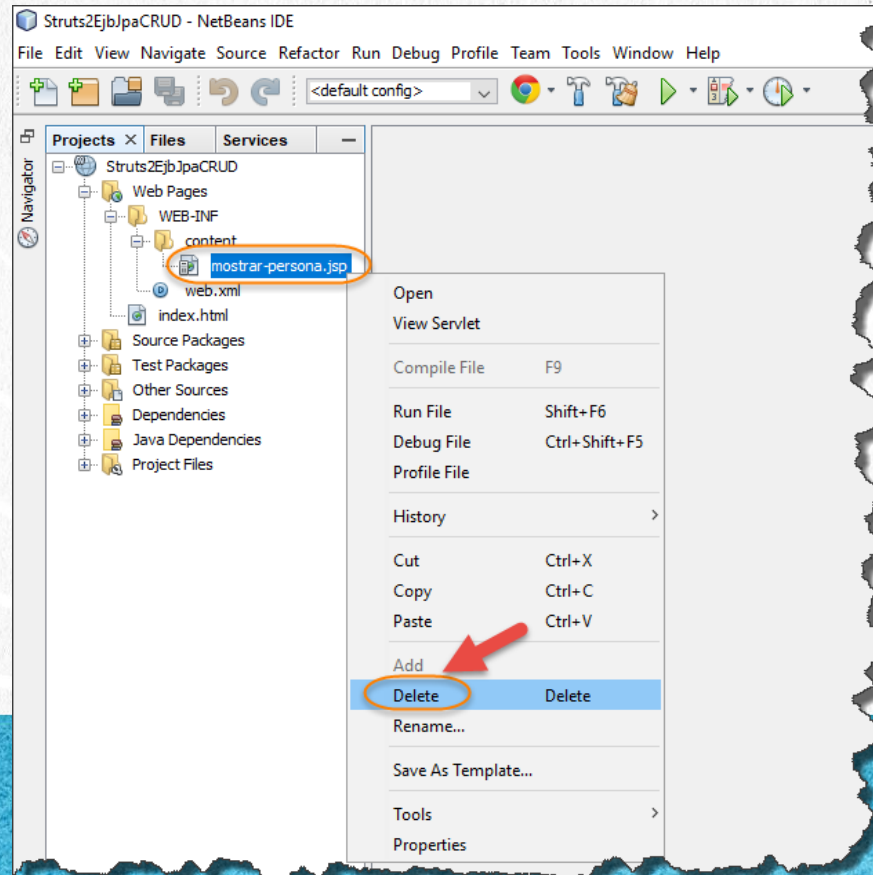
5. HACEMOS CLEAN & BUILD

- El resultado de ejecutar Clean & Build:



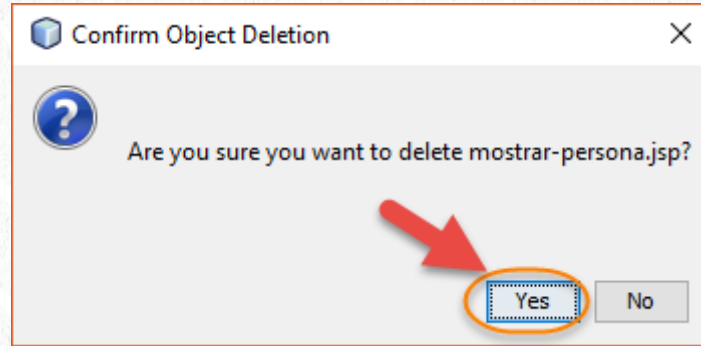
6. ELIMINAMOS EL ARCHIVO JSP

- Eliminamos el archivo mostrar-persona.jsp:



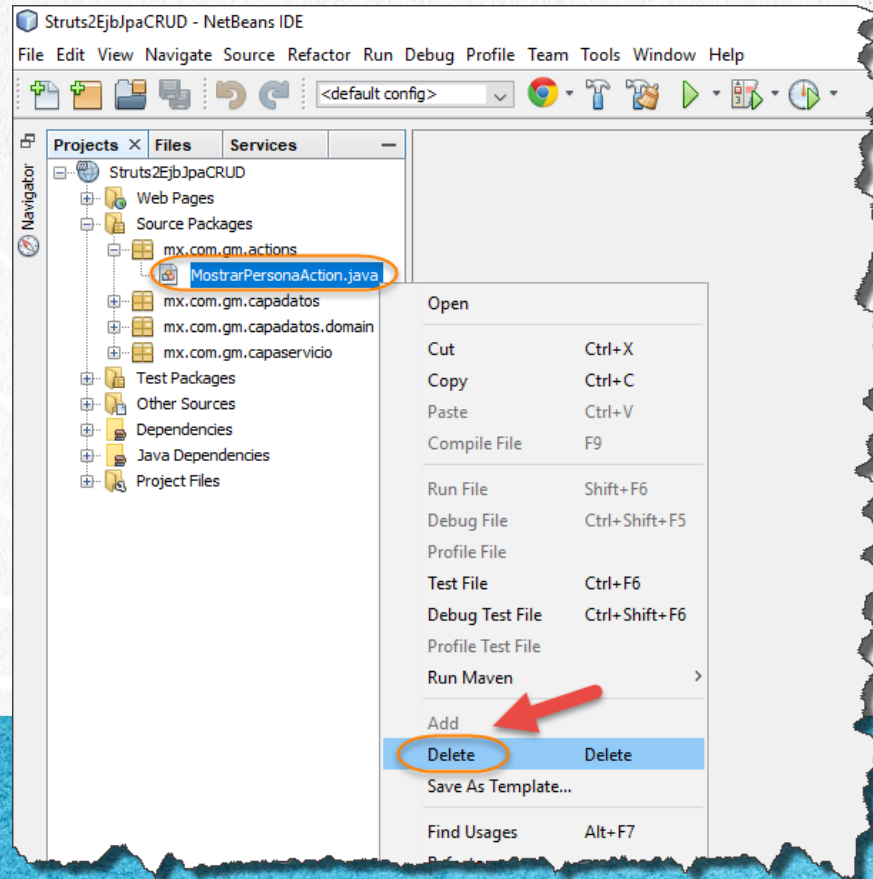
6. ELIMINAMOS EL ARCHIVO JSP

- Eliminamos el archivo mostrar-persona.jsp:



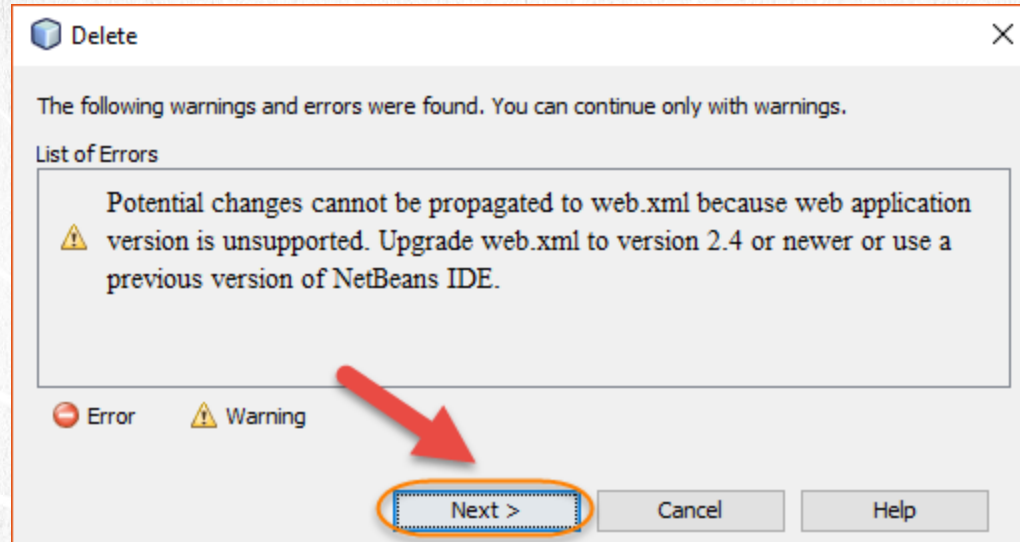
7. ELIMINAMOS EL ARCHIVO JAVA

- Eliminamos el archivo MostrarPersonaAction.java:



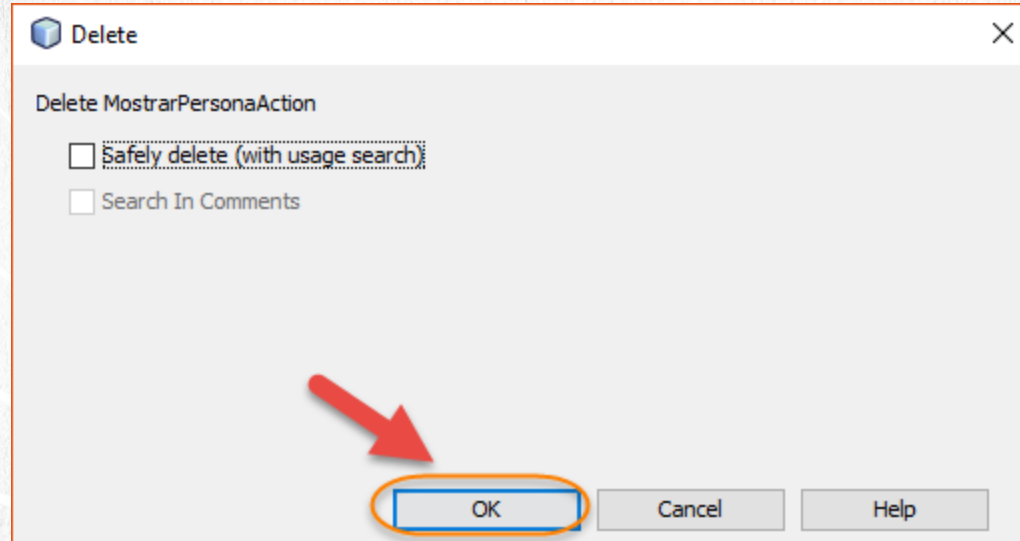
7. ELIMINAMOS EL ARCHIVO JAVA

- Eliminamos el archivo `MostrarPersonaAction.java`:



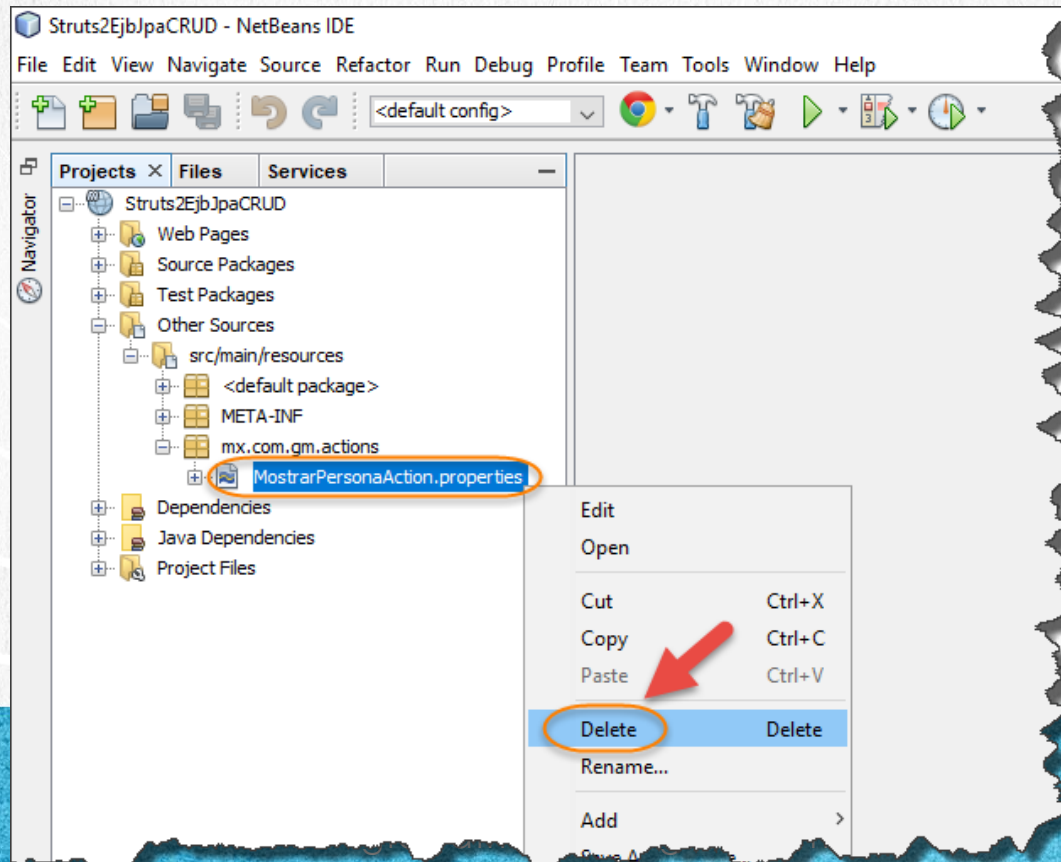
7. ELIMINAMOS EL ARCHIVO JAVA

- Eliminamos el archivo `MostrarPersonaAction.java`:



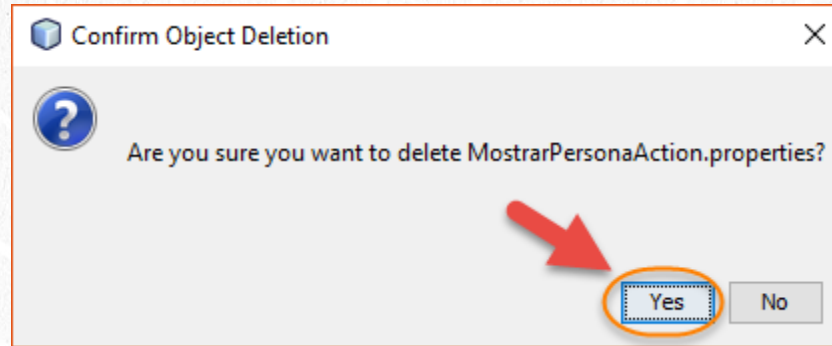
8. ELIMINAMOS EL ARCHIVO JAVA

- Eliminamos el archivo MostrarPersonaAction.properties:



8. ELIMINAMOS EL ARCHIVO JAVA

- Eliminamos el archivo `MostrarPersonaAction.properties`:



PASO 9. CREAR UNA CLASE JAVA

La clase `PersonasAction.java` que vamos a crear a continuación va a hacer las veces de Controlador (Action) y Modelo (Bean).

Vamos a extender de la clase `ActionSupport` y sobreescribiremos el método `execute`.

El modelo lo obtendremos con ayuda de la interface de servicio, el cual inyectaremos con ayuda de CDI de Java EE, esto también es posible debido al plug-in de integración entre Struts y CDI que agregamos al archivo `pom.xml`

Recordemos que debemos respetar las convenciones de Struts2, así que esta clase debe estar dentro de un paquete que contenga la palabra: `struts`, `struts2`, `action` o `actions`, además debe terminar con la palabra `Action`.

Esta clase sí tiene cambios respecto al ejercicio básico, ya que esta clase `Action` es precisamente la que soportará todos los cambios en la vista que queremos aplicar. Desde cambios en los nombres de nuestras clases, hasta nuevas funciones que utilizaremos para poder aplicar las operaciones CRUD (Create-Read-Update-Delete), es decir las operaciones de: listar, agregar, modificar y eliminar registros.

PASO 9. CREAR UNA CLASE JAVA

En la clase `PersonasAction.java` hemos agregado varios métodos de tipo `execute` y no solo uno. De tal manera que por cada acción que ejecutemos desde el navegador, esta clase de tipo `Action` será la encargada de procesar dichas acciones.

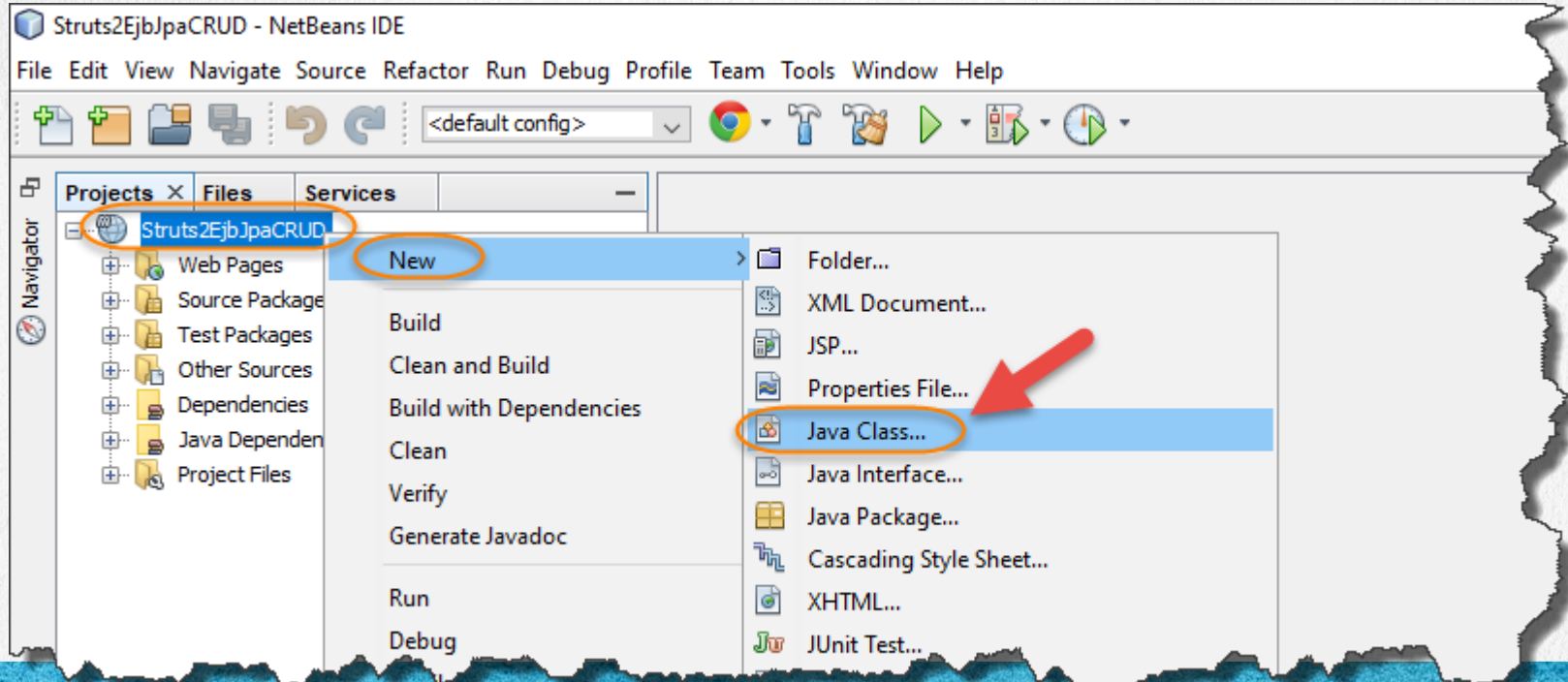
De esta forma estamos demostrando que una clase de tipo `Action` puede tener múltiples métodos de tipo `execute`, pero que estos métodos pueden tener cualquier nombre, y con ayuda de la anotación `@Action` es que definimos el nombre del `path` que procesará el método indicado con esta anotación. Por lo que no es necesario que el nombre del `path` y el nombre del método coincidan.

Otra característica que vemos en estos métodos es que en algunos casos fue necesario especificar la vista que se ejecuta después de haber terminado de ejecutar dicho método (resultado). Para ello hicimos de la anotación `@Result` y se especifican algunos atributos dependiendo del tipo de vista seleccionada, como puede ser el nombre del resultado, la ubicación de la vista y en algunos casos el tipo de resultado como es `redirect`, para que vuelva a ejecutar un `path` el método `Action` y no solamente se ejecute el `JSP`, ya que en los casos de agregar, modificar o eliminar un registro del listado, necesitamos volver a desplegar el nuevo listado actualizado, y para ello no basta con llamar directamente al `JSP` de listado, sino que debemos de pasar por el método `listar` de la clase `PersonasAction` y así se refresque el listado de personas.

Veamos cómo queda nuestra clase `PersonasAction.java`

PASO 9. CREAR UNA CLASE JAVA

- Creamos la clase PersonasAction.java:



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 9. CREAR UNA CLASE JAVA

- Creamos la clase PersonasAction.java:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

PASO 10. MODIFICAMOS EL CÓDIGO

Archivo PersonasAction.java:

Clic para ver el archivo

```
package mx.com.gm.actions;

import com.opensymphony.xwork2.ActionSupport;
import java.util.List;
import javax.inject.Inject;
import mx.com.gm.capadatos.domain.Persona;
import mx.com.gm.capaservicio.PersonaService;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.apache.struts2.convention.annotation.Action;
import org.apache.struts2.convention.annotation.Result;

public class PersonasAction extends ActionSupport {

    private List<Persona> personas;

    private Persona persona;

    @Inject
    private PersonaService personaService;

    Logger log = LogManager.getLogger(PersonasAction.class);

    @Action(value = "/listar", results = {
        @Result(name = "personas", location = "/WEB-INF/content/personas.jsp")
    })
    public String listar() {
        this.personas = personaService.listarPersonas();
        return "personas";
    }
}
```

PASO 10. MODIFICAMOS EL CÓDIGO

Archivo PersonasAction.java:

Clic para ver el archivo

```
@Action(value = "/agregarPersona", results = {
    @Result(name = "persona", location = "/WEB-INF/content/persona.jsp")})
public String agregar() {
    //Creamos un nuevo objeto de tipo persona
    persona = new Persona();
    return "persona";
}

@Action(value = "/editarPersona", results = {
    @Result(name = "persona", location = "/WEB-INF/content/persona.jsp")})
public String editar() {
    persona = personaService.recuperarPersona(persona);
    return "persona";
}

@Action(value = "/eliminarPersona", results = {
    @Result(name = "success", location = "listar", type = "redirect")})
public String eliminar() {
    //Recuperamos el objeto persona, ya que solo tenemos el idPersona
    log.info("Metodo eliminar persona antes de recuperar:" + persona);
    persona = personaService.recuperarPersona(persona); //revisar si se puede quitar el == y sigue funcionando, por
referencia
    log.info("Metodo eliminar persona despues de recuperar:" + persona);
    personaService.eliminarPersona(persona);
    return SUCCESS;
}
```


PASO 10. MODIFICAMOS EL CÓDIGO

Archivo PersonasAction.java:

Clic para ver el archivo

```
//No basta con mandar al JSP, sino a la accion de listar
//por ello redireccionamos a la accion listar
@Action(value = "/guardarPersona", results = {
    @Result(name = "success", location = "listar", type = "redirect")})
public String guardar() {
    //Diferenciamos la accion de agregar o editar con el idPersona
    if (persona.getIdPersona() == null) {
        personaService.agregarPersona(persona);
    } else {
        personaService.modificarPersona(persona);
    }
    return SUCCESS;
}

public Persona getPersona() {
    return persona;
}

public void setPersona(Persona persona) {
    this.persona = persona;
}

public List<Persona> getPersonas() {
    return personas;
}

public void setPersonas(List<Persona> personas) {
    this.personas = personas;
}
```

PASO 11. CREAR EL ARCHIVO DE PROPIEDADES

Creamos un archivo PersonasAction.properties. Este archivo tiene los mensajes que utilizaremos en las páginas JSP de Struts.

Veamos como queda este archivo PersonasAction.properties

Este archivo también se modificó conforme al ejercicio básico anterior, para soportar el nuevo listado y las operaciones de agregar y modificar.



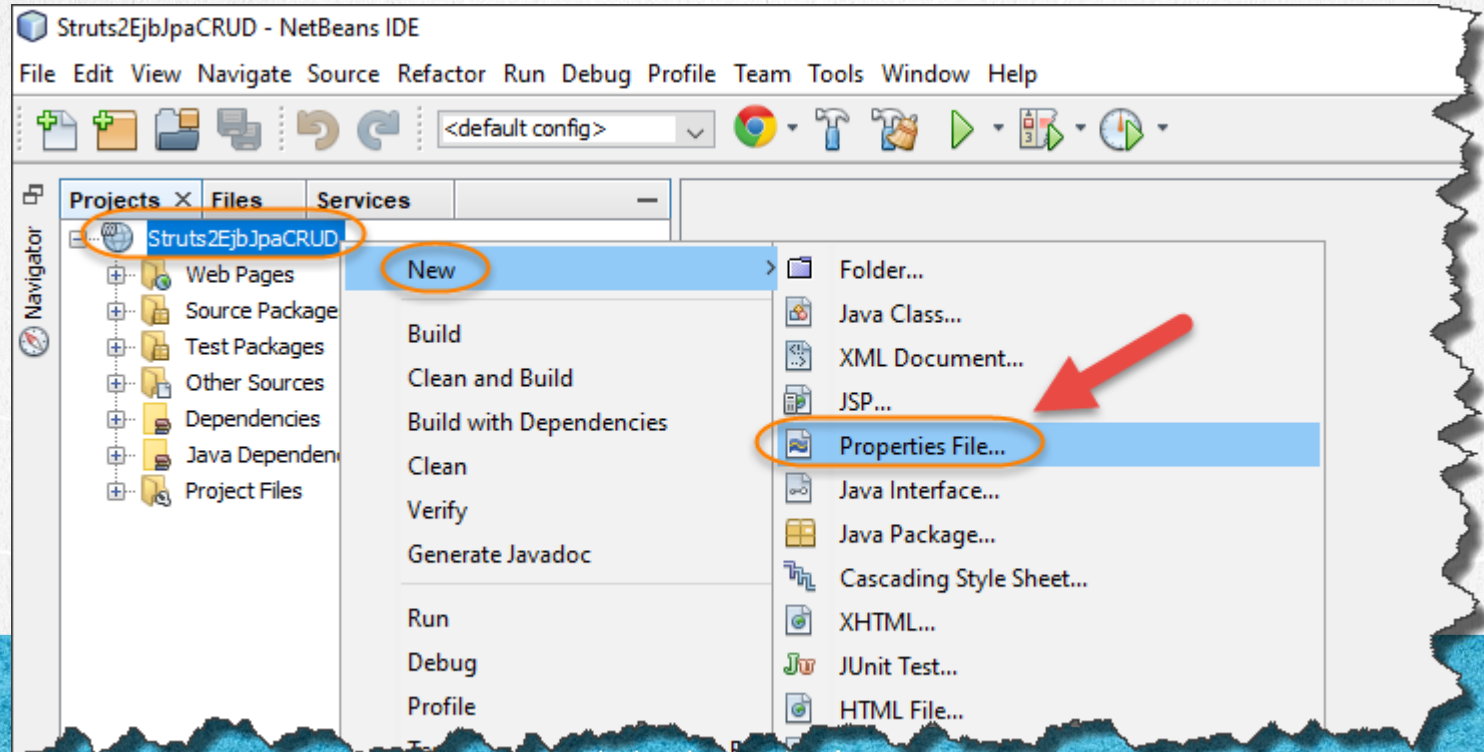
Experiencia y Conocimiento para tu vida

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 11. CREAR EL ARCHIVO DE PROPIEDADES

- Creamos el archivo PersonasAction.properties como sigue:



PASO 11. CREAR EL ARCHIVO DE PROPIEDADES

- Depositamos el archivo en la carpeta de resources según se muestra:

New Properties File

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name:

Project:

Folder:

Created File:

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 12. MODIFICAMOS EL CÓDIGO

Archivo PersonasAction.properties:

Clic para ver el archivo

```
pform.titulo: Personas con Struts 2
pform.contador: No. Registros Encontrados
pform.enviar: Enviar
pform.detalle: Detalle Persona
pform.agregar: Agregar Persona
pform.listado: Regresar listado de Personas
pform.editar: Editar
pform.eliminar: Eliminar
p.idPersona: idPersona
p.nombre: Nombre
p.apePat: Apellido Paterno
p.apeMat: Apellido Materno
p.email: Email
```

PASO 13. MODIFICAMOS EL ARCHIVO INDEX.HTML

En automático el IDE agrega un archivo llamado index.html. Sin embargo si este archivo no se crea debemos agregarlo al proyecto a nivel raíz de Web Pages.

El archivo index.html realmente aún no forma parte del framework de Struts, sin embargo será el punto de entrada para que se ejecute el framework de Struts, ya que desde este archivo indicaremos cual es la acción que deseamos que se ejecute.

En este ejercicio el path que utilizaremos será: **listar**



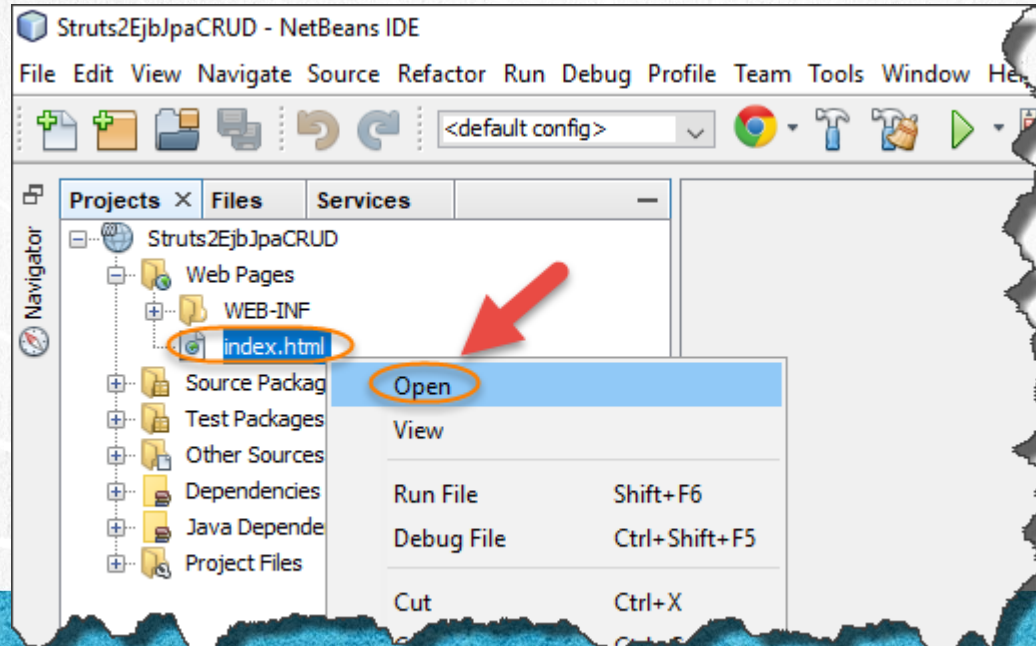
Experiencia y Conocimiento para tu vida

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 13. MODIFICAMOS EL ARCHIVO INDEX.HTML

- Modificamos el archivo index.html. En caso de que este archivo no exista a nivel raíz de la carpeta Web Pages lo creamos:



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 13. MODIFICAMOS EL CÓDIGO

Archivo index.html:

Clic para ver el archivo

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inicio</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <a href="listar">Ir a Personas</a>
  </body>
</html>
```


PASO 14. CREAR EL ARCHIVO JSP

Ahora creamos el archivo: `personas.jsp`. Recordar que este nombre corresponde con el path que se va a utilizar para llamar la acción correspondiente (`PersonasAction.java`), así que separamos por un guión medio cada palabra de la clase de tipo Action.

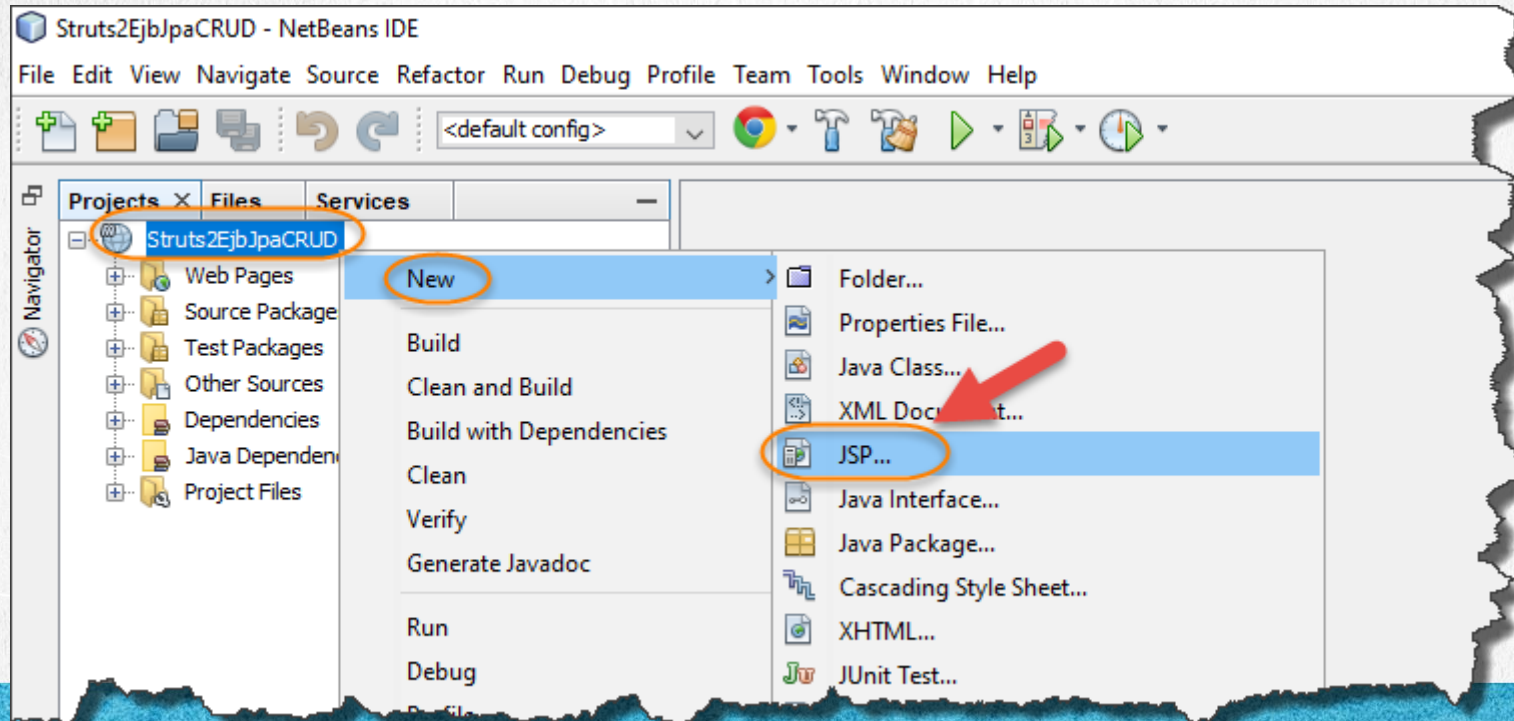
Este es el primer JSP que vamos a crear, el cual nos va a servir para mostrar el listado de objetos de tipo persona, y tendrá los links para agregar, modificar y eliminar un registro de tipo persona.

Además debemos depositar este JSP en la carpeta `/WEB-INF/content` según hemos visto en el tema de convenciones de Struts 2.

Este archivo también cambió conforme al ejercicio anterior básico, ya que hemos agregado el link de Agregar, así como las columnas de Editar y Eliminar en cada uno de los registros del listado. Veamos cómo quedó nuestro archivo `personas.jsp`

PASO 14. CREAR EL ARCHIVO JSP

- Creamos el archivo personas.jsp:



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 14. CREAR EL ARCHIVO JSP

- Creamos el archivo personas.jsp en la ruta mostrada:

New JSP

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name:

Project:

Location:

Folder: [Browse...](#)

Created File:

Options:

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

Description:

[< Back](#) [Next >](#) **Finish** [Cancel](#) [Help](#)

PASO 15. MODIFICAMOS EL CÓDIGO

Archivo personas.jsp:

Clic para ver el archivo

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
  <head>
    <title><s:text name="pform.titulo" /></title>
  </head>
  <body>
    <h1><s:text name="pform.titulo" /></h1>
    <a href="<s:url action="agregarPersona"/>"><s:text name="pform.agregar" /></a>
    <s:if test="personas.size() > 0">
      <div>
        <table border="1">
          <tr>
            <th><s:text name="p.idPersona" /></th>
            <th><s:text name="p.nombre" /></th>
            <th><s:text name="p.apePat" /></th>
            <th><s:text name="p.apeMat" /></th>
            <th><s:text name="p.email" /></th>
            <th><s:text name="pform.editar" /></th>
            <th><s:text name="pform.eliminar" /></th>
          </tr>
```


PASO 15. MODIFICAMOS EL CÓDIGO

Archivo personas.jsp:

Clic para ver el archivo

```
<s:iterator value="personas">
  <tr>
    <td><s:property value="idPersona" /></td>
    <td><s:property value="nombre" /></td>
    <td><s:property value="apellidoPaterno" /></td>
    <td><s:property value="apellidoMaterno" /></td>
    <td><s:property value="email" /></td>
    <td>
      <s:url action="editarPersona" var="editarURL">
        <s:param name="persona.idPersona" value="%{idPersona}"></s:param>
      </s:url>
      <s:a href="%{editarURL}"><s:text name="pform.editar" /></s:a>
    </td>
    <td>
      <s:url action="eliminarPersona" var="eliminarURL">
        <s:param name="persona.idPersona" value="%{idPersona}"></s:param>
      </s:url>
      <s:a href="%{eliminarURL}"><s:text name="pform.eliminar" /></s:a>
    </td>
  </tr>
</s:iterator>
</table>
</div>
</s:if>
</body>
</html>
```

PASO 16. CREAR EL ARCHIVO JSP

Ahora creamos el archivo: `persona.jsp`. En este caso este es un nuevo archivo JSP, el cual no hace un mapeo directo con una acción, ya que utilizaremos la misma clase `PersonasAction` para procesar los valores que utilizará la vista `persona.jsp`.

Este es el segundo JSP que crearemos, y lo utilizaremos básicamente para mostrar el detalle de un objeto de tipo `persona`. Y lo usaremos tanto para agregar o modificar un objeto de tipo `persona`.

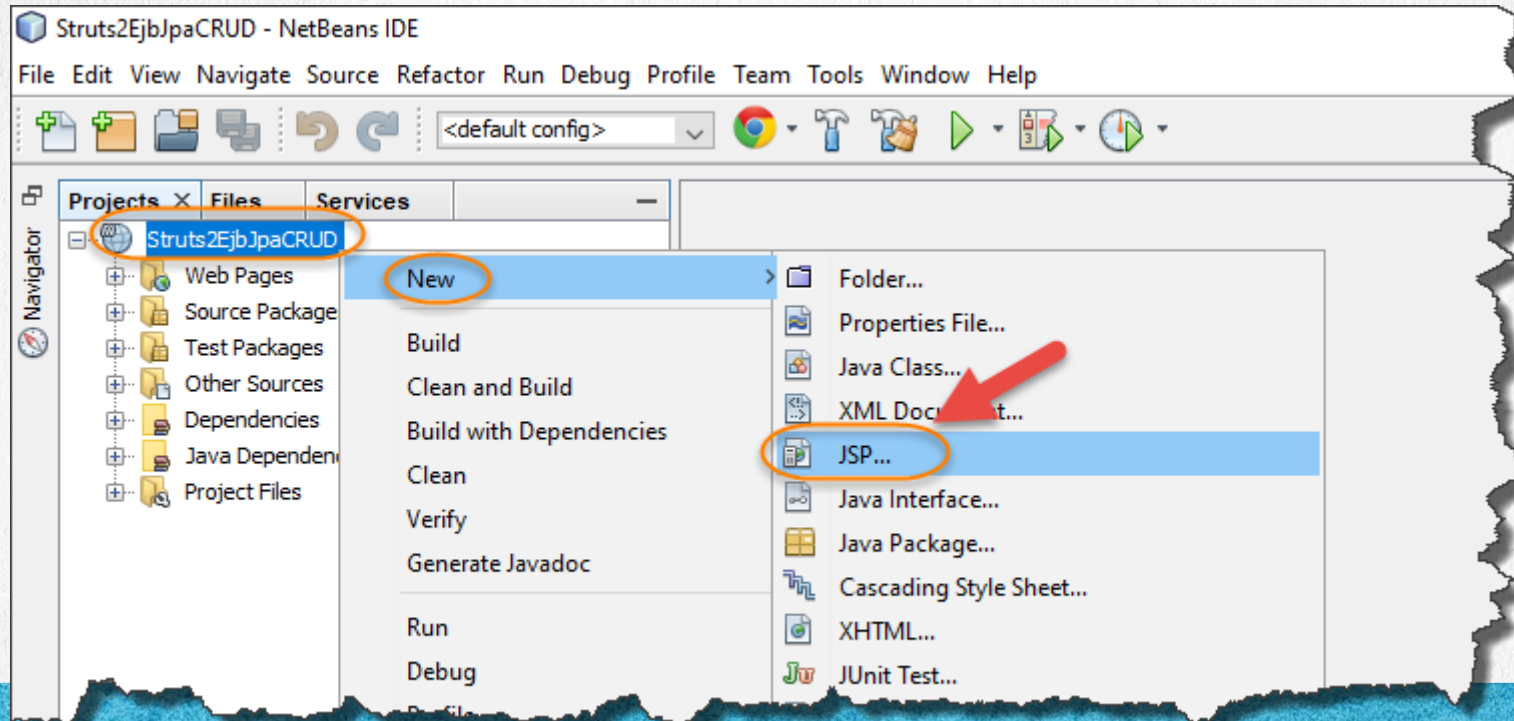
La única diferencia entre agregar y modificar una persona es el valores de `idPersona`, si es igual a nulo quiere decir que aún no tiene una representación en la base de datos y por lo tanto se hace un insert, es decir, se agrega a la base de datos. Por otro lado, si el valor es distinto a nulo, quiere decir que ya hay una representación en base de datos de ese objeto de tipo `Persona`, entonces estamos modificando el objeto (`update`).

Es importante observar el uso de `<s:hidden>` para incluir el campo de `idPersona`. Así que aunque no observemos este valor en el formulario, sí se está enviando al navegador como campo oculto, pudiendo tener un valor nulo si es un nuevo registro, o un valor distinto de nulo si es un registro que se está modificando.

Este archivo no es necesario depositarlo en la carpeta [/WEB-INF/content](#), sin embargo lo depositaremos allí para seguir manteniendo un orden en la organización de archivos JSPs.

PASO 16. CREAR EL ARCHIVO JSP

- Creamos el archivo persona.jsp:



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

PASO 16. CREAR EL ARCHIVO JSP

- Creamos el archivo persona.jsp en la ruta mostrada:

New JSP

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

Description:

PASO 17. MODIFICAMOS EL CÓDIGO

Archivo persona.jsp:

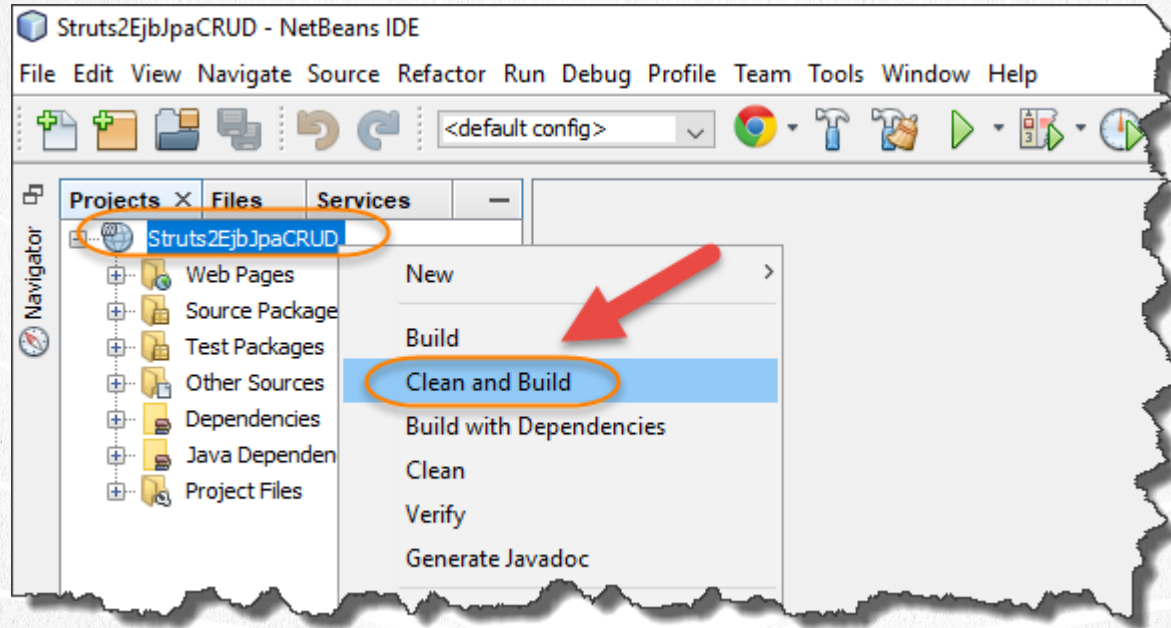
Clic para ver el archivo

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
  <head>
    <title><s:text name="pform.detalle" /></title>
  </head>
  <body>
    <h1><s:text name="pform.detalle" /></h1>
    <a href="<s:url action="listar"/>"><s:text name="pform.listado" /></a>

    <s:form action="guardarPersona">
      <s:hidden name="persona.idPersona" />
      <s:textfield name="persona.nombre" key="p.nombre" />
      <s:textfield name="persona.apellidoPaterno" key="p.apePat" />
      <s:textfield name="persona.apellidoMaterno" key="p.apeMat" />
      <s:textfield name="persona.email" key="p.email" />
      <s:submit action="guardarPersona" key="pform.enviar"/>
    </s:form>
  </body>
</html>
```

PASO 18. HACEMOS CLEAN & BUILD

- Hacemos Clean & Build para asegurarnos que tenemos todo listo:

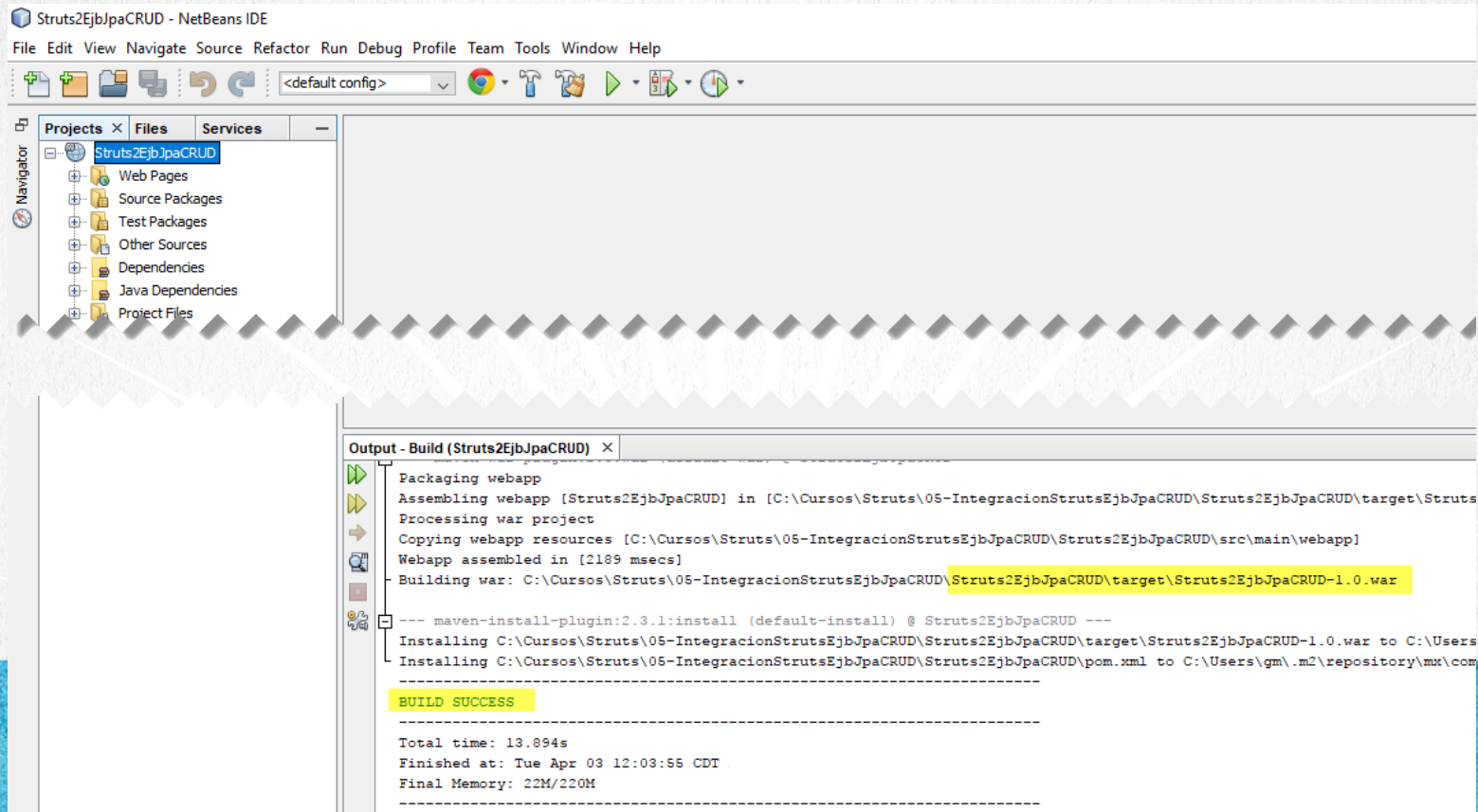


CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

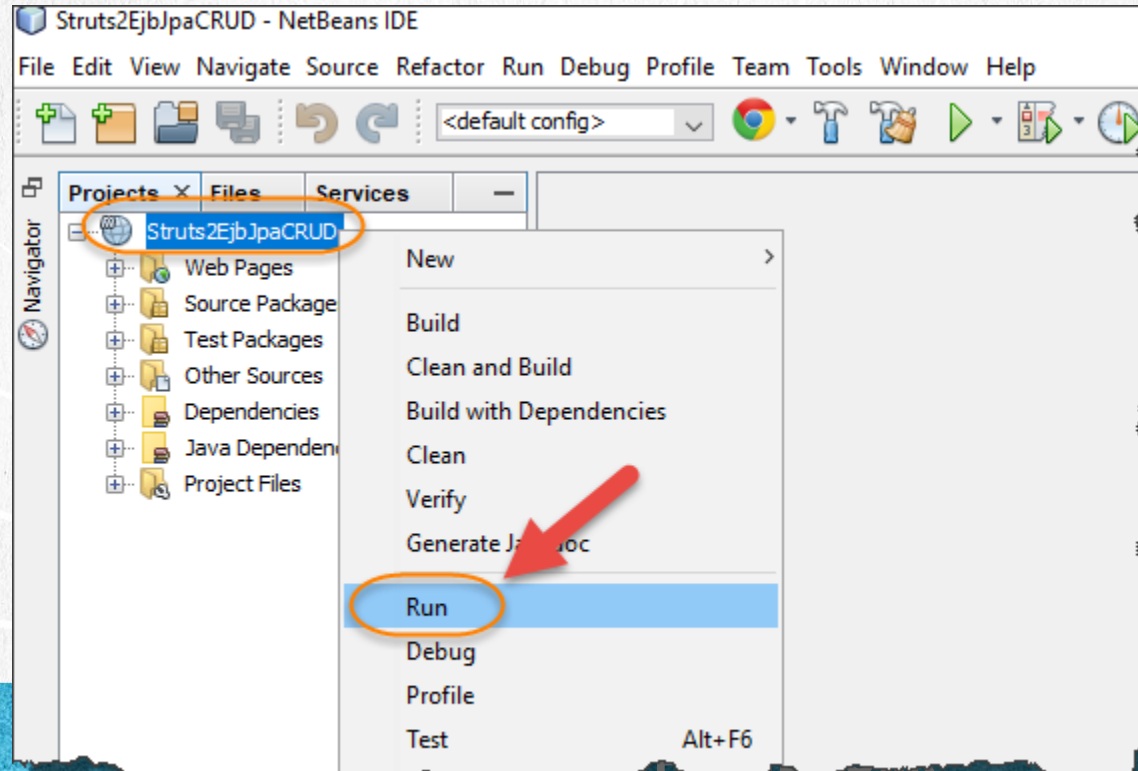
PASO 18. EJECUTAMOS CLEAN & BUILD

- Observamos que el proceso se haya ejecutado con éxito (Build Success):



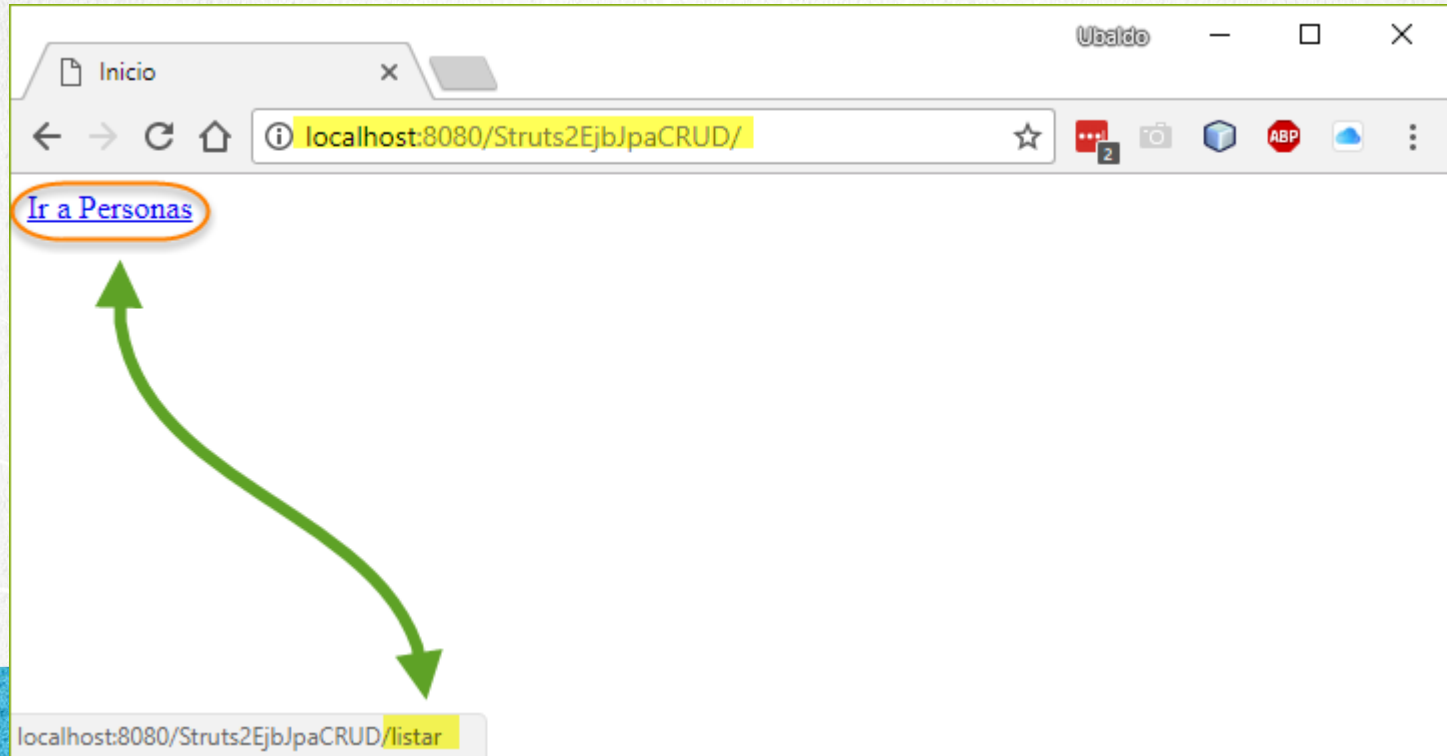
PASO 19. EJECUTAMOS LA APLICACIÓN

- Ejecutamos la aplicación StrutsEjbJpaCRUD como sigue:



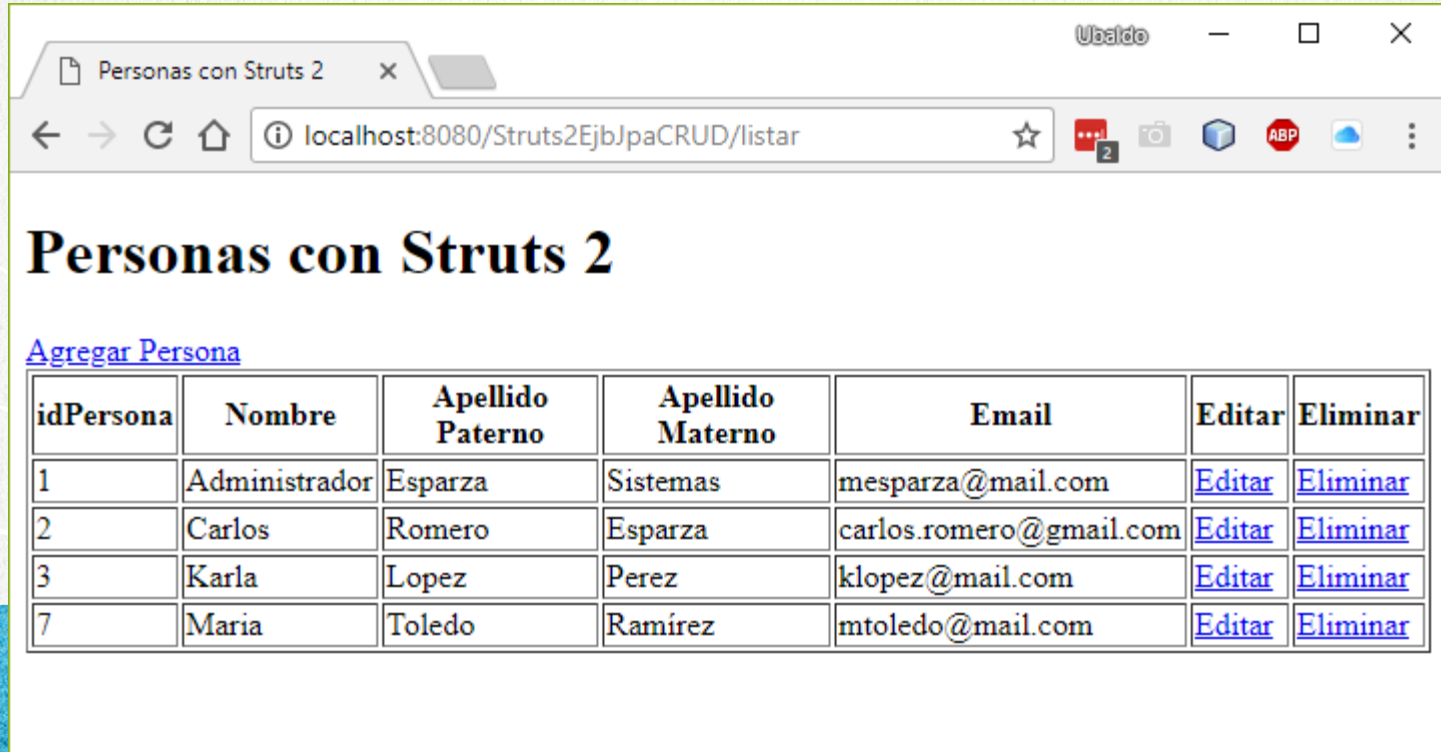
PASO 19. EJECUTAMOS LA APLICACIÓN

- Observamos el resultado como sigue, y damos clic en el link:



PASO 19. EJECUTAMOS LA APLICACIÓN

- Observamos que obtenemos la misma información, solo aplicamos los cambios necesarios en la capa de la vista en el framework de Struts, así que los cambios fueron relativamente muy sencillos de realizar.



Personas con Struts 2

[Agregar Persona](#)

idPersona	Nombre	Apellido Paterno	Apellido Materno	Email	Editar	Eliminar
1	Administrador	Esparza	Sistemas	mesparza@mail.com	Editar	Eliminar
2	Carlos	Romero	Esparza	carlos.romero@gmail.com	Editar	Eliminar
3	Karla	Lopez	Perez	klopez@mail.com	Editar	Eliminar
7	Maria	Toledo	Ramírez	mtoledo@mail.com	Editar	Eliminar

RECOMENDACIONES FINALES

Si por alguna razón falla el ejercicio, se pueden hacer varias cosas para corregirlo:

- 1) Detener el servidor de Glassfish
- 2) Hacer un Clean & Build del proyecto para tener la versión más reciente compilada
- 3) Volver a hacer Run del proyecto (desplegar nuevamente el proyecto en el servidor)

Si lo anterior no funciona, pueden probar cargando el proyecto resuelto el cual es funcional 100% y descartar problemas de configuración en su ambiente o cualquier otro error de código.

La configuración por convenciones de Struts 2, es muy sensible, de tal manera que todo debe estar escrito tal como se especificó en el ejercicio, ya que cualquier cambio en los nombres provocará que no se ejecute correctamente el ejercicio.

La integración con otros frameworks y tecnologías como EJB y JPA también es muy propenso a errores, así que puedes apoyarte del proyecto resuelto que te entregamos, el cual es 100% funcional, y así apoyarte en cualquier momento de esta documentación y los proyectos resueltos que te entregamos.

CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx

CONCLUSIÓN DEL EJERCICIO

Con este ejercicio hemos creado una aplicación que integra las 3 tecnologías como son:

- Struts 2
- EJB
- JPA

Aplicamos también el concepto de CDI para la integración de las tecnologías entre Struts2, EJB y JPA, de esta manera fue bastante sencillo unificar las tecnologías, incluso más sencillo que con Spring Framework, ya que la tecnología de Java EE simplifica varios de los pasos que deben realizarse con Spring Framework.

En este ejercicio hemos aplicado las operaciones CRUD (Create-Read-Update-Delete) para la tabla de persona.

Se deja como ejercicio probar cada una de las acciones para comprobar que todo sigue funcionando solo con los pocos cambios aplicados al proyecto.

CURSO ONLINE

STRUTS 2 FRAMEWORK

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida



CURSO STRUTS FRAMEWORK

www.globalmentoring.com.mx