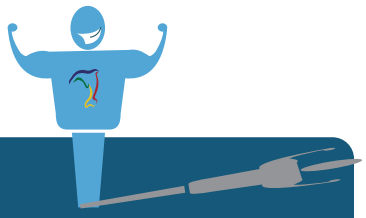
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

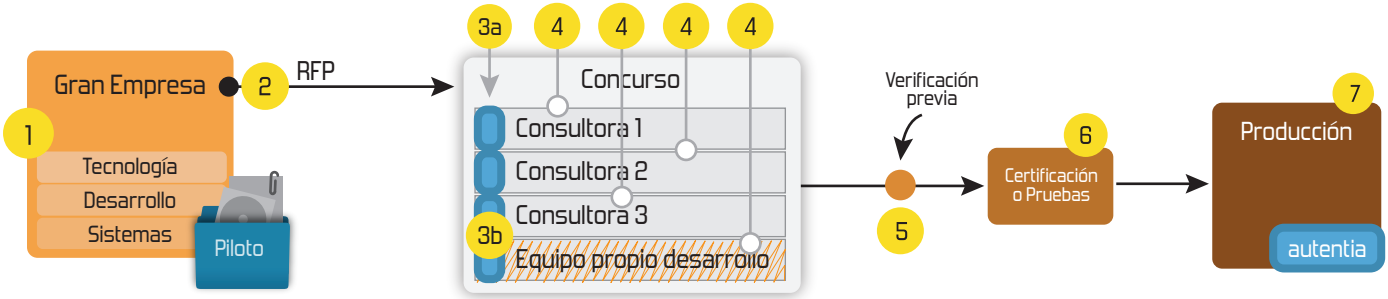
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por:
Roberto Canales Mora 2003-2004.

Si te gusta lo que ves, **puedes contratarme** para impartir **cursos presenciales** en tu empresa o para ayudarte en proyectos (Madrid).

Estamos creando las bases de la empresa en la que seguro te gustaría trabajar ... ayudanos a hacerla crecer ... presentándonos a tus jefes

Contacta: rcanales@autentia.com.

Una vez escuche esta frase:

Cuando todo esta bajo control, es que no vamos suficientemente deprisa

La mayoría de las veces construimos estos tutoriales al mismo tiempo que recordamos como hacer cosas o probamos nuevas tecnologías, poniendo más celo en el contenido que en la forma (cosa que veo que perturba a muchos de los lectores por lo que os pido disculpas). Es cuestión de criterios .... aunque trataremos de mejorar.

Espero que seáis comprensivos y generosos ayudando a encontrar las erratas, faltas de ortografía, enlaces rotos u otros posibles errores en estas páginas. Escribidme siempre que encontréis algún error y pronto ya no habrá .... y os estaremos francamente agradecidos. Este es tu Web .... colabora libremente ...

Roberto Canales

Descargar este documento en formato PDF webstruts.pdf

Aplicación paso a paso con Struts

En numerosas ocasiones me escriben usuarios del Web consultándome como comenzar a construir aplicaciones Web con Java. Hoy vamos a ver un posible modo.... y, como siempre, vamos a tratar cambiar la perspectiva típica ....

Os advierto que en este caso no va a ser un tutorial muy básico....

Análisis del Problema

No debemos confundir el medio y el fin. Nuestro objetivo es construir una aplicación y struts es unos más de los componentes técnicos que vamos a utilizar.

Lo más importante es definir de un modo inequívoco que es lo que queremos hacer, es decir, tomar los requisitos:

Queremos construir un subsistema para el Web www.adictosaltrabajo.com que nos permita:

1. Proporcionar a los usuarios la capacidad de decir de donde nos escriben e introducir algunos datos que nos permitan obtener estadísticas.
2. Posteriormente querríamos mostrar en un mapa visual los datos adquiridos (esto lo dejamos para otro tutorial)

Los objetivos vemos que nos son excesivamente ambiciosos pero nos puede valer. Los requisitos son independientes de la tecnología....

Ahora vamos a, antes de tirar una sola línea de código, profundizar en los problemas para anticipar posibles riesgos ligados a la indeterminación de los requisitos o la falta de análisis de potenciales riesgos.

Análisis del problema

Debemos hacernos algunas preguntas

- ¿Cuales son los datos que queremos que nos introduzcan (directa e indirectamente a través de su navegación)?
- ¿Que rechazo puede producir introducir esos datos por parte del usuario?
- ¿Como puede afectar esta información al tratamiento de datos que hacemos en el Web asociados a la LOPD (Ley Orgánica de Protección de Datos)?
- ¿Cual va a ser el volumen de datos a manejar?
- ¿Que potenciales riesgos de seguridad puede tener la aplicación para nuestro sistema?

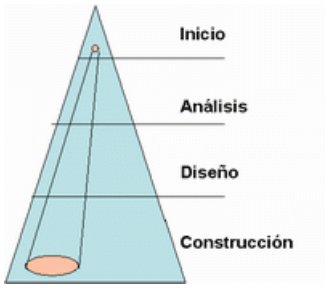
Nos respondemos

- Para favorecer que a los usuarios no le de mucho apuro rellenar la información, vamos a solicitar solamente el país y darles la oportunidad de hacer algún comentario del Web.
- Como no les pedimos datos personales (vinculados a un usuario en concreto) no afecta al tratamiento personal de datos.
- El volumen de datos se plantea que sea pequeño pero debemos establecer mecanismos de seguridad para que algún usuario maliciosos no nos saturase el Web con la entrada masiva de información (siempre hay alguien que nos complica la vida)

También nos debemos preguntar ¿estas son todas las preguntas que nos tenemos que hacer? ¿estamos planteando bien la toma de requisitos?

Ya os adelanto que no lo estamos haciendo demasiado bien aunque esto es otra guerra que ya lucharemos algún día (conceptualización a acotamiento de un problema, valoración rápida de esfuerzo y coste, identificación de riesgos típicos y particulares, etc.)

Si no somos capaces de hacer una buena definición del problema utilizando buenas técnicas de análisis, los problemas en la fase de desarrollo serán mucho más grandes,



Además se nos juntarán muchos problemas más:

- La introducción de nueva funcionalidad (el tiempo pasa y las necesidades evolucionan)
- Las correcciones de errores conceptuales
- Las prisas por no haber llevado un ritmo de trabajo estable y habernos confiado en un principio
- La complejidad de ciertas funciones que en principio parecían mucho mas sencillas.

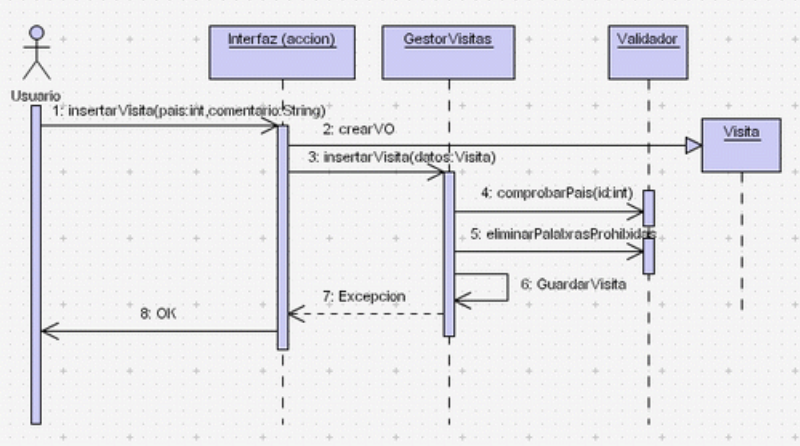
Diseño de la solución

Primer problema

Realizamos una descomposición del trabajo a realizar (**WBS** Work Breakdowns Structure)

- Definir la tabla de países (y llenarla). Estudiar como luego podríamos situar en un plano los resultados
- Definir la tabla de visitantes (con el campo para observaciones)
- Definir el componente que sea capaz de interaccionar con estas tablas (acceso a datos)
- Definir las reglas para validar si los campos que introduce el usuario son válidos
  - Numero de inserciones introducidas en un día (para que no nos boqueen el sistema)
  - Rango de valores (no nos podemos fiar de las peticiones que recibimos)
- Diseñar el interfaz de usuario y los posibles mensajes de error.

E incluso nos planteamos una posible solución (aplicando nuestro conocimiento sobre patrones de asignación de responsabilidad [GRASP])



Si diseñamos, podemos cuestionar el diseño y hacernos preguntas del estilo.

¿Qué es más conveniente a la hora de validar los parámetros?

1. ¿Que el gestor de visitas valide el país (el componente de negocio GestorVisitas)?
2. ¿Que el objeto Visita sea quien valide que los campos con los que se inicializa sean correctos (Visita en el constructor)?
3. ¿Que el interfaz (struts) valide los parámetros antes de continuar?

Pues no es tan sencillo y dependerá de distintas cosas:

- ¿Es posible que cambiemos de FrameWork?
- ¿Es posible que creemos otros interfaces (consola Windows o proceso nocturno) a la misma aplicación?
- ¿Cual es el potencial del utilización simultanea de uso de validaciones? ¿Y de reutilización en distintas aplicaciones de la misma validación?
- ¿Qué facilidades nos proporciona el FrameWork?

Con estas preguntas podríamos sacar unas conclusiones que se podrían transportar al resto de componentes de mi aplicación (podemos prototipar para determinar si las conclusiones son correctas antes de haber tirado demasiado código en paralelo).

Nosotros (sin entrar en más detalles) vamos a tomar las siguientes decisiones (que pueden ser más o menos acertadas):

La comprobación del país la realizamos en GestorVisitas (configurando una consulta sobre la clase Validador)

- Porque así no ligamos la capa de negocio a la de presentación y podríamos reaprovechar la validación si cambiamos la interfaz y decidimos que no sea Web.
- No la ponemos en la clase Visita porque en algún otro subsistema nos puede interesar volver a hacer la comprobación sin estar ligada la validación con la clase Visita

La eliminación de palabras conflictivas la hacemos en la capa de presentación

- Porque puede ser general para todos los parámetros de nuestra aplicación Web (esto es un potencial problema de seguridad) y el FrameWork nos lo facilita (aunque no vamos a profundizar en la resolución del problema)
- Porque en otros subsistemas (como en procesos batch nocturnos) podríamos requerir llamar a esta función de negocio y no es necesario realizar las comprobaciones de seguridad por cada llamada (el entorno no debería ser tan hostil)

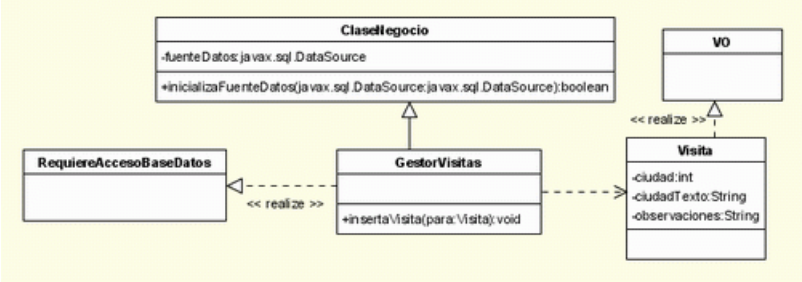
Otras consideraciones de diseño

La obtención de una conexión a la base de datos (en sistemas concurrentes) es un potencial riesgo en las aplicaciones Web (cuello de botella) y vamos a tratar de anticipar el problema. Suerte que ya hicimos otro tutorial donde analizamos como configurar un pool de conexiones a la base de datos con Struts.

Aún así tenemos que hacernos más preguntas:

- ¿La capa de negocio y presentación van a estar en la misma máquina? ¿Y en el fututo?
  - Si es así, es muy posible que queramos utilizar las facilidades del servidor Web de configuración y administración de pilas de conexiones.
  - Sino es así, es posible que mi aplicación sea más compleja y/o que tenga que obtener las conexiones de otros sitios (un servidor de aplicaciones o mi propio gestor)

Bueno, se ponga como se ponga, parece que hace falta que nuestras funciones de negocio no sean las encargadas de gestionar las conexiones a la base de datos. Si ellas no las gestionan, significa que les debe llegar algo que les permita acceder a la conexión a la base de datos cuando sea necesario (esto es una posible aplicación del patrón de diseño de inversión de responsabilidad). Además este algo debe ser independiente del tipo de aplicación que estemos construyendo (del tipo de interfaz de usuario).



De momento hemos pintado unos cuantos elementos que es posible que no entendamos:

- Nuestro **GestorVisitas** tendrá siempre una variable llamada **fuenteDatos** (inicialmente nula) que nos permitirá acceder a un conexión a la base de datos.
- El **GestorVisitas** implementa un interfaz marcador (sin métodos) que nos permitirá determinar en ejecución que esta clase requiere acceder a la base de datos
- La clase **Visita** implementa el interfaz VO (Value Object) que nos será útil cuando queramos que clases que representen valores sean tratadas de un modo homogéneo (por ejemplo caches y paginaciones)

(No estaría mal repasar un poquito los patrones de diseño generales y los J2EE)

Conclusiones del diseño

Todavía no hemos tirado una línea y ya tenemos claras bastantes cosas.

¿SORPRENDIDOS? Es que para esto vale DISEÑAR

Pensar así (aunque el diseño no sea muy avanzado) requiere formación y entrenamiento. Nno os pongáis nerviosos que todo llega (yo todos los días aprendo algo nuevo).

Construcción

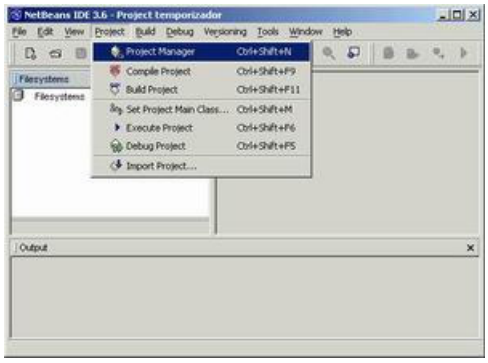
Jamás debemos anticiparnos a la hora de empezar a escribir código.

> ***Cuanto mejor esté definido lo que tenemos que hacer, menos nos costará hacerlo y menos frustrados nos encontraremos por retocar constantemente nuestro desarrollo.***

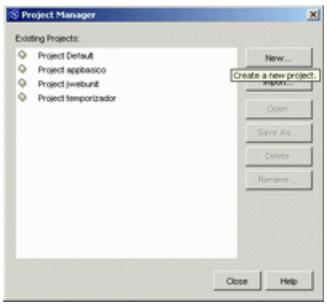
Instalación del entorno

Lo primero que hacemos es buscar una herramienta cómoda para construir. A mi me gusta NetBeans y aunque ya hay una primera versión 4, vamos a dejarla un poco que se estabilice antes de usarla (estar siempre a la última es asumir demasiado riesgo)

Construimos un nuevo proyecto (da igual la versión 3.5 o 3.6)

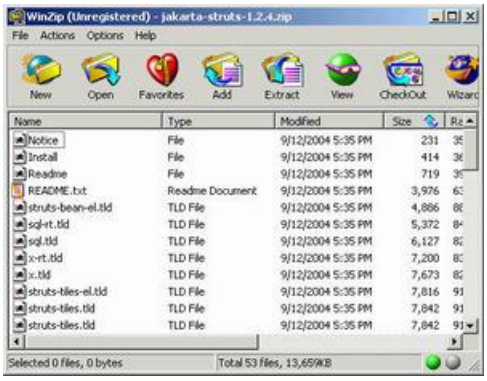


Lo creamos

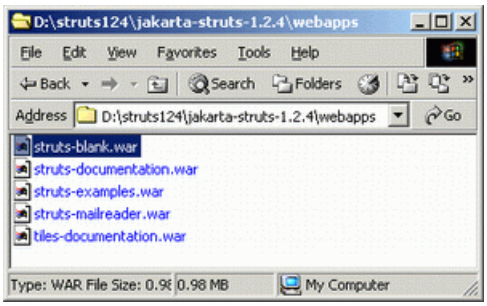


Le asignamos nombre

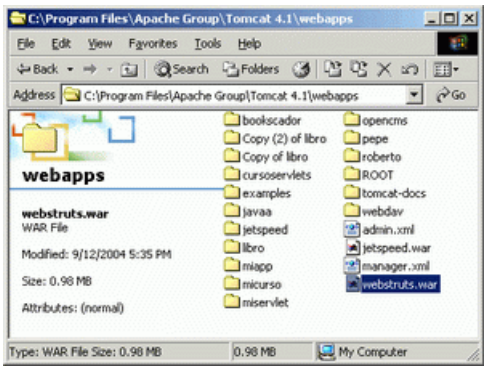
Nos descargamos la última versión de Struts (en este caso si utilizo la última versión porque me interesa comprobar su estado)



Descomprimimos el fichero y localizamos la aplicación Web ejemplo en blanco

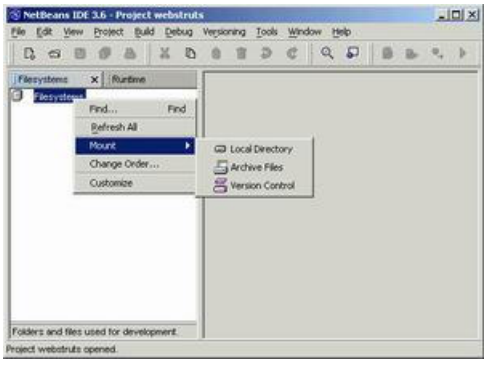


Situamos el fichero en nuestra instalación del TOMCAT

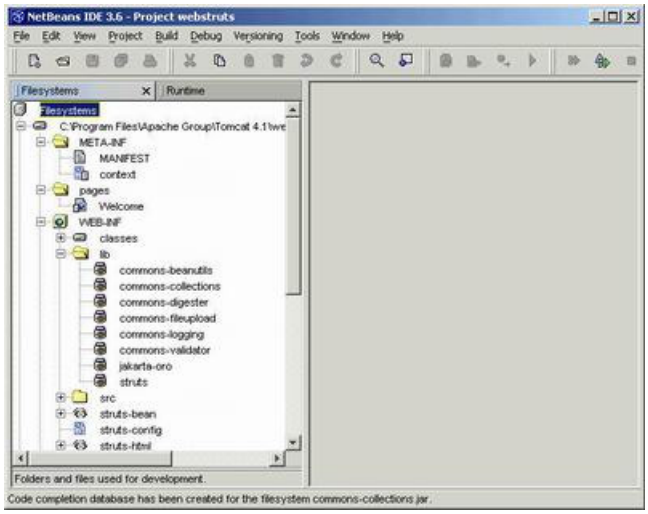


Arrancamos TOMCAT y automáticamente se descomprime el directorio y está ya lista la aplicación Web para ser utilizada.

Montamos en NetBeans el directorio descomprimido



Y ya tenemos todo lo necesario para empezar a construir nuestra aplicación



Arrancamos la aplicación para verificar que no hay ningún problema de compatibilidad

Decisión del punto donde empezar a construir

Ahora tenemos que empezar por algún sitio (esto es normalmente lo más difícil de decidir).

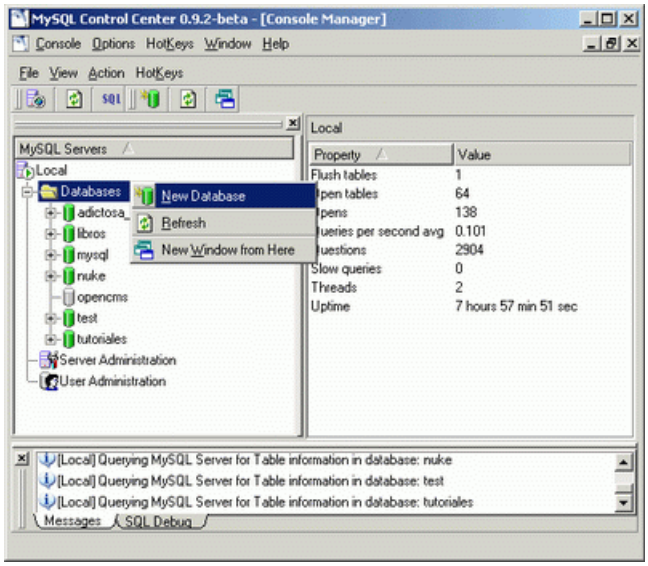
- Si empezamos por la parte de presentación, es muy posible que construyamos una aplicación demasiado vinculada al FrameWork y al problema particular (a corto plazo) por lo que tendremos poca capacidad de reutilización.
- Si empezamos por la capa de acceso a negocio, es muy posible que despistemos que algunas cosas que dependen de contexto, como por ejemplo como obtener una conexión a la base de datos.

Si utilizamos siempre el mismo FrameWork (sea el que sea), esta pregunta solo nos la tenemos que hacer solo una vez y aplicarla en la fase de diseño. Si estamos constantemente cambiando el modo de hacer las cosas, será difícil anticipar conclusiones. Si en una organización grande se deja esta decisión a criterio de cada equipo, sacar conclusiones globales es difícil y más aún extrapolar soluciones de problemas encontrados en producción.

Os recomiendo que empecéis siempre por la base de datos.... esto es realmente lo importante. Si en el futuro se pusiera de moda otro lenguaje y tenemos bien modelada la base de datos, la sustitución será menos traumática.

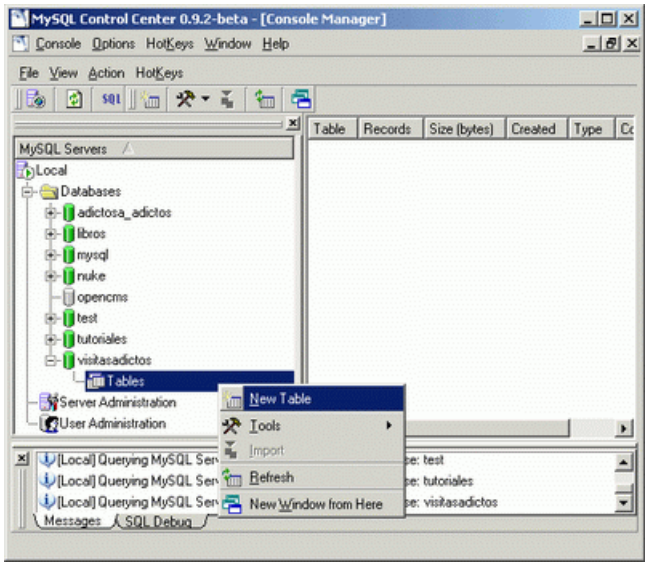
Diseño de la base de datos

Creamos una nueva base de datos con el panel de control de MySQL. Podéis ver como instalar MySQL y su consola en este otro tutorial.

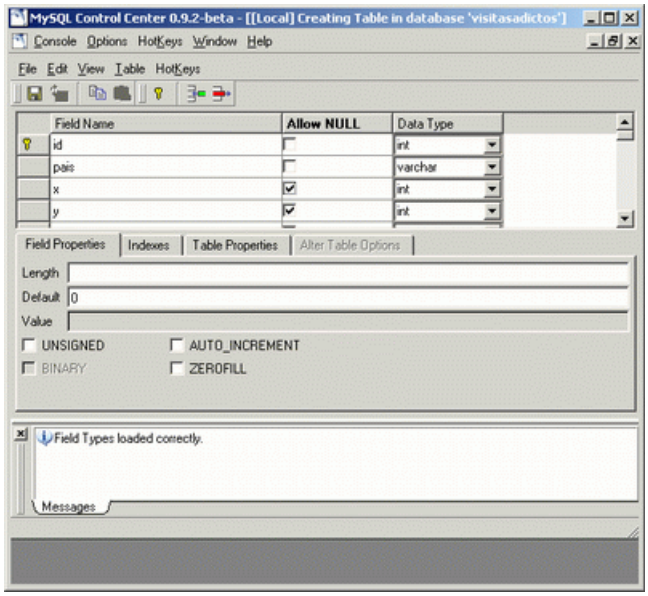


Le asignamos un nombre a la base de datos

Creamos las tablas

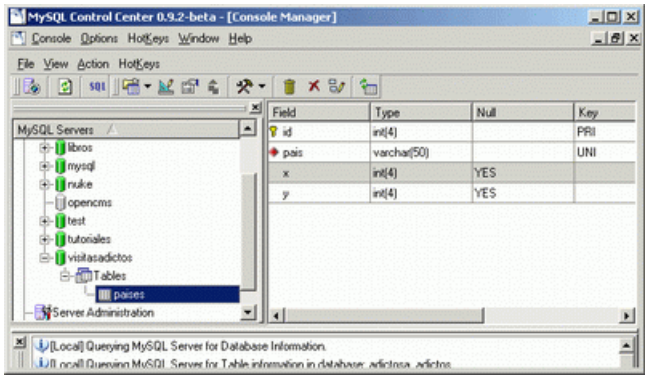


Definimos los campos de la primera tabla

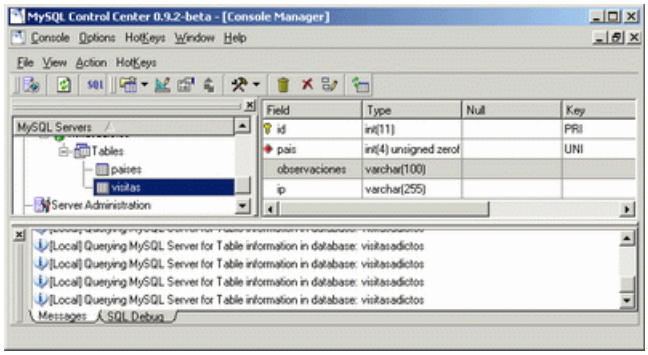


Le asignamos el nombre a la tabla

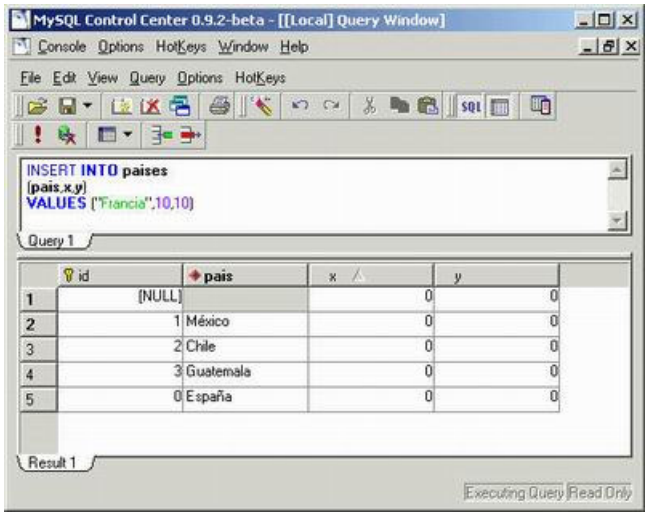
Y comprobamos el resultado



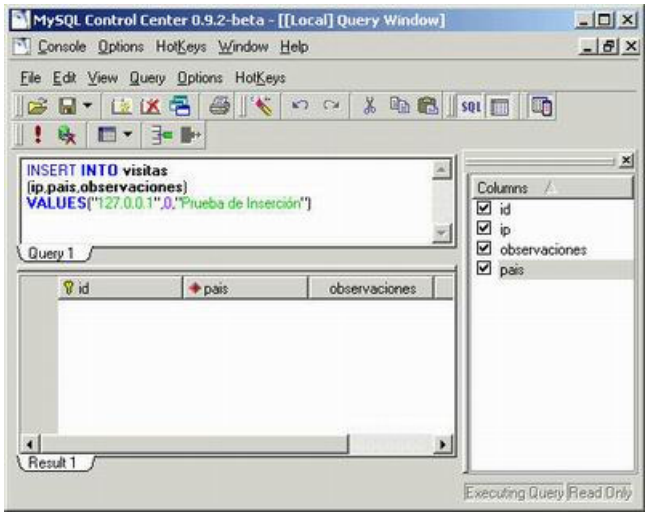
Y repetimos la operación con la tabla visitas (ojo que el país ahora no debe ser clave única)



Insertamos los primeros valores de prueba con lo que conseguimos descubrir las sentencias SQL que necesitaríamos en nuestros programas. Podéis ver en otro de nuestro tutoriales como generar el código Java para acceder a estas tablas sin saber demasiado.



Introducimos también algún valor en la tabla de países



Construcción de la aplicación

Lo primero que hacemos es construir el formulario de entrada de datos.

En un formulario pueden ocurrir distintos problemas:

1. Que el usuario por equivocación haga un doble click e introduzca dos veces los datos
2. Que inserte valores incorrectos y necesitemos volver a mostrar los campos y destacar de algún modo el campo incorrecto.

Utilizaremos un JSP con las etiquetas particulares que proporciona Struts para resolver los problemas anteriormente mencionados (aunque con el primero hay que hacer algo más).

Tenemos que entender como funciona un poquito el FrameWork Struts

1. Un usuario hace una petición
2. Cada petición está asociada a una acción que contiene la regla de negocio particular
3. La acción suele necesitar un conjunto de datos que se encapsulan en un objeto llamado Form (este Form puede ser una clase física o un elemento general cuyo comportamiento se define en el fichero de configuración, también llamado DynaForm)
4. Los datos se pueden validar y si falla la validación se puede volver a mostrar el formulario de entrada.
5. Una vez ejecutada la acción, puede que el resultado sea favorable o desfavorable por lo que se delega sobre distintos elementos de presentación en cada caso.
6. Todo mensaje mostrado puede sacarse a un fichero de recursos para favorecer su mantenimiento y la internacionalización de las aplicaciones (aunque ahora no nos haga falta).
7. Cada elemento se construye por separado y se relacionan en el fichero struts-config.xml

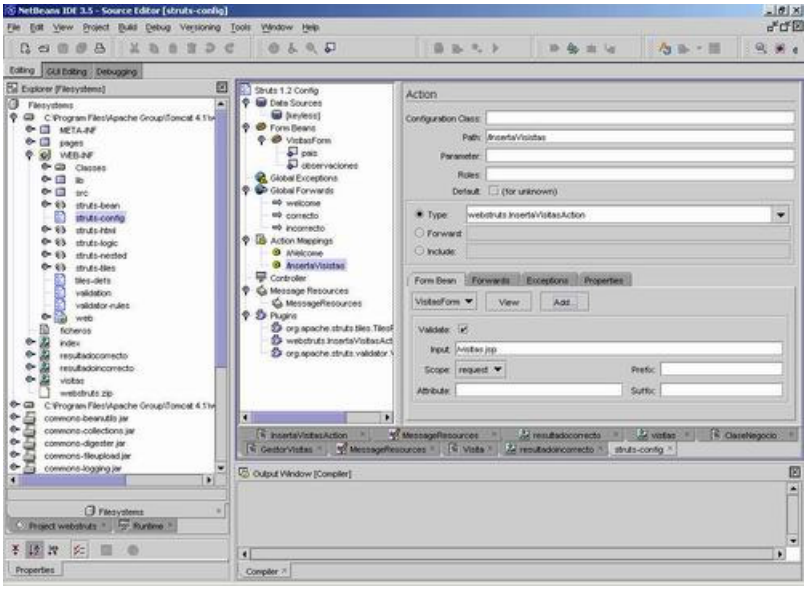
Ya se que no es fácil la primera vez (y menos aún identificar los errores según aparecen). Vamos a construir únicamente el código que es particular del interfaz que hemos elegido y marcaremos en el código los puntos en los que hace falta acceder a las reglas de negocio y los datos (que eso ya es Java puro y duro)

La estructura

Aunque la aplicación no es plenamente funcional con esta estructura de ficheros seréis capaz de reproducir el ejemplo. En rojo hemos marcado los únicos ficheros que hemos cambiado o creado. Debéis tener precaución con los ficheros en el directorio lib (ver el tutorial sobre como configura el pool de conexiones a la base de datos con struts). No os preocupéis de momento en la posición de los ficheros creados (eso es un problema menor a estas alturas aunque en el futuro si nos deberá preocupar)

```
*   index.jsp
*   resultadocorrecto.jsp
*   visitas.jsp
*
****META-INF
*       context.xml
*       MANIFEST.MF
*
****pages
*       Welcome.jsp
*
****WEB-INF
    *   struts-bean.tld
    *   struts-config.xml
    *   struts-html.tld
    *   struts-logic.tld
    *   struts-nested.tld
    *   struts-tiles.tld
    *   tiles-defs.xml
    *   validation.xml
    *   validator-rules.xml
    *   web.xml
    *
    ****classes
    *   *   MessageResources.properties
    *   *
    *   ****resources
    *   *       MessageResources.properties
    *   *
    *   ****webstruts
    *           InsertaVisitasAction.class
    *           InsertaVisitasAction.java
    *
    ****lib
    *       commons-beanutils.jar
    *       commons-collections.jar
    *       commons-dbcp-1.1.jar
    *       commons-digester.jar
    *       commons-fileupload.jar
    *       commons-lang.jar
    *       commons-logging.jar
    *       commons-pool-1.1.jar
    *       commons-validator.jar
    *       jakarta-oro.jar
    *       jdbc2_0-stdext.jar
    *       jstl.jar
    *       mysql-connector-java-3.0.8-stable-bin.jar
    *       standard.jar
    *       struts-legacy.jar
    *       struts.jar
    *
    ****src
        *   build.xml
        *
        ****java
            ****resources
                application.properties
```

A mi me gusta tocar los ficheros de configuración a mano, sobre todo al principio, pero no olvidéis que hay herramientas gráficas (consolas para struts) para poder hacerlo (incluso se integran fácilmente con nuestro editor)



El fichero de configuración * struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">
<struts-config>

<!-- ============================================ Data Source Configuration -->

<data-sources>
<data-source type="org.apache.commons.dbcp.BasicDataSource">
<set-property property="driverClassName" value="com.mysql.jdbc.Driver" />
<set-property property="url" value="jdbc:mysql://localhost/visitasadictos" />
<set-property property="username" value="root" />
<set-property property="password" value="" />
<set-property property="maxActive" value="10" />
<set-property property="maxWait" value="5000" />
<set-property property="defaultAutoCommit" value="false" />
<set-property property="defaultReadOnly" value="false" />
<set-property property="validationQuery" value="SELECT COUNT(*) FROM paises" />
</data-source>
</data-sources>

<!-- ============================================ Form Bean Definitions -->

<form-beans>
<form-bean
name="VisitasForm"
type="org.apache.struts.action.DynaActionForm">
<form-property name="pais" type="java.lang.Integer"/>
<form-property name="observaciones" type="java.lang.String"/>
</form-bean>
</form-beans>

<!-- ============================================ Global Exception Definitions -->

<global-exceptions>
<!-- sample exception handler <exception
key="expired.password" type="app.ExpiredPasswordException" path="/changePassword.jsp"/>
end sample -->
</global-exceptions>

<!-- ============================================ Global Forward Definitions -->

<global-forwards>
<!-- Default forward to "Welcome" action -->
<!-- Demonstrates using index.jsp to forward -->
<forward name="welcome" path="/Welcome.do"/>
<forward name="correcto" path="/resultadocorrecto.jsp"/>
</global-forwards>

<!-- ============================================ Action Mapping Definitions -->

<action-mappings>
<!-- Default "Welcome" action -->
<!-- Forwards to Welcome.jsp -->
<action path="/Welcome" forward="/pages/Welcome.jsp"/>

<action path="/InsertaVisistas"
type="webstruts.InsertaVisitasAction"
name="VisitasForm"
scope="request"
validate="true"
input="/visitas.jsp"/>
</action-mappings>

<!-- ============================================ Controller Configuration -->

<controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

<!-- ============================================ Message Resources Definitions -->

<message-resources parameter="MessageResources" />

<!-- ============================================ Plug Ins Configuration -->

<plug-in className="org.apache.struts.tiles.TilesPlugin" >

<!-- Path to XML definition file -->
<set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />
<!-- Set Module-awareness to true -->
<set-property property="moduleAware" value="true" />
</plug-in>

<plug-in className="webstruts.InsertaVisitasAction" >
</plug-in>

<!-- ============================================ Validator plugin -->

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
<set-property property="pathnames"
value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```

La acción * InsertaVisitasAction.java

```
/*
 * InsertaVisitasAction.java
 *
 * Created on October 31, 2004, 12:27 PM
 *
 * Roberto Canales Mora
 */

package webstruts;

import org.apache.struts.action.*;
import javax.servlet.http.*;
import java.util.*;

public class InsertaVisitasAction extends Action implements org.apache.struts.action.PlugIn {

   public ActionForward execute(ActionMapping actionMapping,ActionForm actionForm,

                                HttpServletRequest httpServletRequest,

                                HttpServletResponse httpServletResponse) throws Exception {

      ActionForward retValue = null;
      try // ejecutamos la funcion de negocio
      {
         // Obtenemos acceso al formulario
         DynaActionForm formulario = (DynaActionForm) actionForm;

         // aqui va la inserción de los datos
         System.out.println("El pais es: " + formulario.get("pais"));
         System.out.println("Observaciones: " + formulario.get("observaciones"));

         // redirigimos a la presentación
         retValue = actionMapping.findForward("correcto");
      }
      catch(Exception e) // en caso de problemas retornar a la página origen
      {
          // recuperamos acceso al JSP deorigen
          retValue = actionMapping.getInputForward();

          // damos de alta los errores
          ActionErrors errores = new ActionErrors();
          errores.add(errores.GLOBAL_ERROR, new ActionError("error.logica"));
          errores.add("pais", new ActionError("error.enpais"));
          this.addErrors(httpServletRequest, errores);
      }

      return retValue; // redirigimos a la presentacion
   }

   /*
    * Los métodos destroy e init son obligatorios al implementar el interfaz PlugIn
     */
   public void destroy() {
   }

   public void init(org.apache.struts.action.ActionServlet actionServlet,

org.apache.struts.config.ModuleConfig moduleConfig) throws javax.servlet.ServletException {

      Hashtable vPaises = new Hashtable();

      // aqui hay que poner el código que nos proporcione los datos de la base de datos
      vPaises.put("1","España"); // este codigo hay que quitarlo
      vPaises.put("2","Chile");

      // mostramos un mensaje para asegurarnos que los datos están en memoria
      System.out.println("Tenemos el Vector");

      // ponemos la tabla en el ambito de la aplicación
      actionServlet.getServletContext().setAttribute("tablapaises",vPaises);
   }
}
```

Esta acción tiene una peculiaridad y es que implementa en interfaz plugIn (métodos destroy e init) que nos permite que se ejecute código nada mas inicializarse el componente. La tabla de países no creo que cambie demasiado a menudo por lo que nos puede interesar recuperarla solo una vez para todos los usuarios del sistema.

El JSP de entrada de datos * visitas.jsp

```
 <%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>

<html>
<head>
<title><bean:message key="index.titulo"/></title>
</head>

<body>
<center>
```

```
<h2><bean:message key="index.cuentanostuorigen"/></h2>
<hr width='60%'>

<html:form action='/InsertaVisistas'>
<table border="0">
<tr>

<td>Pais</td><td>

<html:select property="pais" >
<html:options collection="tablapaises" property="key" labelProperty="value" />
</html:select>


</td><td><html:errors property="pais" /> </td>
</tr>
<tr>
<td><!-- podría ser una clave en el fichero de recursos -->
Observaciones</td><td> <html:textarea property="observaciones" cols="50" rows="3" />

</td><td><html:errors property="observaciones" /></td>
</tr>
<tr><td> <html:submit value="Enviar"/></td> </tr>

</table>
</html:form>

<br/> <font color="#FF0000"> <html:errors/> </font>
</center>
</body>
</html>
```

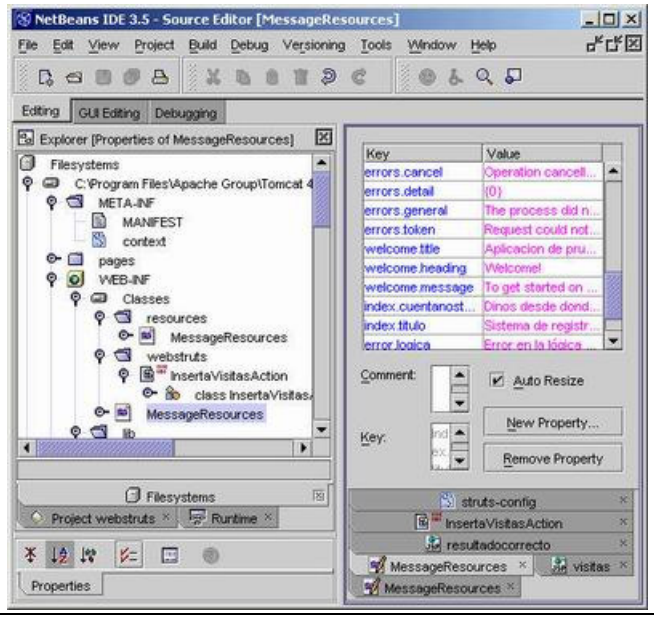
Analizamos este código con detenimiento porque tiene muchas más cosas de las que puede parecer en un principio.

¿De donde salen los valores del formulario? De los datos que ha puesto el plugIn en la aplicación

```
<html:select property="pais" >
<html:options collection="tablapaises" property="key" labelProperty="value" />
</html:select>
```

¿De donde salen los textos de la página que no están directamente escritos? Del fichero de recursos

```
<bean:message key="index.titulo"/>
```



¿Para que vale esta línea? Para mostrar errores cuando se produzcan problemas en validaciones

```
<html:errors property="pais" />
```

Debemos buscar en la acción este código para entenderlo correctamente

```
errores.add("pais", new ActionError("error.enpais"));
```

JSP resultante * resultadocorrecto.jsp

```
<%@page contentType="text/html"%>
<html>
<head><title>JSP Page</title></head>
<body>

<h2>La acción se ejecutó satisfactoriamente</h2>

</body>
</html>
```

Añadir el código de acceso a datos

Ahora que ya tenemos el código particular de Struts y nos queda el código de acceso a la base de datos. Esto es Java puro y duro.

Creamos el resto de las clases (aunque el código no es definitivo ya que el tratamiento de errores de momento es pésimo)

Visita.java

```
package webstruts;

/*
* Visita.java
* Created on October 31, 2004, 6:17 PM
* @author Roberto Canales
*/
public class Visita {
private int id;
private int pais;
private String observaciones;
private String ip;

/** Creates a new instance of Visita */
public Visita() {}

/** Creates a new instance of Visita */
public Visita(int pais, String observaciones,String ip) {
this.pais = pais;
this.observaciones = observaciones;
this.ip = ip;
}
/** Getter for property id.
* @return Value of property id.
*/
public int getId() {return id;}

/** Setter for property id.
* @param id New value of property id.
*/
```

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

/** Getter for property observaciones.
* @return Value of property observaciones.
*/
public java.lang.String getObservaciones() {return observaciones;}

/** Setter for property observaciones.
* @param observaciones New value of property observaciones.
*/
public void setObservaciones(java.lang.String observaciones) {
this.observaciones = observaciones;}

/** Getter for property pais.
* @return Value of property pais.
*/
public int getPais() {return pais;}

/** Setter for property pais.
* @param pais New value of property pais.
*/
public void setPais(int pais) {this.pais = pais;}

/** Getter for property ip.
* @return Value of property ip.
*/
public java.lang.String getIp() {return ip;}

/** Setter for property ip.
* @param ip New value of property ip.
*/
public void setIp(java.lang.String ip) {this.ip = ip;}
}
```

GestorVisitas.java

```
/*
 * GestorVisitas.java
 *
 * Created on October 31, 2004, 6:21 PM
 */

package webstruts;

import javax.sql.*;
import java.sql.*;

/**
 *
 * @author  Roberto Canales Mora
 */
public class GestorVisitas extends ClaseNegocio {

    /** Creates a new instance of GestorVisitas */
    public GestorVisitas() {
    }

    void depura(String mensaje)
    {
        System.out.println("GestorVisitas - " + mensaje);
    }


    public boolean insertaVisita(Visita pVisita) {

        try
        {
            Connection con =  this.dameConexion();
            PreparedStatement pstmt =

con.prepareStatement("insert into visitas (pais,observaciones,ip) values (?,?,?)");

            // establecemos los valores variables
            pstmt.setInt(1,pVisita.getPais()); // establecemos el entero
            pstmt.setString(2,pVisita.getObservaciones()); // establecemos el entero
            pstmt.setString(3,pVisita.getIp()); // establecemos el entero

            // ejecutamos la consulta
            int resultado = pstmt.executeUpdate();

            depura("El número de elementos afectados es " + resultado);

            // retornamos la conexion al pool
            con.close();
            return true;
        }
        catch (SQLException e)
        {
           depura("Error al insertar " + e.getMessage());
           return false;
        }
    }
}
```

ClaseNegocio.java

```
/*
 * ClaseNegocio.java
 * Created on October 31, 2004, 6:20 PM
 */

package webstruts;

import javax.sql.*;
import java.sql.*;

/**
 *
 * @author Roberto Canales Mora
 */
public class ClaseNegocio {

    private javax.sql.DataSource fuenteDatos = null;

    /** Creates a new instance of ClaseNegocio */
    public ClaseNegocio() {
    }

    public boolean inicializaFuenteDatos(javax.sql.DataSource pFuenteDatos) {
        fuenteDatos = pFuenteDatos;
        return true;
    }

    public java.sql.Connection dameConexion() {

        try
        {
            return fuenteDatos.getConnection();
        }
        catch(Exception e)
        {

        }
        return null;
    }
}
```

InsertaVisistasAction.java (todavía hay que recuperar los países de la base de datos pero no queremos mezcla demasiadas cosas)

```
/*
 * InsertaVisitasAction.java
 *
 * Created on October 31, 2004, 12:27 PM
 *
 * Roberto Canales Mora
 */

package webstruts;

import org.apache.struts.action.*;
import javax.servlet.http.*;
import java.util.*;

public class InsertaVisitasAction extends Action implements org.apache.struts.action.PlugIn {

    public ActionForward execute(ActionMapping actionMapping,

                                 ActionForm actionForm,HttpServletRequest httpServletRequest,

                                 HttpServletResponse httpServletResponse) throws Exception {

        ActionForward retValue = null;
        try // ejecutamos la funcion de negocio
        {
            // Obtenemos acceso al formulario
            DynaActionForm formulario = (DynaActionForm) actionForm;

            // Construimos el objeto de negocio
            GestorVisitas gestorVisitas = new GestorVisitas();

            // le damos acceso a la fuente de datos
            gestorVisitas.inicializaFuenteDatos(this.getDataSource(httpServletRequest));

            // recuperamos el pais y observaciones
            Integer pais = (Integer)formulario.get("pais");
            String observaciones = formulario.get("observaciones").toString();

            // llamamos al metodo de negocio
            boolean resultado = gestorVisitas.insertaVisita(new Visita(pais.intValue(),
                                        observaciones,
                                        httpServletRequest.getRemoteAddr()));

            // redirigimos a la presentación
            if(resultado == true)
                retValue = actionMapping.findForward("correcto");
            else
                retValue = actionMapping.findForward("incorrecto");
```

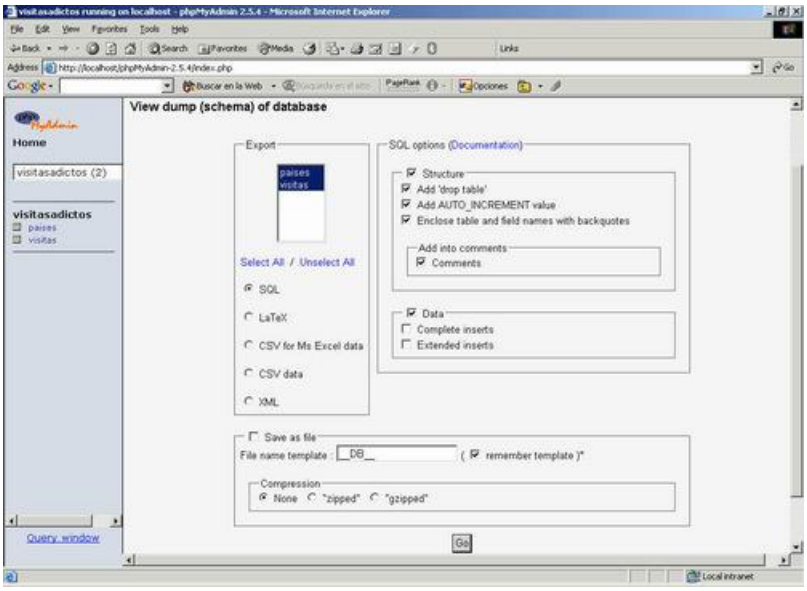
```
        }
        catch(Exception e) // en caso de problemas retornar a la página origen
        {
            // recuperamos acceso al JSP deorigen
            retValue = actionMapping.getInputForward();

            // damos de alta los errores
            ActionErrors errores = new ActionErrors();
            errores.add(errores.GLOBAL_ERROR, new ActionError("error.logica"));
            errores.add("pais", new ActionError("error.enpais"));
            this.addErrors(httpServletRequest, errores);
        }

        return retValue; // redirigimos a la presentacion
    }

    /*
     * Los métodos destroy e init son obligatorios al implementar el interfaz PlugIn
      */
    public void destroy() {
    }

    public void init(org.apache.struts.action.ActionServlet actionServlet,

                                        org.apache.struts.config.ModuleConfig moduleConfig)

                                        throws javax.servlet.ServletException {

        Hashtable vPaises = new Hashtable();

        // aqui hay que poner el código que nos proporcione los datos de la base de datos
        vPaises.put("1","España"); // este codigo hay que quitarlo
        vPaises.put("2","Chile");

        // mostramos un mensaje para asegurarnos que los datos están en memoria
        System.out.println("Tenemos el Vector");

        // ponemos la tabla en el ambito de la aplicación
        actionServlet.getServletContext().setAttribute("tablapaises",vPaises);
    }
}
```

Obtención de los datos

Para que el ejemplo sea completamente utilizable necesitamos las tablas.

Vamos a recuperar la estructura de la tabla valiéndonos de phpMyAdmin (recordamos otro de nuestro tutoriales)

Elegimos las tablas y el modo de extracción.



Y nos genera algo tal que esto

```
#
# Table structure for table `paises`
#
DROP TABLE IF EXISTS `paises`;
CREATE TABLE `paises` (
`id` int(4) NOT NULL auto_increment,`pais` varchar(50) NOT NULL default '',
`x` int(4) default '0',`y` int(4) default '0',
PRIMARY KEY (`id`),UNIQUE KEY `paisesindex` (`pais`)
) TYPE=MyISAM AUTO_INCREMENT=5 ;
#
# Dumping data for table `paises`
#
INSERT INTO `paises` VALUES (1, 'España', 0, 0);
INSERT INTO `paises` VALUES (2, 'Chile', 0, 0);
INSERT INTO `paises` VALUES (3, 'Guatemala', 0, 0);
INSERT INTO `paises` VALUES (4, 'Francia', 10, 10);
# --------------------------------------------------------
#
# Table structure for table `visitas`
#
DROP TABLE IF EXISTS `visitas`;
CREATE TABLE `visitas` (
`id` int(11) NOT NULL auto_increment,
`pais` int(4) unsigned zerofill NOT NULL default '0000',
`observaciones` varchar(100) NOT NULL default '12',
`ip` varchar(255) NOT NULL default '',
PRIMARY KEY (`id`), KEY `indexpais` (`pais`)
) TYPE=MyISAM AUTO_INCREMENT=5 ;
#
# Dumping data for table `visitas`
#
INSERT INTO `visitas` VALUES (1, 0002, 'comentario1', '127.0.0.1');
INSERT INTO `visitas` VALUES (2, 0001, 'comentario2', '127.0.0.1');
INSERT INTO `visitas` VALUES (3, 0001, 'comentario3', '127.0.0.1');
INSERT INTO `visitas` VALUES (4, 0002, 'comentario4', '127.0.0.1');
```

Con esto ya tenemos todo lo necesario para orientar el programa.

Conclusiones

Si tienes las ideas medianamente claras y eres capaz de descomponer un problema grande en problemas pequeñitos, no es tan difícil construir una aplicación.

Parece que hemos invertido demasiado esfuerzo para algo aparentemente trivial y que podríamos haber resuelto prácticamente con un par de JSPs o Servlets pero no nos engañemos: Todas las aplicaciones tiende a complicarse y si ha hemos constuido así será fácil ampliarla por cualquier sitio.

Aunque parezca contradictorio con el párrafo anterior, francamente creo que tendemos a tecnificar demasiado los problemas y nos dejamos llevar por las modas; No en todos los casos es necesario solucionar los problemas de un modo tan flexible (esto implica un coste inicial muy alto que puede hacer inviable un plan de negocio).

Struts es un buen posible punto de partida pero tampoco nos tenemos que equivocar:

- Es solo un modo más de plantear el Front-End (interfaz con el cliente) de una aplicación y hay 200 modos más.
- Requiere un entrenamiento y aprendizaje que se justificará si lo utilizamos habitualmente o queremos construir sistemas altamente robustos.
- Muchas empresas crean su propios FrameWorks y puedes llevarte sorpresas si te acostumbras solo a utilizar Struts
- Algunas empresas tienen desaconsejado el uso de Struts (normalmente porque han tenido malas experiencias con consultoras que han construido soluciones muy acopladas por entender mal el producto)
- El mantenimiento es enrevesado y puede despistar a equipos poco cualificados

- Es necesario ampliarlo para resolver problemas en entornos profesionales

---

Si desea contratar formación, consultoria o desarrollo de piezas a medida puede contactar con

Formación en nuevas tecnologías

Somos expertos en:
J2EE, C++, OOP, UML, Vignette, Creatividad ..
y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Comunicación entre TAGs, Beans y JSPs	Os mostramos las posibilidades de comunicación entre JSPs, Bean y etiquetas de usuario.
Primeras aplicaciones con Bea Weblogic Platform	Os mostramos como instalar Bea Weblogic Platform así como a crear la primera aplicación, con su entorno visual, utilizando la implementación particular basada en Struts....
Desarrollo Struts con XDoclet	Alejandro Perez nos enseña como simplificar el desarrollo de aplicaciones J2EE basadas en Struts, automatizando la generación de código con XDoclet
Framework desarrollo eclipse	Aquí os mostramos algunas de las características de Eclipse
Aplicación de Patrones de Diseño en Java	En este tutorial os mostramos como las técnicas avanzadas de diseño (como patrones de diseño) contribuyen a la contrucción de aplicaciones profesionales en Java.
Introducción al UML	Este es el primer articulo sobre el diseño de proyectos orientados a objeto con UML, donde se describe los primeros diagramas a utilizar
Patrones de diseño J2EE	Os mostramos una interpretación particular de los patrones de diseño J2EE
Gestión de contenidos y errores comunes	Os explicamos en que consiste la gestión de contenidos y cuales son los errores cometidos por multitud de empresas a la hora de abordar su implantación
Struts Jakarta	Cuando se ha trabajado creando aplicaciones Java poco a poco se va viendo la necesidad de normalizar los desarrollo. Uno de los Framework (entornos) más extendidos es Struts