Aplicaciones Web con JSPs y Servlets en NetBeans

Creación de un Proyecto

Para crear una aplicación Web usando JSPs en **NetBeans** lo primero que hay que hacer es crear un proyecto. Un proyecto nos permite administrar los archivos con el código fuente y compilado de una aplicación. Para crear un proyecto se sigue el siguiente procedimiento:

1. Ejecute el programa **NetBeans**. Al hacerlo aparecerá la ventana principal del programa como se ilustra en la figura 1.

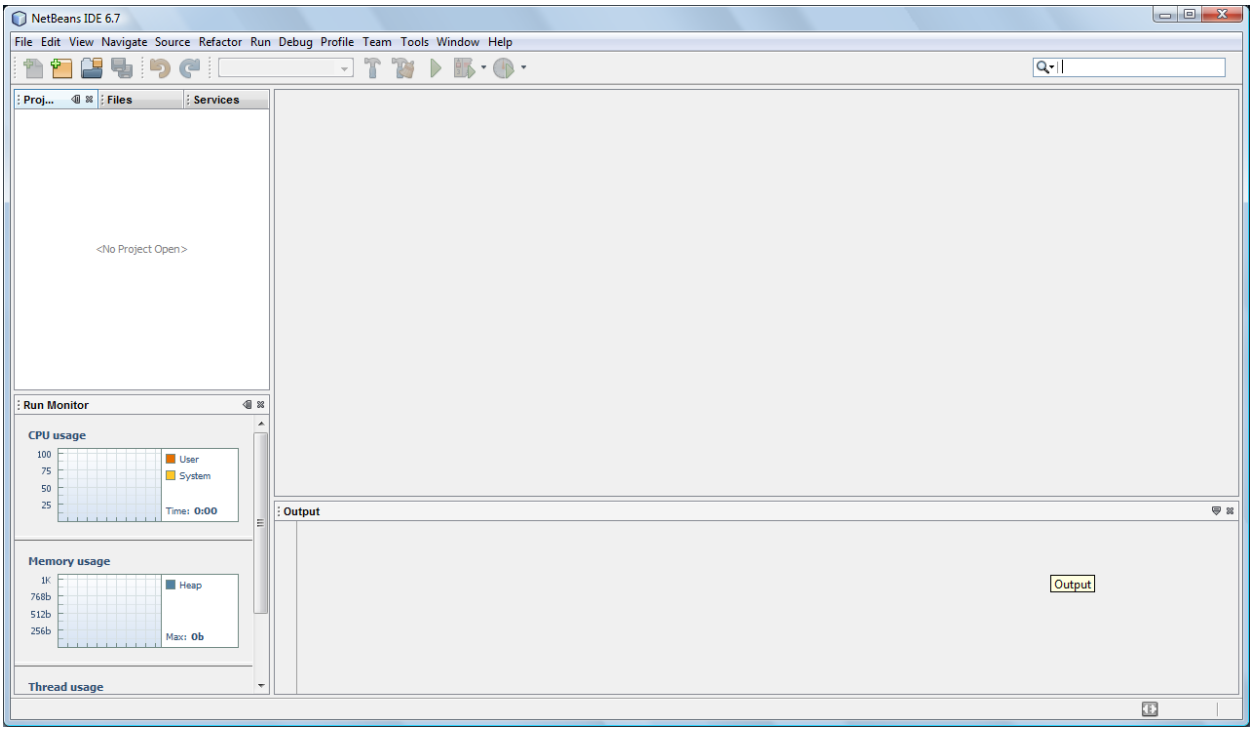

Figura 1

2. Del menú principal de NetBeans, figura 1.1, seleccione la opción **File/New Project …** , presione las teclas **Ctrl+Mayúsculas+N** o haga clic en el icono **New Project** mostrado en la figura 2.

3. Aparecerá el primer cuadro de diálogo del asistente para crear un nuevo proyecto, figura 3.

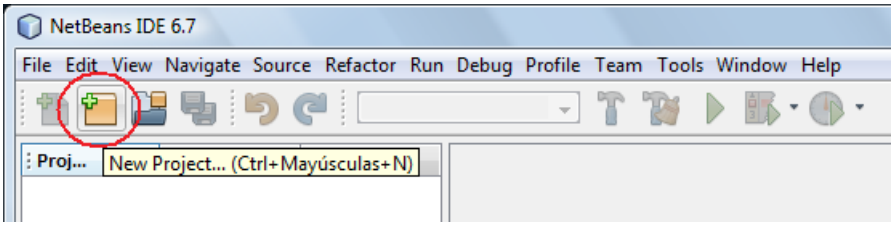

Figura 2

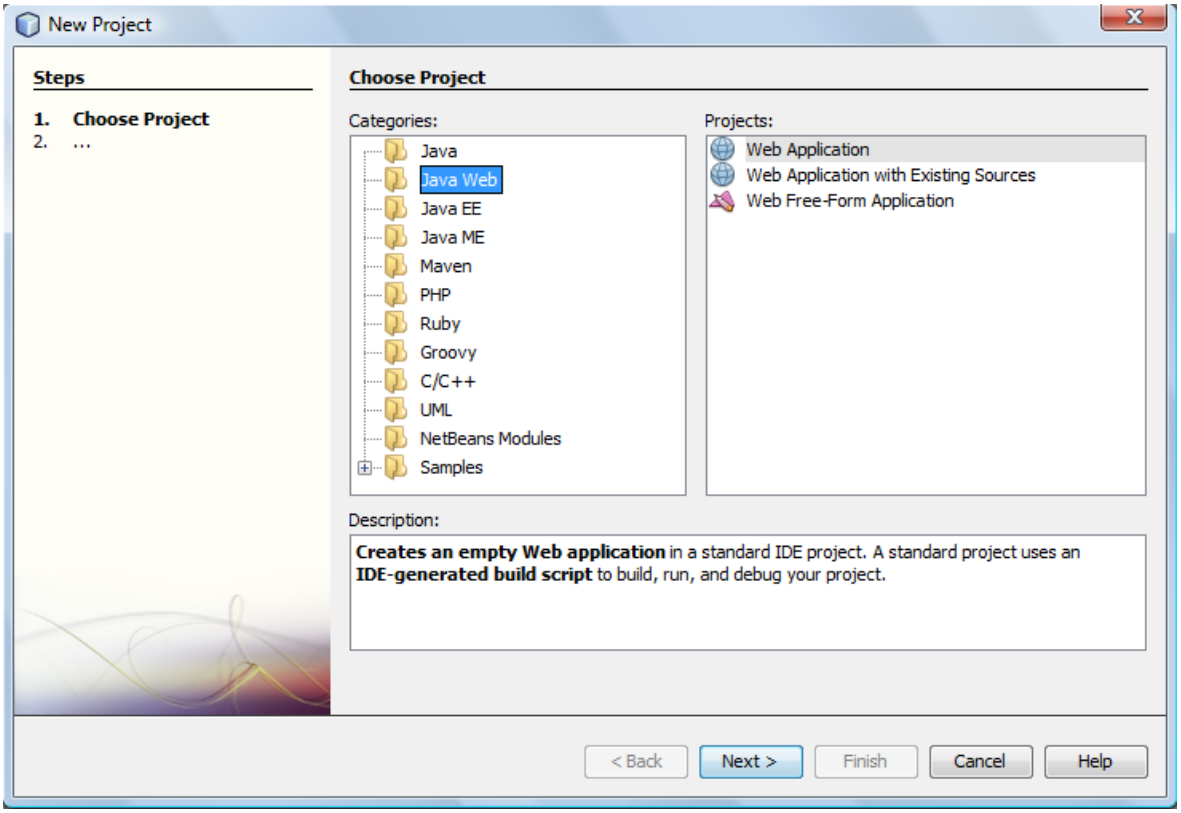

Figura 3

4. En este cuadro de diálogo del asistente seleccionaremos el tipo de proyecto que deseamos crear. Como vamos a crear una aplicación Web, seleccionaremos la opción **Web** en el recuadro **Categories:** y la opción **Web Application** en el recuadro **Projets:**, y luego presionaremos el botón **Next>**.

5. Aparecerá el segundo cuadro de diálogo del asistente para crear proyectos, mostrada en la figura 4. En este cuadro seleccionaremos el nombre y la ubicación del proyecto.

   a) En el campo de texto **Project Name**, establezca el nombre del proyecto. Por ejemplo, "**amanteMusicaWeb**".

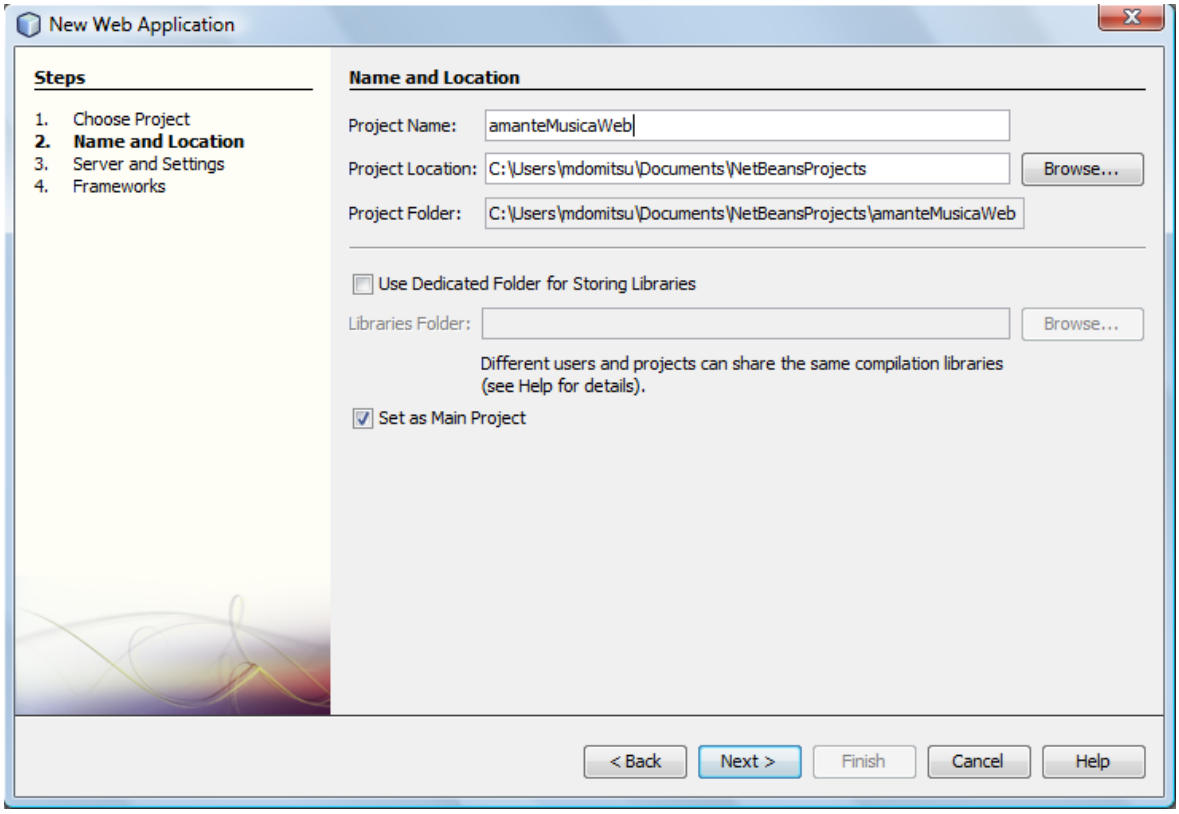

Figura 4

b) En el campo de texto **Projet Location**, establezca el directorio donde se almacenará el proyecto. En este caso se dejo el directorio por ausencia: "**C:\Users\mdomitsu\Documents\NetBeansProjects**". En el campo de texto siguiente, **Project Folder**, puede verse el nombre del directorio en el que se almacenarán los archivos del proyecto. En este ejemplo es: "**C:\Users\mdomitsu\Documents\NetBeansProjects\amanteMusicaWeb**".
c) Deje la casilla de verificación **Use Dedicated Folder for Storing Libraries** sin seleccionar.
d) Asegúrese que la casilla de verificación **Set as Main Project** esté seleccionada.
e) Presione el botón **Next>**.

6. Aparecerá el tercer cuadro de diálogo del asistente para crear proyectos, figura 5. Este cuadro nos permite establecer el servidor Web o servidor de aplicaciones en el que se desplegará la aplicación, la versión del JDK a emplear y el nombre de contexto de la aplicación (la liga dentro del servidor web o de aplicaciones que tiene la aplicación). En este caso se dejarán los valores predefinidos. Presione el botón **Next>**.

7. En el cuarto cuadro de diálogo del asistente para crear proyectos, figura 6, se puede seleccionar el marco de trabajo deseado para implementar la aplicación Web. Como no se utilizará ninguno, deje las casillas de verificación sin seleccionar y presione el botón **Finish**.

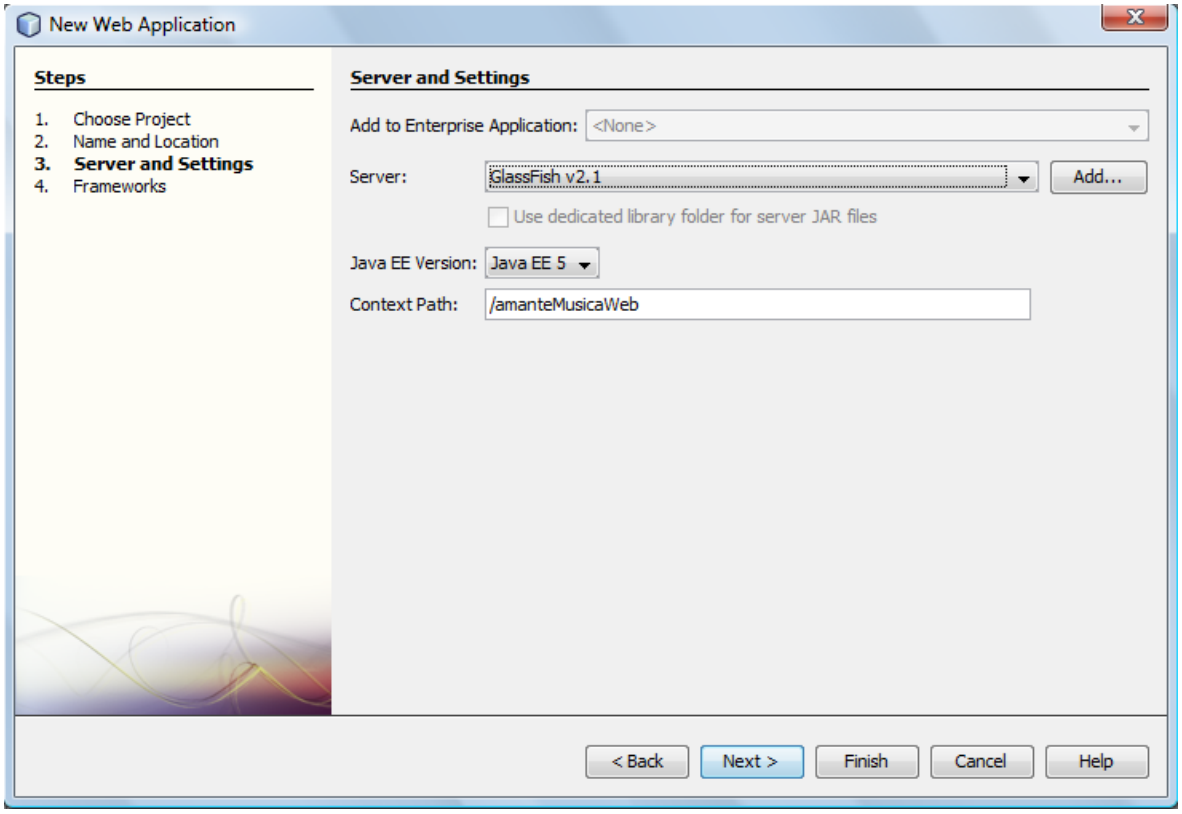

Figura 5

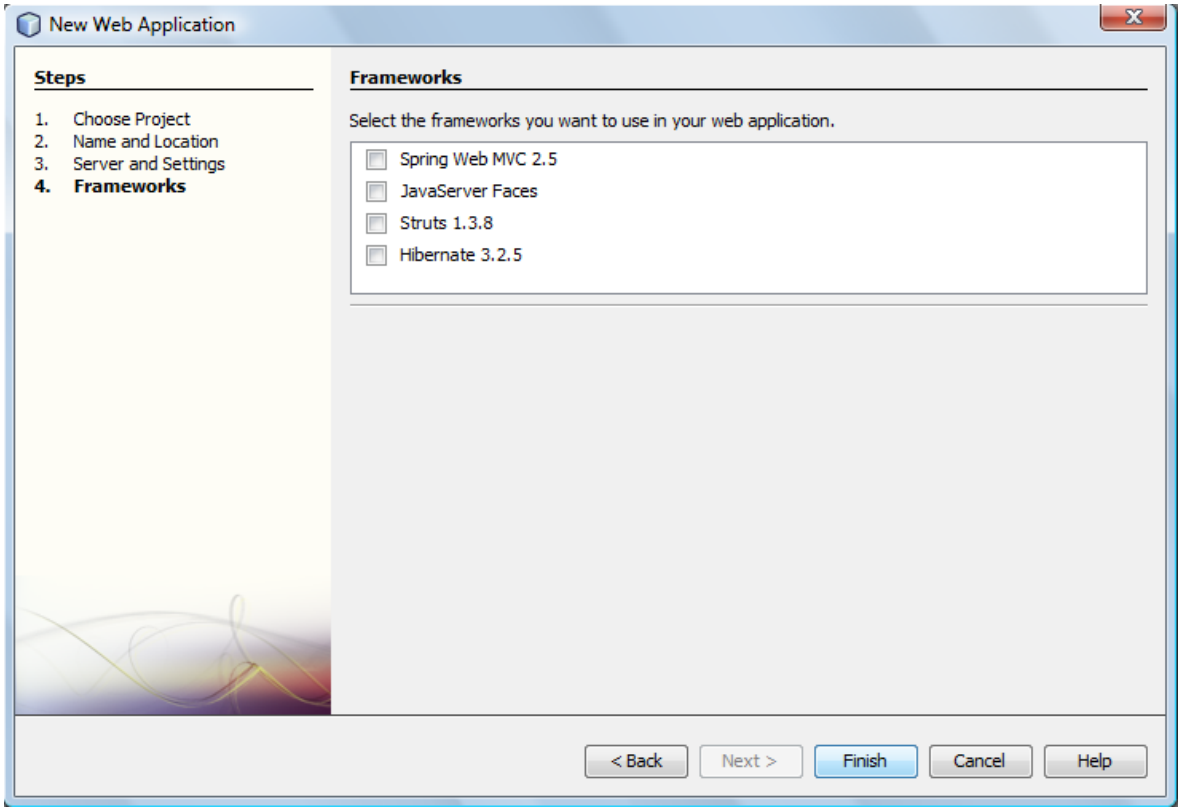

Figura 6

8. Desaparecerá el asistente para crear un nuevo proyecto y aparecerá lo mostrado en la figura 7. Del lado derecho aparece el editor de NetBeans con el esqueleto de la página JSP inicial: **index.jsp**, mientras que del lado izquierdo aparece el árbol de los proyectos, en el que aparece el proyecto **amanteMusicaWeb**.

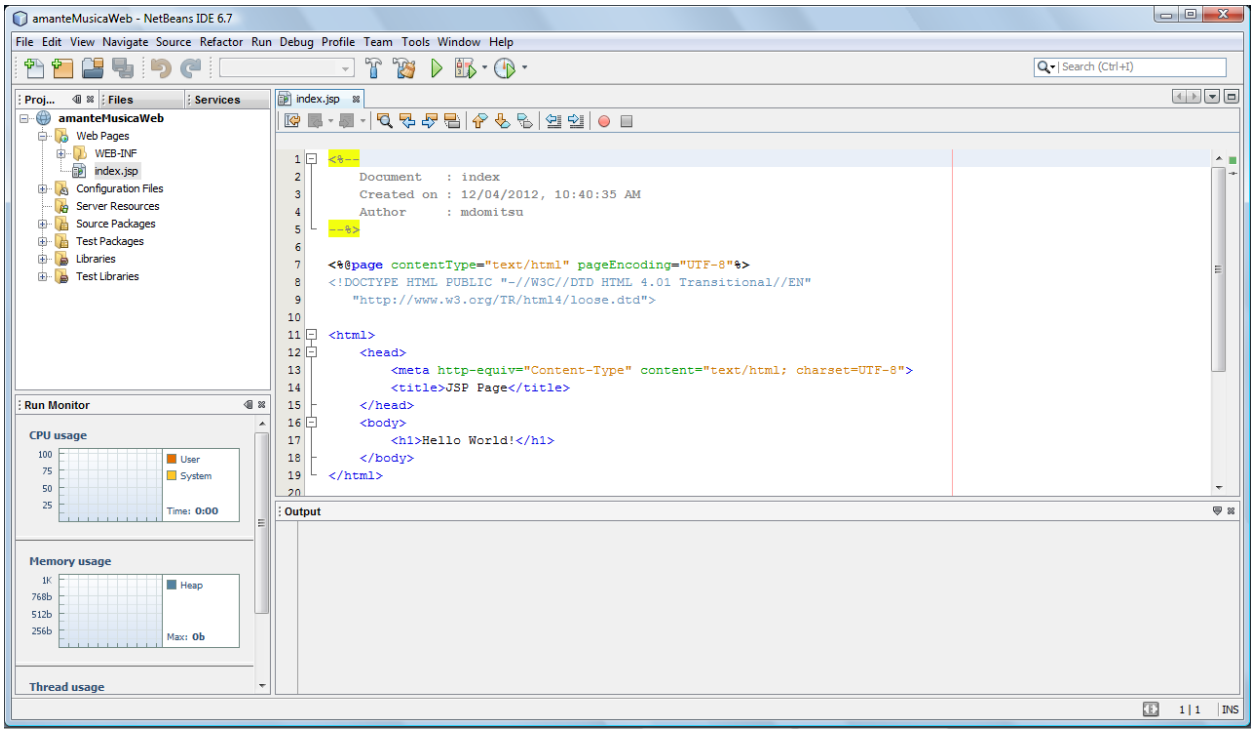

Figura 7

9. Si en el recuadro del árbol de los proyectos hacemos clic en la pestaña Files, aparecerá un árbol con todos los archivos de los proyectos, figura 8. Otra vez, en este momento sólo aparecen los archivos del proyecto amanteMusicaWeb.

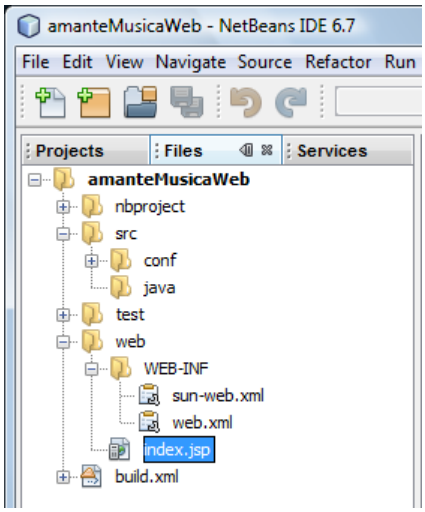

Figura 8

10. Edite el archivo index.jsp para que su código sea el siguiente:

index.jsp

```
<%--
    Document   : index
    Created on : 12/04/2012, 10:40:35 AM
    Author     : mdomitsu

    Esta página JSP es la página inicial de la aplicación Web AmanteMusicaWeb
    Despliega el menú de opciones.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!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=UTF-8">
    <link rel="stylesheet" type="text/css" href="estilos.css" />
    <title>Amante Música: Versión JSP</title>
  </head>

  <body>
    <%-- Incluye la barra de título --%>
    <%@include file="jspf/titulo.jspf"%>

    <%-- Incluye el menú --%>
    <%@include file="jspf/menuPpal.jspf"%>
  </body>
</html>
```

11. Guarde la página JSP seleccionando del menú principal la opción **File/Save**, presione las teclas **Ctrl+S** o haga clic en el icono **Save All**, figura 9.

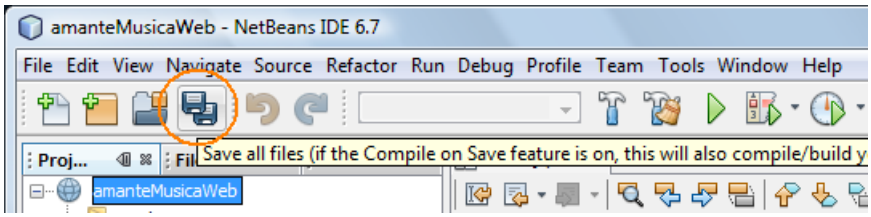

Figura 9

Descriptor de Despliegue de una Aplicación Web

El **Descriptor de Despliegue de una Aplicación Web** es un archivo XML en el que cada aplicación Web guarda toda la información que el servidor tiene que saber acerca de la aplicación. El nombre de ese archivo es web.xml y se encuentra en el directorio web\WEB-INF del directorio del proyecto, figura 10:

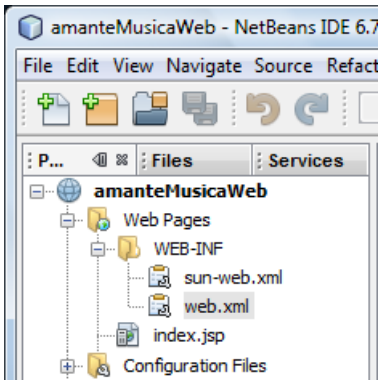

Figura 10

Si hacemos doble clic sobre su nodo, Netbeans lo abre en un asistente que nos permite inspeccionar/modificar el contenido de ese archivo, figura 11.

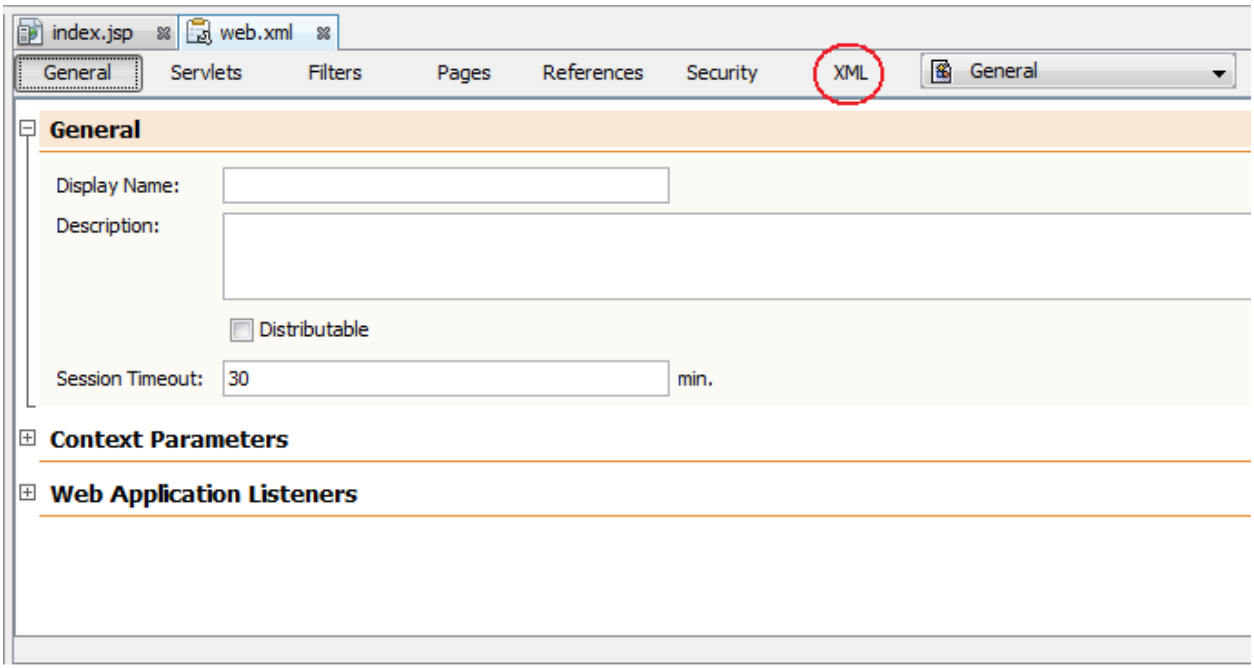

Figura 11

Sin embargo podemos inspeccionar el contenido del archivo haciendo clic en la pestaña **XML**, figura 12. Aquí podemos que en el descriptor hay, hasta este momento dos entradas:

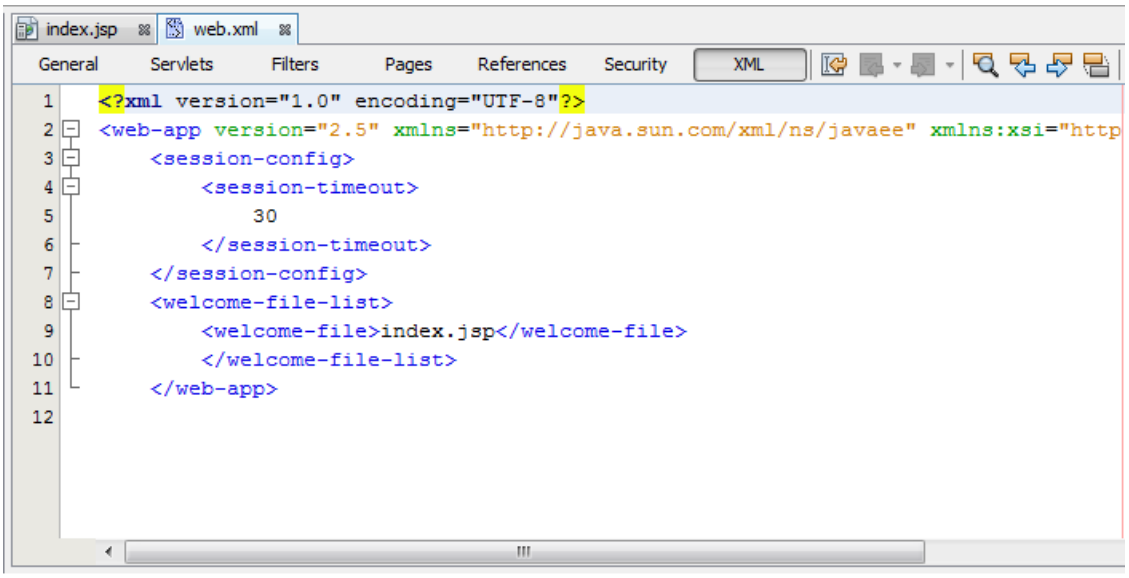

Figura 12

```
<session-timeout>
    30
</session-timeout>
```

Que indica el tiempo (en minutos) que el servidor mantendrá la sesión antes de terminarla si no detecta actividad por el usuario.

```
<welcome-file-list>
   <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Que indica el nombre del archivo o archivos que serán usados como página inicial de la aplicación.

Ubicación de los Archivos de una Aplicación Web

La figura 13, muestra los directorios y archivos generados al crear el proyecto. El proyecto se guarda en una carpeta con el nombre del proyecto, **amanteMusicaWeb** en este caso. Las páginas JSP se almacenan en un directorio llamado **web**, dentro del directorio del proyecto.

Por otro lado si se crean clases, el directorio raíz de su estructura de paquetes será directorio **scr**.

Creación de una Página JSP

Para crear una página JSP se sigue el siguiente procedimiento:

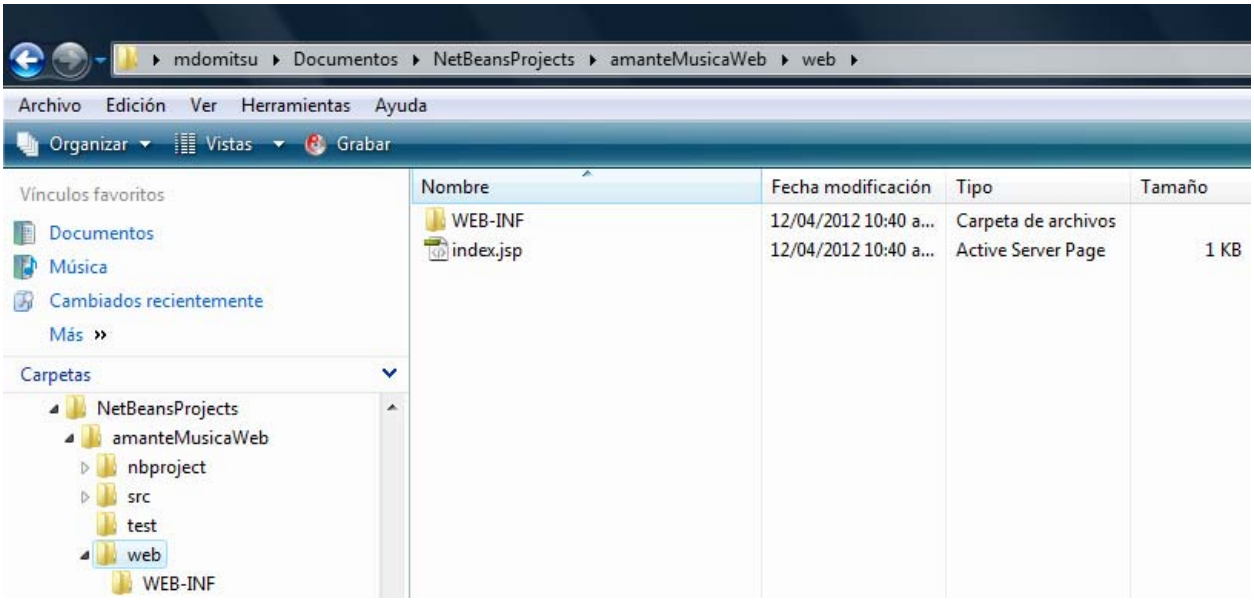

Figura 13

1. Del menú principal de NetBeans, figura 1, seleccione la opción **Files/New File**, presione las teclas **Ctrl+ N** o haga clic en el icono **New File**, como se muestra en la figura 14:

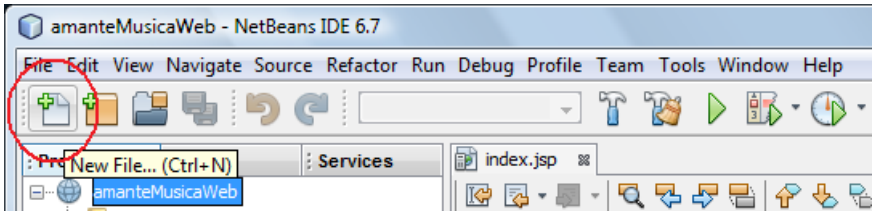

Figura 14

2. Aparecerá el primer cuadro de diálogo del asistente para crear un nuevo archivo, figura 15. Como deseamos crear una página JSP, seleccionaremos la opción **Web** en el recuadro **Categories:** y la opción **JSP** en el recuadro **File Types:**, y luego presionaremos el botón **Next>**.

3. Aparecerá el segundo cuadro de diálogo del asistente para crear un archivo, figura 16. En este cuadro de diálogo seleccionaremos el nombre y la ubicación de la página JSP.

   a) En el campo de texto **JSP File Name**, establezca el nombre de la página JSP. Por ejemplo, "**canciones**".
   b) Deje el resto de los campos a sus valores prestablecidos.
   c) Presione el botón **Finish**.

4. Desaparecerá el asistente para crear un nuevo archivo y aparecerá el esqueleto de la página JSP creada.

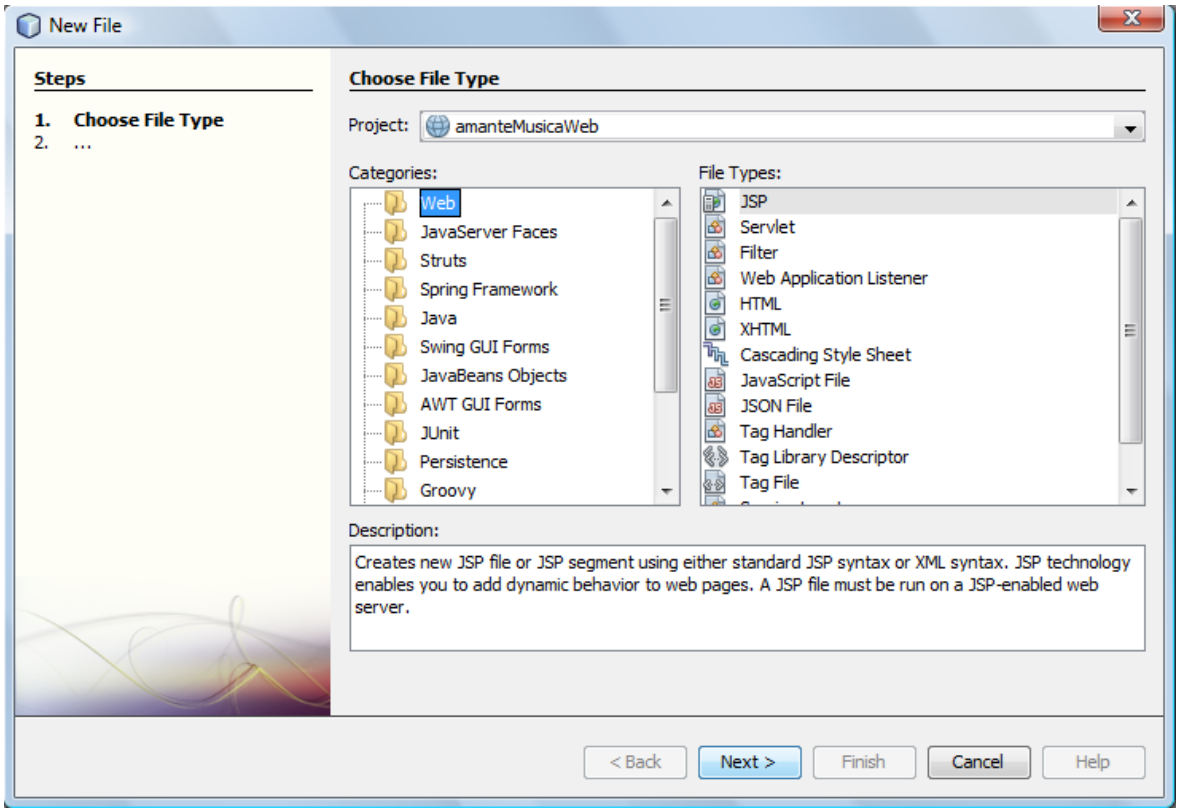

Figura 15

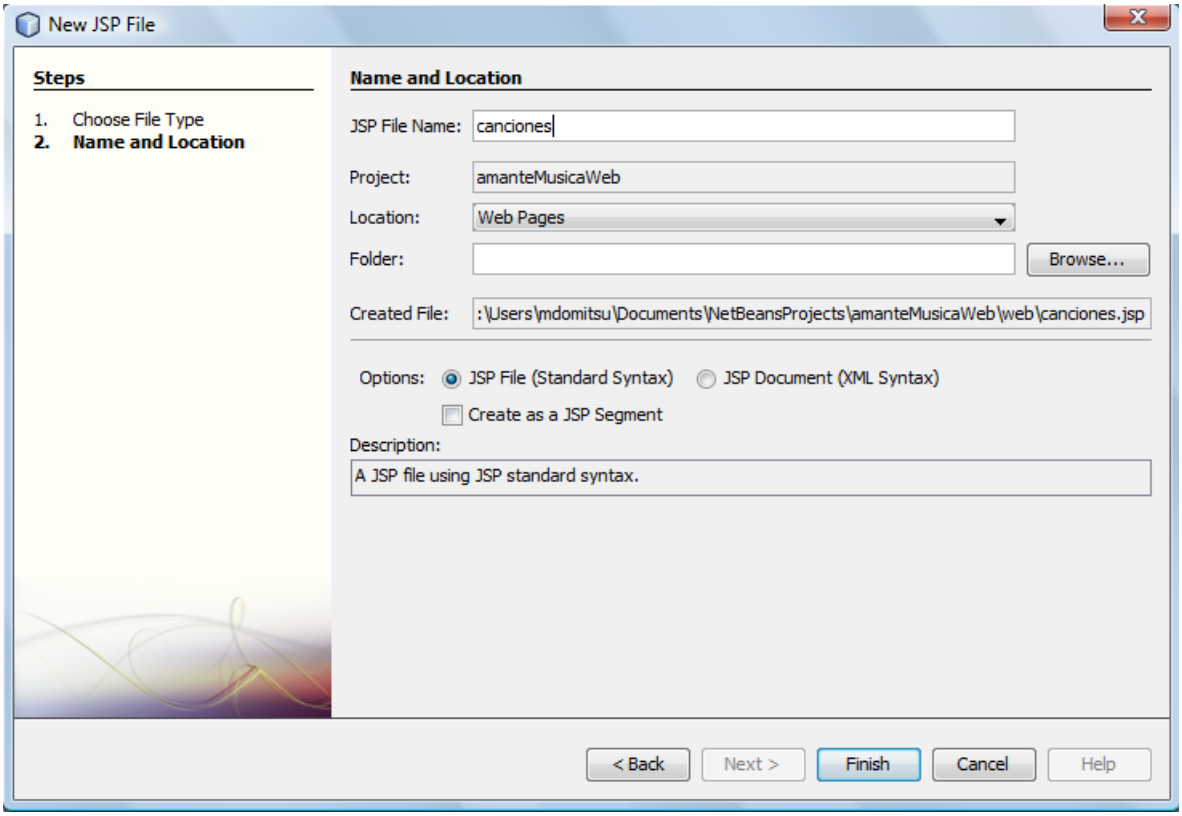

Figura 16

5. Edite la página JSP **canciones.jsp** para que su código sea el mostrado en la página siguiente:

canciones.jsp

```
<%--
    Document   : canciones
    Created on : 12/04/2012, 12:51:06 PM
    Author     : mdomitsu

    Esta página JSP es la página con el menú canciones de la aplicación
    Web AmanteMusica versión JSTL. Despliega el menú de opciones.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!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=UTF-
8">
    <link rel="stylesheet" type="text/css" href="estilos.css" />
    <title>Amante Música - Versión JSTL: Menú Canciones</title>
  </head>

  <body>
    <%-- Incluye la barra de título --%>
    <%@include file="jspf/titulo.jspf"%>

    <%-- Incluye el menú --%>
    <%@include file="jspf/menuCanciones.jspf"%>

    </body>
</html>
```

6. Guarde la página JSP.

Edición de Múltiples Páginas JSP

Podemos tener más de una archivo abierto en la ventana de edición, cada uno en su propio panel, como se muestra en la figura 17, en la que tenemos en la ventana de edición de NetBeans dos páginas JSP: index.jsp y canciones.jsp. Las pestañas en la parte superior nos permiten seleccionar la clase o página JSP que se desea editar.

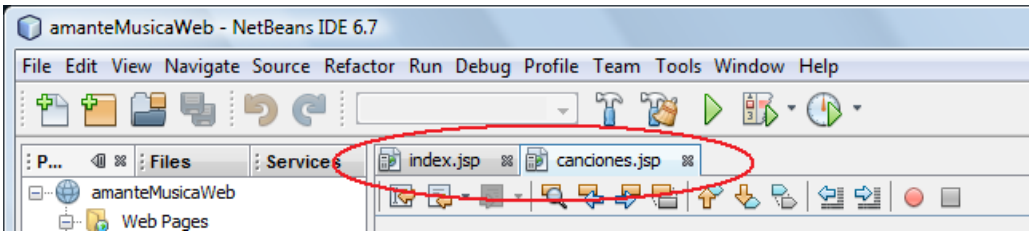

Figura 17

Podemos eliminar un archivo de la ventana de edición, haciendo clic en el icono con la X que se encuentra en la pestaña de cada clase, figura 18.

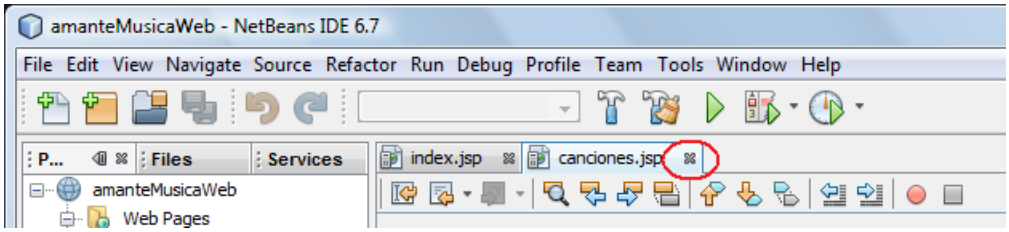

Figura 18

Si deseamos abrir un archivo existente en la ventana de edición podemos hacer clic en su nombre en el árbol de archivos o en el árbol de proyectos que se encuentran a la izquierda de la ventana de edición, como se muestra en la figura 19.

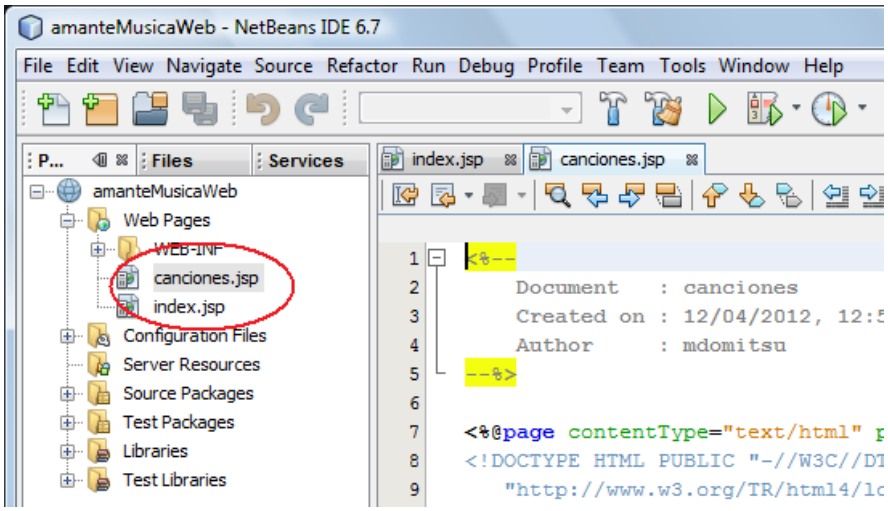

Figura 19

Creación de un Fragmento de Página JSP

Un fragmento de página JSP es un archivo texto que contiene código HTML o JSP y que deseamos incluirlo en una o más páginas JSP. Para crear un fragmento de una página JSP se sigue el siguiente procedimiento:

1. Siga los pasos 1 a 3 del procedimiento para crear una página JSP hasta que aparezca el segundo cuadro de diálogo del asistente para crear una página JSP, figura 20:

   a) En el campo de texto **JSP File Name**, establezca el nombre del fragmento de página JSP. Por ejemplo, "**titulo**".

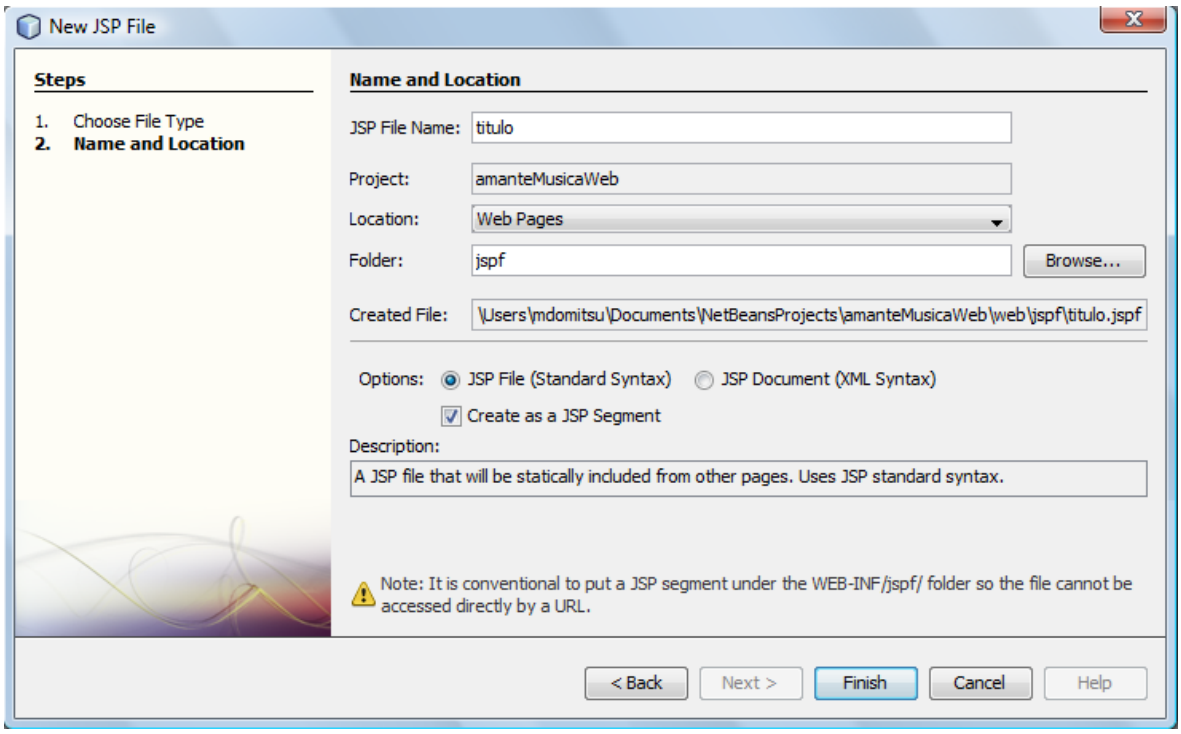

Figura 20

b) En el campo de texto **Folder**, establezca el nombe del directorio en el que se almacenará el archivo con el fragmento de página JSP. Por ejemplo "**jspf**".
c) Asegúrese que la casilla de verificación **Create as a JSP Segment** esté seleccionada.
d) Deje el resto de los campos a sus valores prestablecidos.
e) Presione el botón **Finish**.

2. Desaparecerá el asistente para crear un nuevo archivo y aparecerá el esqueleto del fragmento de página JSP creado, figura 21.

3. Edite el fragmento de página JSP **titulo.jspf** para que su código sea el mostrado en el listado siguiente:

titulo.jspf

```
<%-- Este fragmento de página JSP es la barra de título de la aplicación
     AmanteMusica
--%>
<%@ page pageEncoding="UTF-8" %>
   <div class="encabezado">
      <div class="titulo">
       <img class="titulo" alt="titulo" src="imagenes/titulo.PNG"
       width="800px" height="150px">
     </div>
   </div>
```

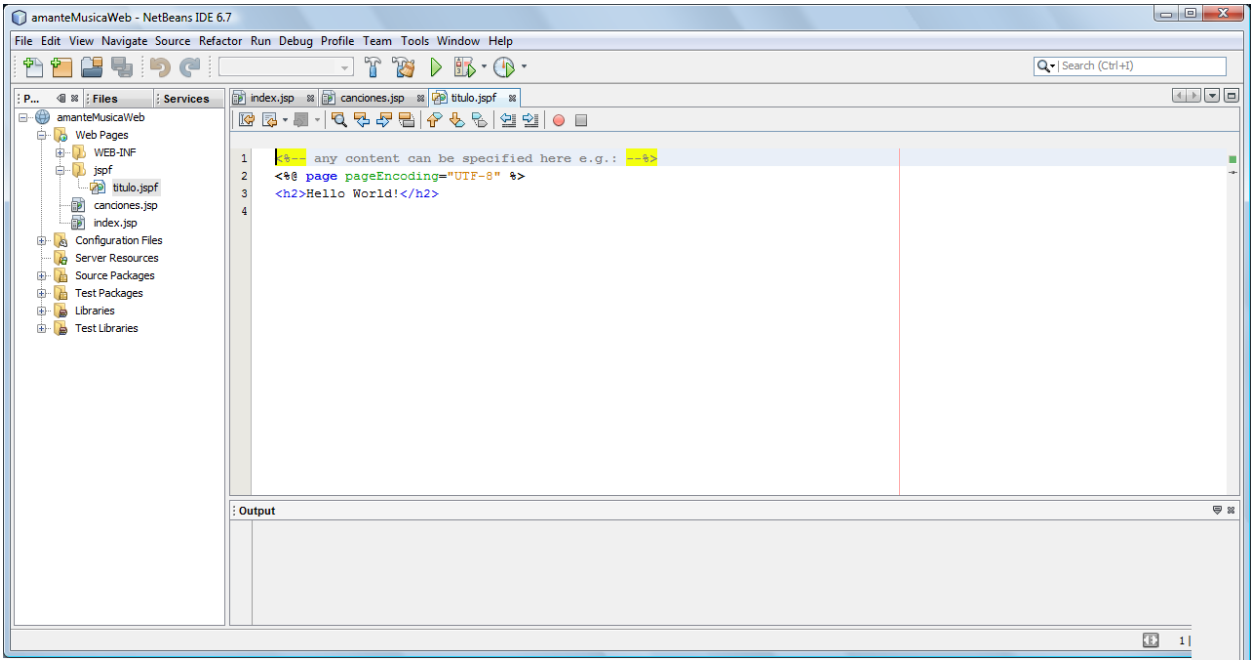

Figura 21

4. Guarde el fragmento de página página JSP.

5. Repita el procedimiento anterior para crear los fragmentos de página JSP menuPpal.jspf y menuCanciones.jspf, modificando su código para que queden como los listados siguientes:

menuPpal.jspf

```
<%-- Este fragmento de página JSP es el menú principal de la aplicación
     AmanteMusica versión JSP
--%>
<%@ page pageEncoding="UTF-8" %>
   <div id="menuppal">
      <ul id="menu">
         <li><a href="canciones.jsp">Catálogo de canciones</a></li>
         <li><a href="canciones.jsp">Catálogo de películas</a></li>
         <li><a href="canciones.jsp">Catálogo de géneros</a></li>
      </ul>
   </div>
```

menuCanciones.jspf

```
<%-- Este fragmento de página JSP es el menú principal de la aplicación
     AmanteMusica versión JSTL
--%>
<%@ page pageEncoding="UTF-8" %>
   <div id="menuppal">
      <ul id="menu">
         <li><a href="control?tarea=agregarCancion">Agregar
canción</a></li>
         <li><a href="control?tarea=actualizarCancion">Actualizar
canción</a></li>
```

```
         <li><a href="control?tarea=eliminarCanciones">Eliminar
canciones</a></li>
         <li><a href="control?tarea=listarCanciones">Listar
canciones</a></li>
         <li><a href="control?tarea=listarCancionesGenero">Listar
canciones por género</a></li>
         <li><a href="control?tarea=listarCancionesPeriodo">Listar
canciones por periodo</a></li>
         <li><a href="index.jsp">Página Inicial</a></li>
      </ul>
   </div>
```

Compilación de una Página JSP

Para compilar la página JSP que se encuentra en la ventana de edición seleccione del menú principal la opción **Run/Compile "*NombreJSP*.jsp". "*NombreJSP*"** es el nombre de la página JSP a compilar, por ejemplo **"index"**. También puede presionar la tecla **F9**.

Durante la compilación, NetBeans muestra los mensajes resultantes del proceso, como se muestra en la figura 22.

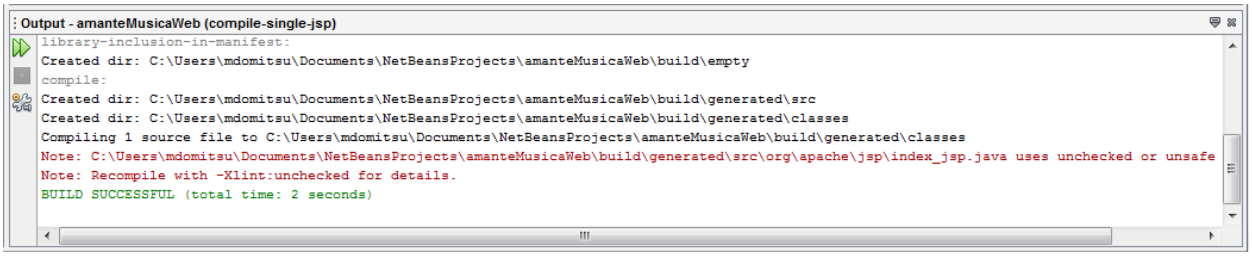

Figura 22

Compilación del Proyecto

Para compilar todas las clases y páginas JSP de un proyecto, seleccione del menú principal la opción **Run/Build Main Project**, presione la tecla **F11** o presione el icono **Build Main Project,** mostrado en la figura 23.

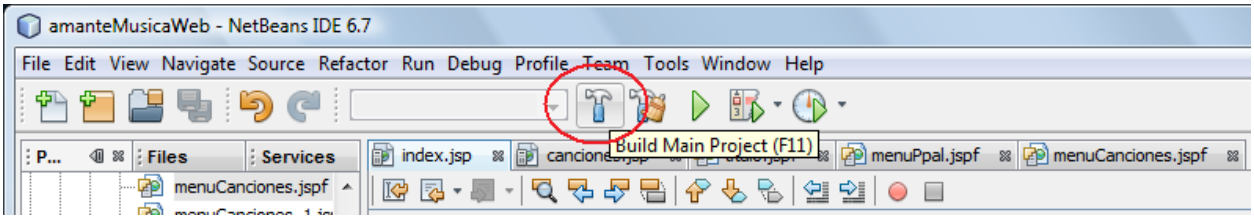

Figura 23

Durante la compilación, NetBeans muestra los mensajes resultantes del proceso, como se muestra en la figura 22.

Ubicación del Archivo de despliegue de la Aplicación Web

Al compilar todo el proyecto de una aplicación Web, NetBeans empaca los archivos con el código "byteCode" de las clases, las páginas JSP, los archivos con los recursos y los archivos de configuración en un archivo WAR, con el nombre del proyecto y la extensión **".war"** y lo almacena en el directorio **"dist"** dentro del directorio del proyecto, figura 24.

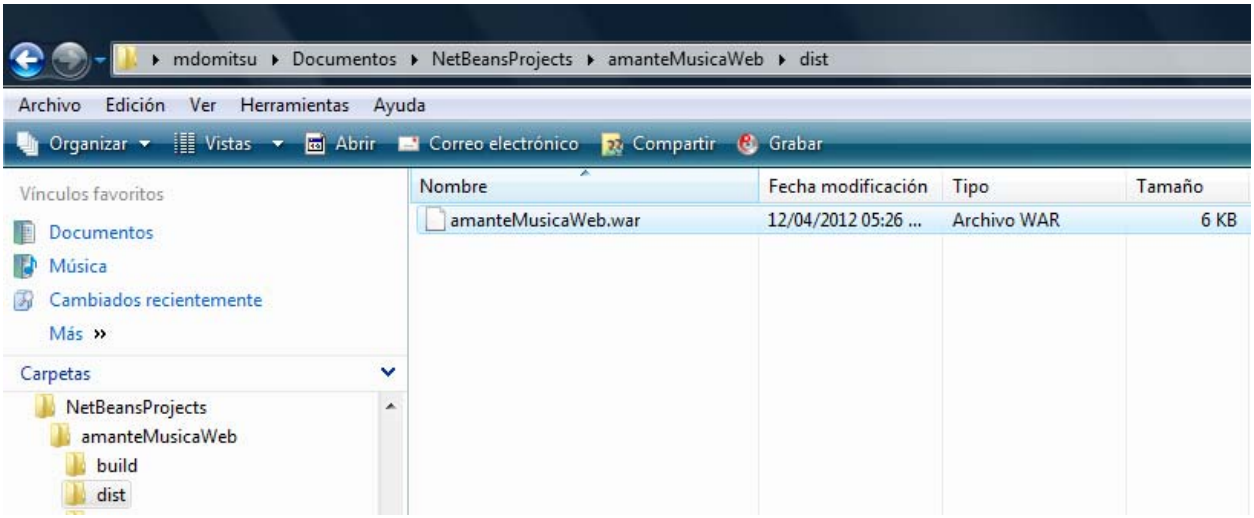

Figura 24

Ejecución de una Aplicación Web

1. Para ejecutar la aplicación dentro de NetBeans, seleccione del menú principal la opción **Run/Run Main Project**, presione la tecla **F6** o haga clic en el icono **Run Main Project**, mostrado en la figura 25.

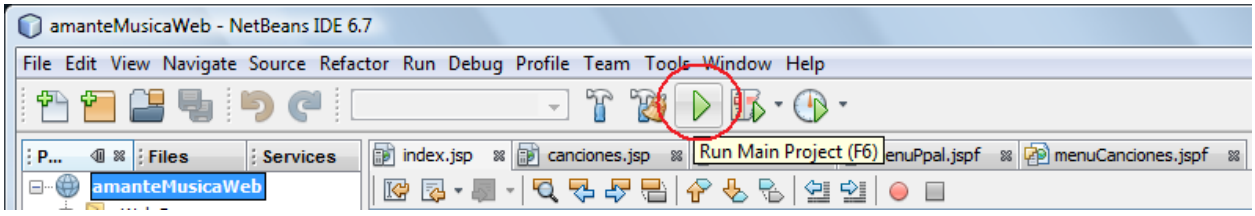

Figura 25

2. Al hacerlo NetBeans, compilará las páginas JSP y las clases del proyecto. Enseguida iniciará la ejecución del servidor de aplicaciones que tiene empotrado, mostrando el proceso en la ventana de salida: **Ouput:,** figura 26. Al terminar le enviará la página JSP inicial (index.jsp) al navegador Web, figura 27.

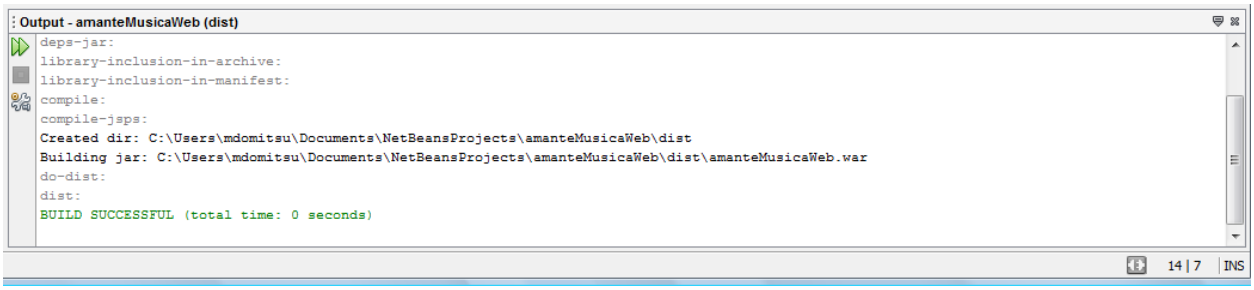

Figura 26

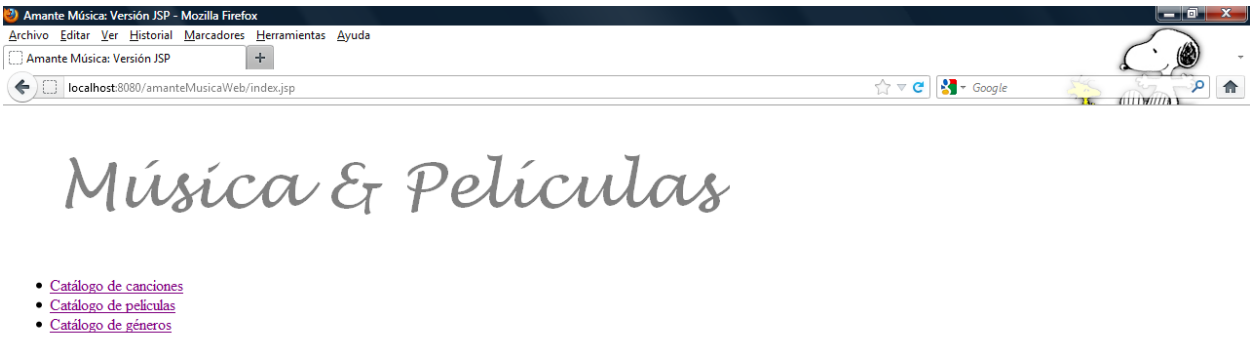

Figura 27

Creación de una Hoja de Estilos en Cascada Externa

Para crear una Hoja de Estilos en Cascada externa se sigue el siguiente procedimiento:

1. Del menú principal de NetBeans, figura 1, seleccione la opción **Files/New File**, presione las teclas **Ctrl+ N** o haga clic en el icono **New File**, como se muestra en la figura 14:

2. Aparecerá el primer cuadro de diálogo del asistente para crear un nuevo archivo, figura 28. Como deseamos crear una Hoja de Estilos en Cascada externa, seleccionaremos la opción **Web** en el recuadro **Categories:** y la opción **Cascading Style sheets** en el recuadro **File Types:**, y luego presionaremos el botón **Next>**.

3. Aparecerá el segundo cuadro de diálogo del asistente para crear un archivo, figura 29. En este cuadro de diálogo seleccionaremos el nombre y la ubicación de la Hoja de Estilos en Cascada externa.

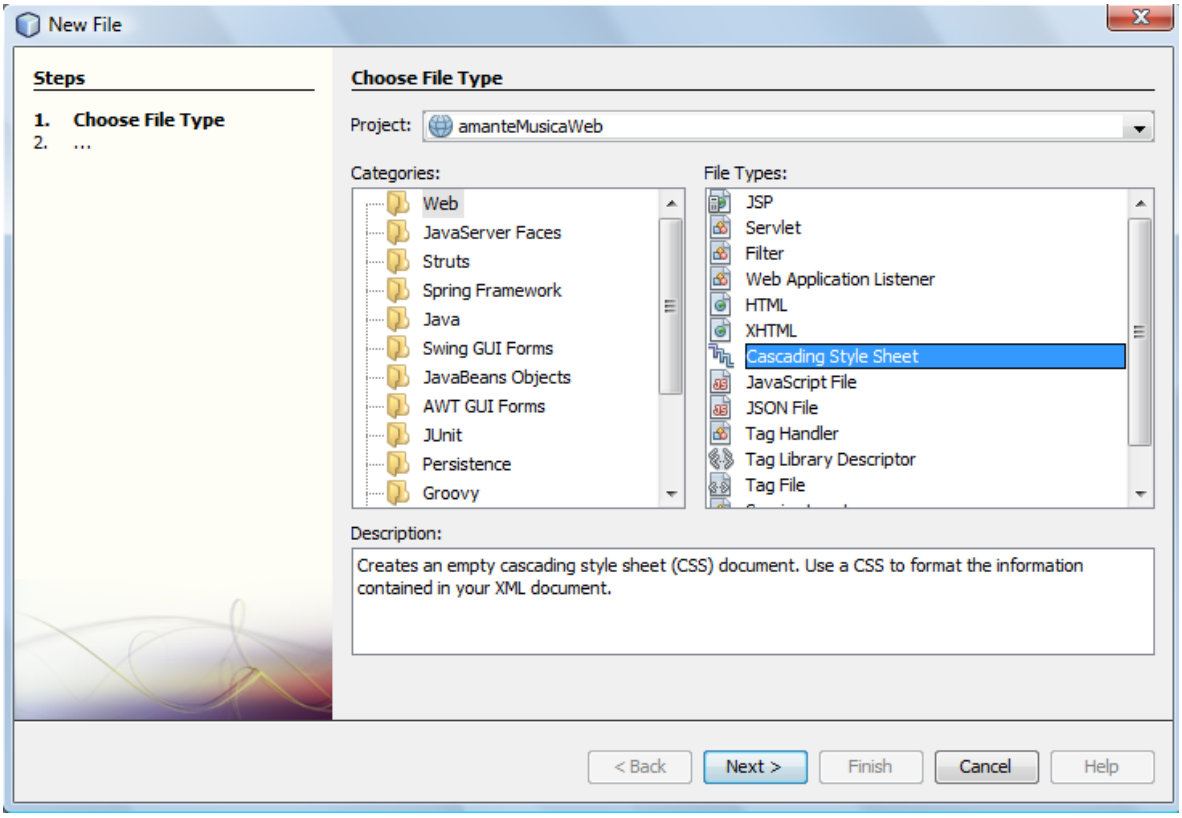

Figura 28

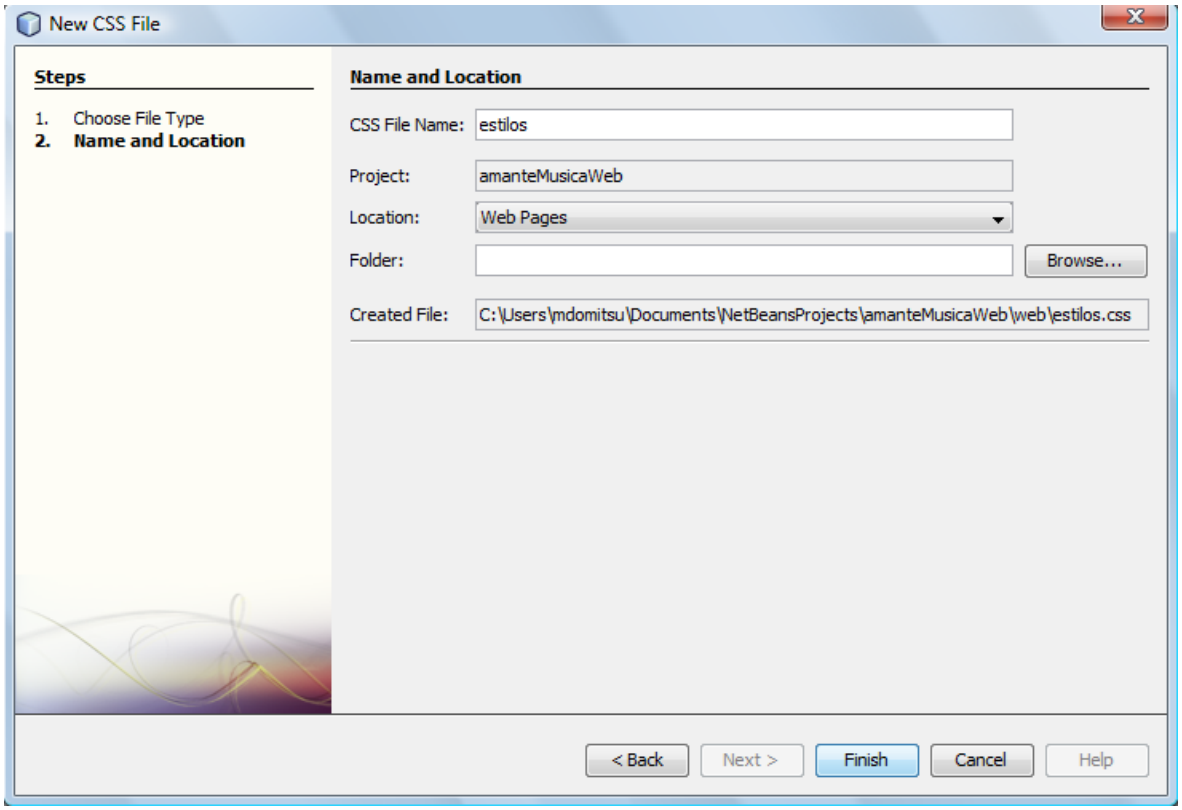

Figura 29

a) En el campo de texto **CSS File Name**, establezca el nombre de la Hoja de Estilos en Cascada externa. Por ejemplo, "**estilos**".
b) Deje el resto de los campos a sus valores prestablecidos.
c) Presione el botón **Finish**.

4. Desaparecerá el asistente para crear un nuevo archivo y aparecerá el esqueleto de la Hoja de Estilos en Cascada externa creada.

5. Edite la Hoja de Estilos en Cascada externa **estilos.css** para que su código sea el mostrado en el listado siguiente:

estilos.css

```
/*
    Document   : estilos
    Created on : 12/04/2012, 06:05:12 PM
    Author     : mdomitsu
    Description:
        Establece las reglas de estilo generas para la aplicación
        AmanteMusicaWeb.
*/

/* Regla de estilo la división con el encabezado */
div.encabezado
{
  height: 150px;
  width: 100%;
  border-bottom: 3px ridge Silver;
  text-align: center;
}

/* Regla de estilo para la división con el logotipo dentro del encabezado
*/
div.titulo {
    margin-left: auto;
    margin-right: auto;
    width: 800px;
}

/* Regla de estilo para la división con el menú */
div#menuPpal {
   top: 151px;
   width: 15%;
   height: auto;
   background-color: #ccc;
}

/* Regla de estilo para la lista no ordenada que forma el menú */
ul#menu {
  margin: 0;
  padding: 0;
  list-style: none;
  border-top: 1px solid #000;
}
```

```
/* Regla de estilo para la los elementos de la lista no ordenada que
forman las opciones del menú */
ul#menu li {
  position: relative;
  width: 15%;
  float: left;
  clear: left;
}

/* Regla de estilo para las ligas que forman las opciones del menú */
ul#menu li a {
  display: block;
  text-decoration: none;
  color: #000;
  background: #ccc;
  line-height: 2em;
  height: 2em;
  padding: 0 5px;
  border: 1px solid #000;
  border-top: none;
}

/* Regla de estilo para cuando se pasa el raton sobre las ligas que forman
las opciones del menú */
#menu li:hover a {
   color: #ccc;
   background-color: #777;
}

/* Regla de estilo para segmento principal de las páginas de la aplicación
*/
div.principal {
  position: absolute;
  top: 151px;
  left: 20%;
  width: 80%;
  border-right: 3px ridge Silver;
}

div.contenido {
  padding-left: 1em;
  padding-right: 1em;
  padding-top: 1em;
}

/* Regla de estilo para las ligas de las opciones del menú */
div.opcion a {
  text-decoration: none;
  color: white;
}

/* Regla de estilo para centrar horizontalmente una tabla */
table.centrada {
   margin-left: auto;
   margin-right: auto;
}
```

```
caption {
   caption-side: top;
   margin-left: auto;
   margin-right: auto;
   margin-bottom: 0.5em;
   font-size: 200%;
   font-weight: bold;
}

/* Regla de estilo para encabezado de columna de tabla */
th {
    text-align: center;
}

/* Regla de estilo para justificar a la derecha celdas de tabla */
td.derecha {
    text-align: right;
}

/* Regla de estilo para las celdas de tabla */
td.gris {
    background-color: rgb(220, 220, 220);
    padding-left: 1em;
    padding-right: 1em;
    border: 1px solid rgb(100, 100, 100);
}

/* Regla de estilo para mensajes de error */
h3.msjError {
  color: red;
  margin-left: 50px;
}

/* Regla de estilo para mensajes de error */
td.msjError {
  color: red;
}

/* Regla de estilo para mensajes de aviso */
td.msjAviso {
  color: green;
  white-space: pre;
}

/* Regla de estilo para mensajes de error */
div.msjError {
  color: red;
  white-space: pre;
}

/* Regla de estilo para centrar horizontalmente encabezados */
h1 {
   text-align: center;
}

tr.encabezado {
```

```
    background-color: rgb(220, 220, 220);
}

tr.non {
    background-color: white;
}

tr.par {
    background-color: yellow;
}

input[readonly="true"] {
    background-color: rgb(220, 220, 220);
}
```

6. Guarde la Hoja de Estilos en Cascada externa. Recompile el proyecto y vuelva a ejecutar la aplicación. En el navegador se desplegará la página inicial de la aplicación, figura 30:

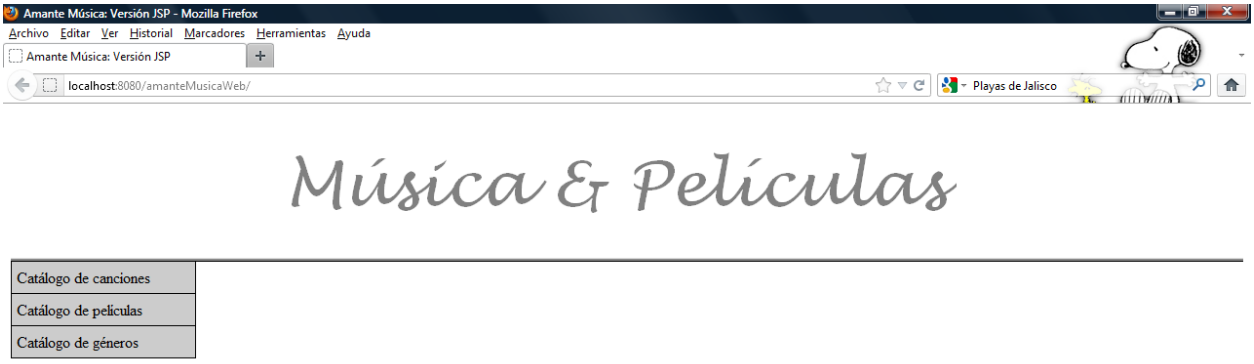

Figura 30

Creación de un Archivo JavaScript Externo

Para crear un archivo externo con código JavaScript se sigue el siguiente procedimiento:

1. Del menú principal de NetBeans, figura 1, seleccione la opción **Files/New File**, presione las teclas **Ctrl+ N** o haga clic en el icono **New File**, como se muestra en la figura 14:

2. Aparecerá el primer cuadro de diálogo del asistente para crear un nuevo archivo, figura 31. Como deseamos crear un archivo externo con código JavaScript,

seleccionaremos la opción **Web** en el recuadro **Categories:** y la opción **JavaScript File** en el recuadro **File Types:**, y luego presionaremos el botón **Next>**.

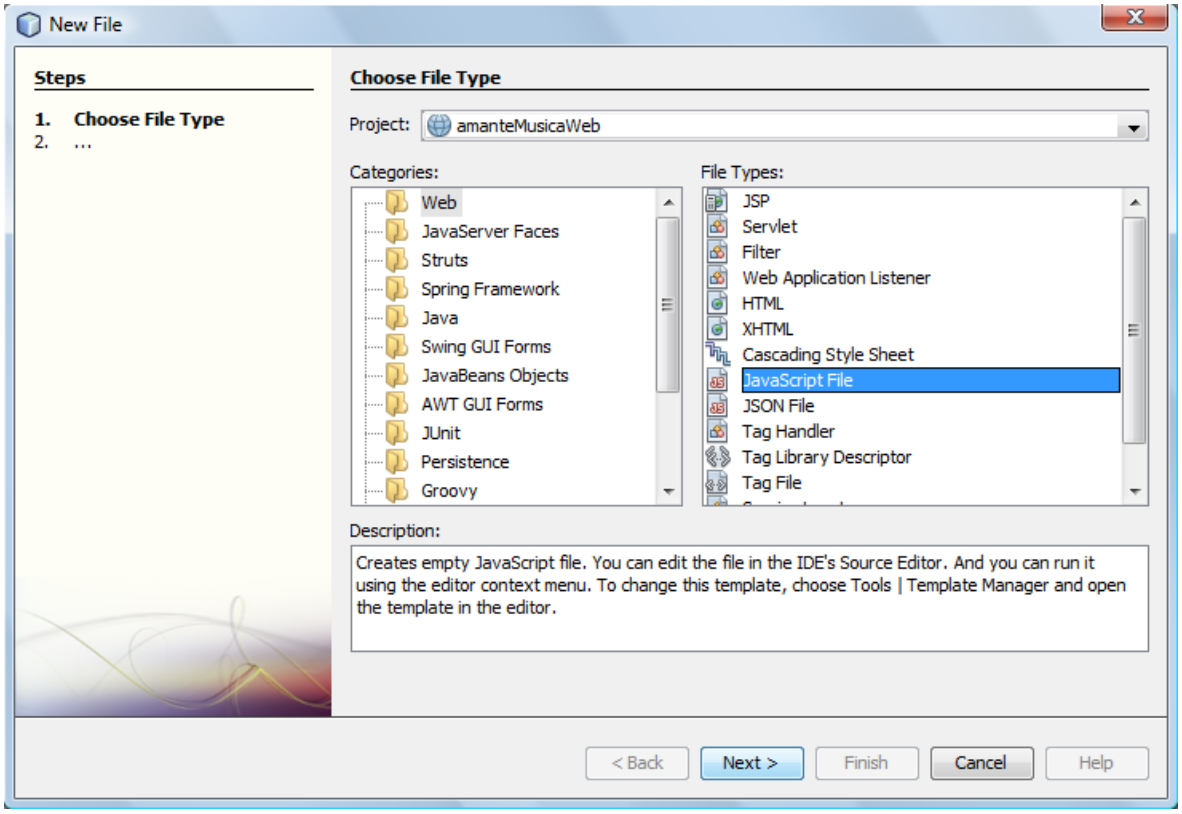

Figura 31

3. Aparecerá el segundo cuadro de diálogo del asistente para crear un archivo, figura 32. En este cuadro de diálogo seleccionaremos el nombre y la ubicación del archivo externo con código JavaScript.

   a) En el campo de texto **File Name**, establezca el nombre del archivo externo con código JavaScript. Por ejemplo, "**valida**".
   b) Deje el resto de los campos a sus valores prestablecidos.
   c) Presione el botón **Finish**.

4. Desaparecerá el asistente para crear un nuevo archivo y aparecerá el esqueleto del archivo externo con código JavaScript.

5. Edite el archivo externo con código JavaScript **valida.js** para que su código sea el mostrado en el listado siguiente:

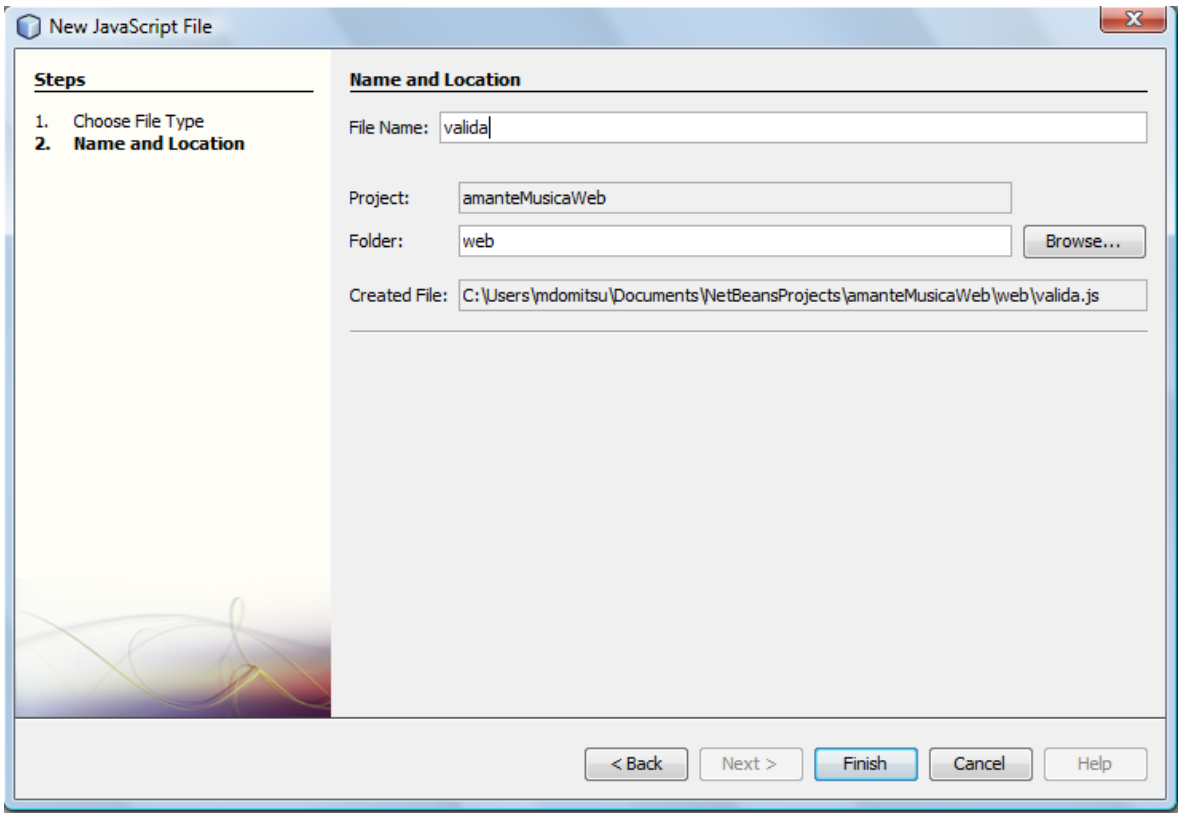

Figura 32

valida.js

```
//
// Funciones de validacion de formularios de Javascript.
//

var nbsp = 160;               // caracter de espacio
var node_text = 3;            // Tipo de nodo de texto de DOM
var cadenaVacia = /^\s*$/ ;
var campoEnfocarGlobal;       // Variable global para almacenar el
campoEnfocar

//
// enfocaAtrasado()
//
// Hace que el elemento almacenado en la variable global
// campoValidarGlobalDelayed obtenca el foco. Ajuste para
// remediar un bug de IE y otros
//
function enfocaAtrasado() {
  campoEnfocarGlobal.focus();
}

//
// enfoca()
//
// Hace que el elemento del parametro obtenga el foco
//
```

```
// Parametros:
//    - campoEnfocar: Elemento a obtener el foco
//
function enfoca(campoEnfocar) {
  // Guarda campoEnfocar en la variable global para conservar el valor
  // cuando la funcion termine
  campoEnfocarGlobal = campoEnfocar;
  setTimeout('enfocaAtrasado()', 100);
}

//
// trim()
//
// Elimina los caracteres blancos al principio o final
// de la cadena del parametro
//
// Parametros
//    - str: Cadena a procesar
//
function trim(str) {
  return str.replace(/^\s+|\s+$/g, '');
}

//
// despliegaMensaje()
//
// Despliega un mensaje de error o aviso
//
// Parametros
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - claseMensaje: Clase asociada al mensaje para usarse en CSS,
warn/error.
//    - mensaje:      Mensaje a desplegar
//
// Debe llamarse antes a la funcion validacionComun
//
function despliegaMensaje(idMensaje, claseMensaje, mensaje) {
  var mensajeDesplegar;

  // Si el mensaje a desplegar es una cadena vacía
  if (cadenaVacia.test(mensaje))
    // Hacer que la cadena a desplegar sea el caracter de espacio
    mensajeDesplegar = String.fromCharCode(nbsp);
  else
    // Hacer que la cadena a desplegar sea el mensaje
    mensajeDesplegar = mensaje;

  // Despliega el mensaje
  var elem = document.getElementById(idMensaje);
  elem.firstChild.nodeValue = mensajeDesplegar;

  // Establece la clase CSS para establecer las propiedades del mensaje
  elem.className = claseMensaje;
}

//
// validacionComun()
```

```
//
// Codigo comun para todas las funciones de validacion:
// Si la version del navegador es vieja, pasa la validacion
// para que la validacion la haga el servidor
//
// Parametros:
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - requerido:    Campo requerido
//
// Regresa:
//    - true:     Pasa la validacion
//    - false:    Falla la validation
//    - prosigue: Continua con la siguiente validacion
//
var prosigue = 2;

function validacionComun(campoValidar, idMensaje, requerido) {
  // Si el navegador es viejo
  if (!document.getElementById)
    // Deja la validacion al servidor
    return true;

  var elem = document.getElementById(idMensaje);
  // Si el navegador es viejo, deja la validacion al servidor
  if (!elem.firstChild) return true;
  // El elemento en que se desplegara el mensaje no es el correcto
  if (elem.firstChild.nodeType != node_text) return true;

  // Si el campo a validar esta vacio
  if(cadenaVacia.test(campoValidar.value)) {
     // Si el campo esta vacio y es obligatorio
     if(requerido) {
        despliegaMensaje(idMensaje, "msjError", "Error: Se requiere un
valor");
        enfoca(campoValidar);
        return false;
     }
     // Si el campo esta vacio y no es obligatorio
     else {
        // Borra un posible mensaje de error previo
        despliegaMensaje(idMensaje, "msjAviso", "");
        return true;
    }
  }
  return prosigue;
}

//
// validaPresente()
//
// Valida si se ha teclaeado algo en el campo de texto
//
// Parametros:
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//
```

```
// Regresa:
//    true si se teclaeado algo, falso en caso contrario
//
function validaPresente(campoValidar, idMensaje) {
  var stat = validacionComun (campoValidar, idMensaje, true);
  if(stat != prosigue) return stat;

  // Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
  return true;
}

//
// validaCadena()
//
// Valida una cadena. Una cadena esta formada de caracteres alfanumericos
y
// tiene una extension maxima
//
// Parametros:
//    - longMax: Longitud maxima de una cadena
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - requerido:    Campo requerido
//
// Regresa:
//    true si es una clave valida, false en caso contrario
//
function validaCadena(longMax, campoValidar, idMensaje, requerido) {
  var stat = validacionComun (campoValidar, idMensaje, requerido);
  if (stat != prosigue) return stat;

  var cadena = trim(campoValidar.value);
  var reCadena = new RegExp( '^\\w{1,' + longMax + '}$' );

  if (!reCadena.test(cadena)) {
    despliegaMensaje (idMensaje, "msjError", "Error: Cadena muy larga");
    enfoca(campoValidar);
    return false;
  }

  // Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
  return true;
}

//
// validaClave()
//
// Valida una clave. Una clave esta formada de 3 letras mayusculas y 4
digitos
//
// Parametros:
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//
// Regresa:
```

```
//    true si es una clave valida, false en caso contrario
//
function validaClave(campoValidar, idMensaje) {
  var stat = validacionComun (campoValidar, idMensaje, true);
  if (stat != prosigue) return stat;

  var clave = trim(campoValidar.value);
  var reClave =  /^[A-Z]{3}[0-9]{4}$/;

  if (!reClave.test(clave)) {
    despliegaMensaje (idMensaje, "msjError", "Error: Clave no valida");
    enfoca(campoValidar);
    return false;
  }

  // Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
  return true;
}

//
// validaEntero()
//
// Valida un entero. El entero tiene de 1 a numDigitos dígitos
//
// Parametros:
//    - numDigitos:   Número máximo de digitos
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - requerido:    Campo requerido
//
// Regresa:
//    true si ok
//
function validaEntero(numDigitos, campoValidar, idMensaje, requerido) {
  var stat = validacionComun (campoValidar, idMensaje, requerido);
  if (stat != prosigue) return stat;

  var entero = trim(campoValidar.value);
  var reEntero = /^\d+$/;

  if (!reEntero.test(entero)) {
    despliegaMensaje (idMensaje, "msjError", "Error: No es un entero");
    enfoca(campoValidar);
    return false;
  }

  reEntero = new RegExp( '^\\d{1,' + numDigitos + '}$' );

  if (!reEntero.test(entero)) {
    despliegaMensaje (idMensaje, "msjError", "Error: Numero muy largo");
    enfoca(campoValidar);
    return false;
  }

  // Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
```

```
  return true;
}

//
// validaEnteroRango()
//
// Valida un entero en el rango [min, max].
//
// Parametros:
//    - min:          Limite inferior del entero
//    - max:          Limite superior del entero
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - requerido:    Campo requerido
//
// Regresa:
//    true si ok
//
function validaEnteroRango(min, max, campoValidar, idMensaje, requerido) {
  var stat = validacionComun (campoValidar, idMensaje, requerido);
  if (stat != prosigue) return stat;

  var entero = trim(campoValidar.value);
  var reEntero = /^\d+$/;

  if (!reEntero.test(entero)) {
    despliegaMensaje (idMensaje, "msjError", "Error: No es un entero");
    enfoca(campoValidar);
    return false;
  }

  reEntero = new RegExp( '^\\d{' + min + ', ' + max + '}$' );

  if (!reEntero.test(entero)) {
    despliegaMensaje (idMensaje, "msjError", "Error: Entero fuera de
rango");
    enfoca(campoValidar);
    return false;
  }
  // Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
  return true;
}

//
// validaFecha()
//
// Valida una fecha en el formato dd/mm/aaaa.
//
// Parametros:
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - requerido:    Campo requerido
//
// Regresa:
//    true si ok
//
```

```
function validaFecha(campoValidar, idMensaje, requerido) {
  var stat = validacionComun (campoValidar, idMensaje, requerido);
  if (stat != prosigue) return stat;

  var fecha = trim(campoValidar.value);
  var reFecha = /^(([0-2]?\d)|([3][0-1]))\/(([0]?\d)|([1][0-2]))\/\d{4}$/

  if (!reFecha.test(fecha)) {
    despliegaMensaje (idMensaje, "msjError", "Error: Fecha no valida");
    enfoca(campoValidar);
    return false;
  }

// Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
  return true;
}

//
// validaEmail()
//
// Valida una direccion de correo
//
// Parametros:
//    - campoValidar: Elemento a validar
//    - idMensaje:    id del elemento en que se desplegara el mensaje
//    - requerido:    Campo requerido
//
// Regresa:
//    true si ok
//
function validaEmail(campoValidar, idMensaje, requerido) {
  var stat = validacionComun (campoValidar, idMensaje, requerido);
  if (stat != prosigue) return stat;

  var email = trim(campoValidar.value);
  var reEmail =  /^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-
]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$/i;

  if (!reEmail.test(email)) {
    despliegaMensaje (idMensaje, "msjError", "Error: E-mail no válido");
    enfoca(campoValidar);
    return false;
  }

  // Borra un posible mensaje de error previo
  despliegaMensaje (idMensaje, "msjAviso", "");
  return true;
}
```

6. Guarde el archivo externo con código JavaScript.

Páginas JSP con JSTL

Si en una página JSP queremos usar elementos de acción de la Biblioteca de Elementos Estándar de JSP, JSTL, debemos agregarle al proyecto de la aplicación esa biblioteca. Para ello se sigue el siguiente procedimiento:

1. Haga clic con el botón derecho sobre el nodo **Libraries** del árbol del proyecto y seleccione la opción **Add Library …** del menú emergente, figura 33.

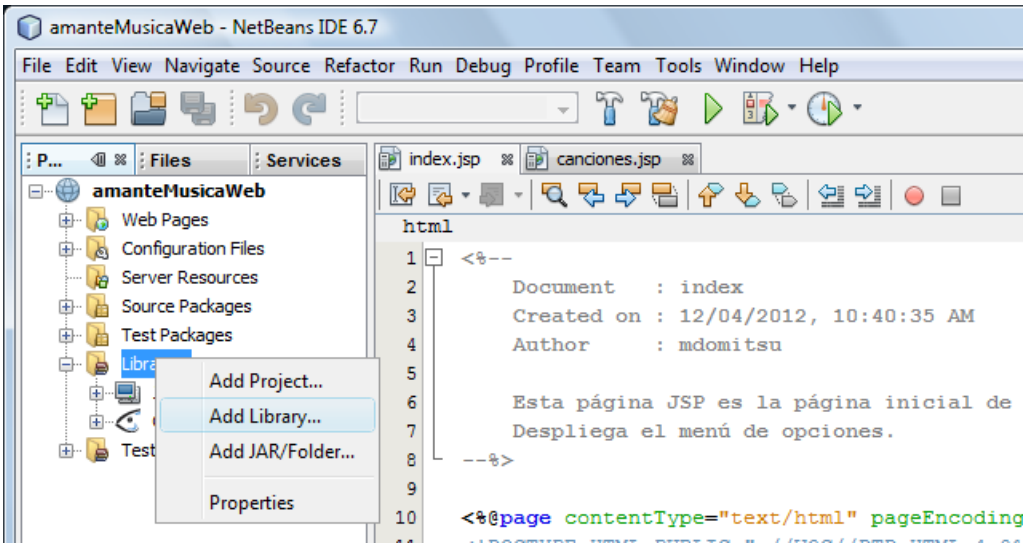

Figura 33

2. Aparece un cuadro de diálogo para seleccionar la biblioteca a agregar, en este caso la Biblioteca de Etiquetas Estándar de JSP: **JSTL 1.1**, Figura 34. Lo seleccionamos y presionamos el botón **Add Library**.

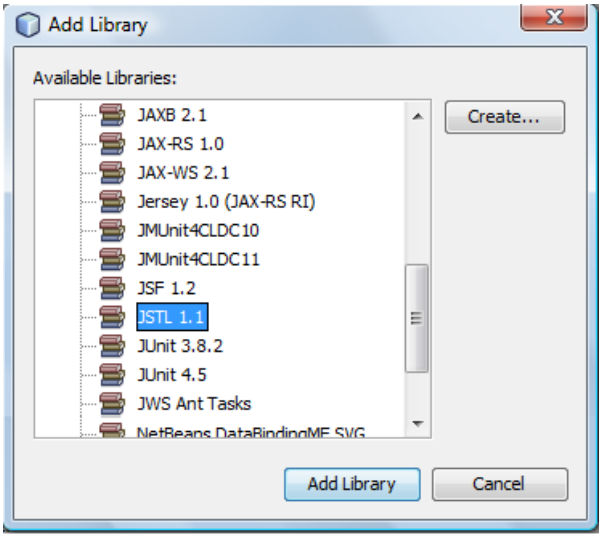

Figura 34

1. El cuadro de diálogo desaparece y veremos que al nodo **Libraries**, del árbol del proyecto, se le han agregado los archivos JAR JSTL 1.1 estándar.jar y JSTL 1.1 jstl.jar que contienen la Biblioteca de Etiquetas Estándar de JSP, figura 35.

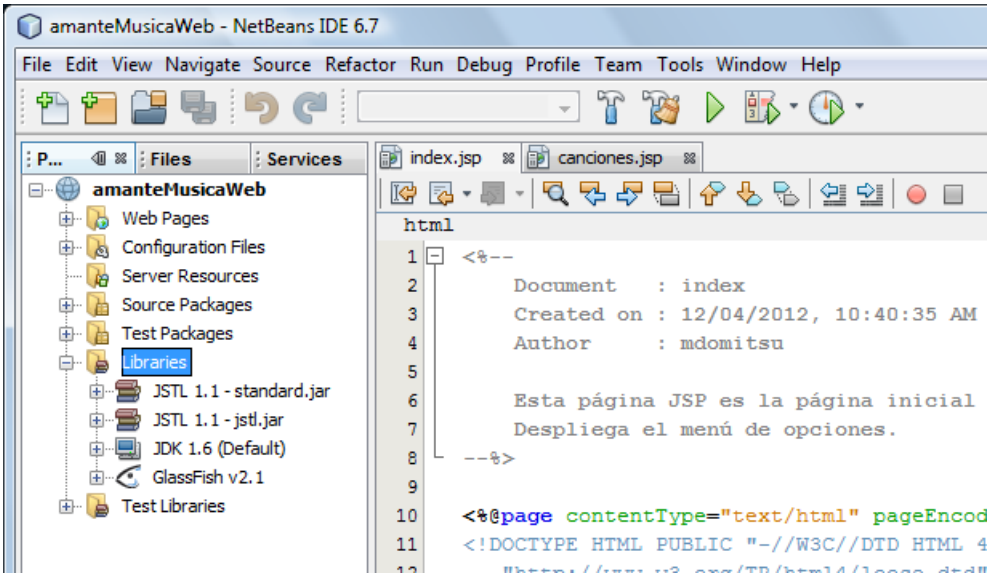

Figura 35

Creación de un Servlet

Para crear un Servlet se sigue el siguiente procedimiento:

1. Del menú principal de NetBeans, figura 1, seleccione la opción **Files/New File**, presione las teclas **Ctrl+ N** o haga clic en el icono **New File**, como se muestra en la figura 14:

2. Aparecerá el primer cuadro de diálogo del asistente para crear un nuevo archivo, figura 36. Como deseamos crear un Servlet, seleccionaremos la opción **Web** en el recuadro **Categories:** y la opción **Servlet** en el recuadro **File Types:**, y luego presionaremos el botón **Next>**.

3. Aparecerá el segundo cuadro de diálogo del asistente para crear un archivo, figura 37. En este cuadro de diálogo seleccionaremos el nombre y la ubicación del Servlet.

   a) En el campo de texto **Class Name**, establezca el nombre de la clase del Servlet. Por ejemplo, "**Control**".
   b) En el campo de texto **Package**, establezca el nombre del paquete en el que se almacenará el servlet. Por ejemplo "**acciones**"
   c) Presione el botón **Finish**.

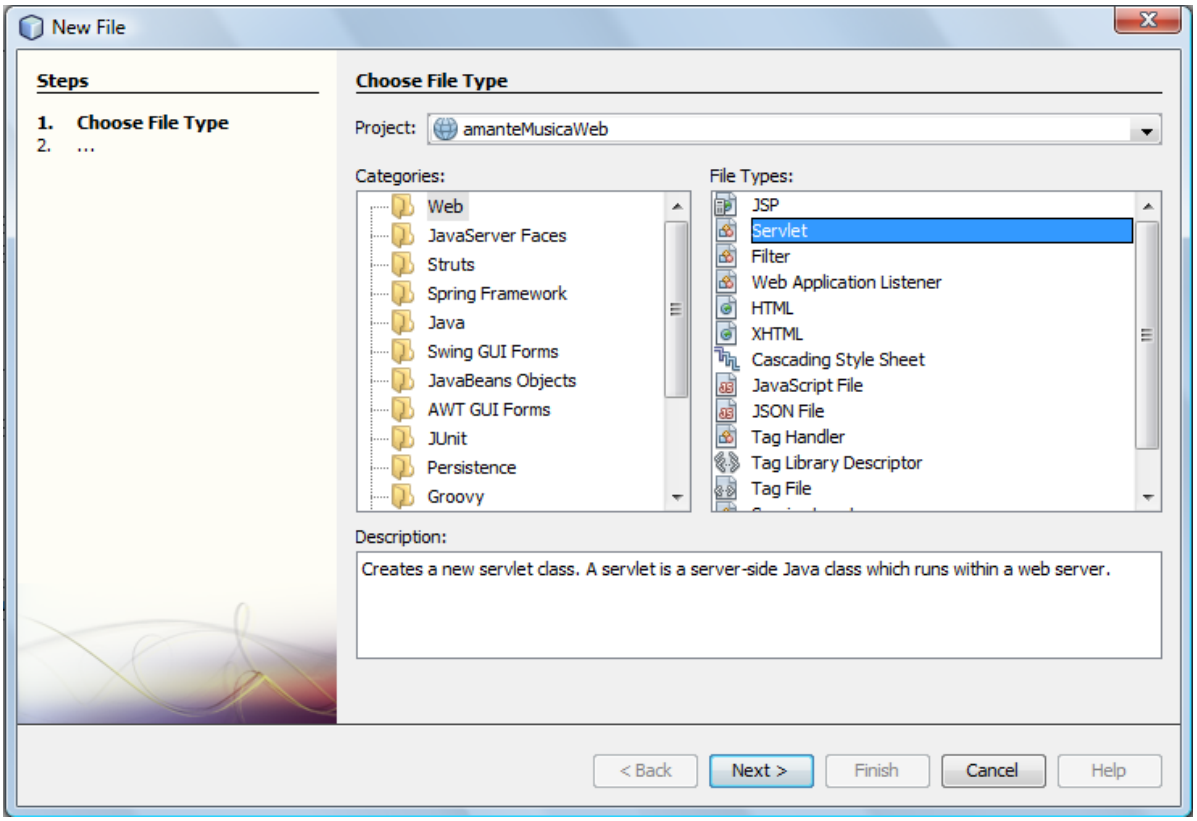

Figura 36

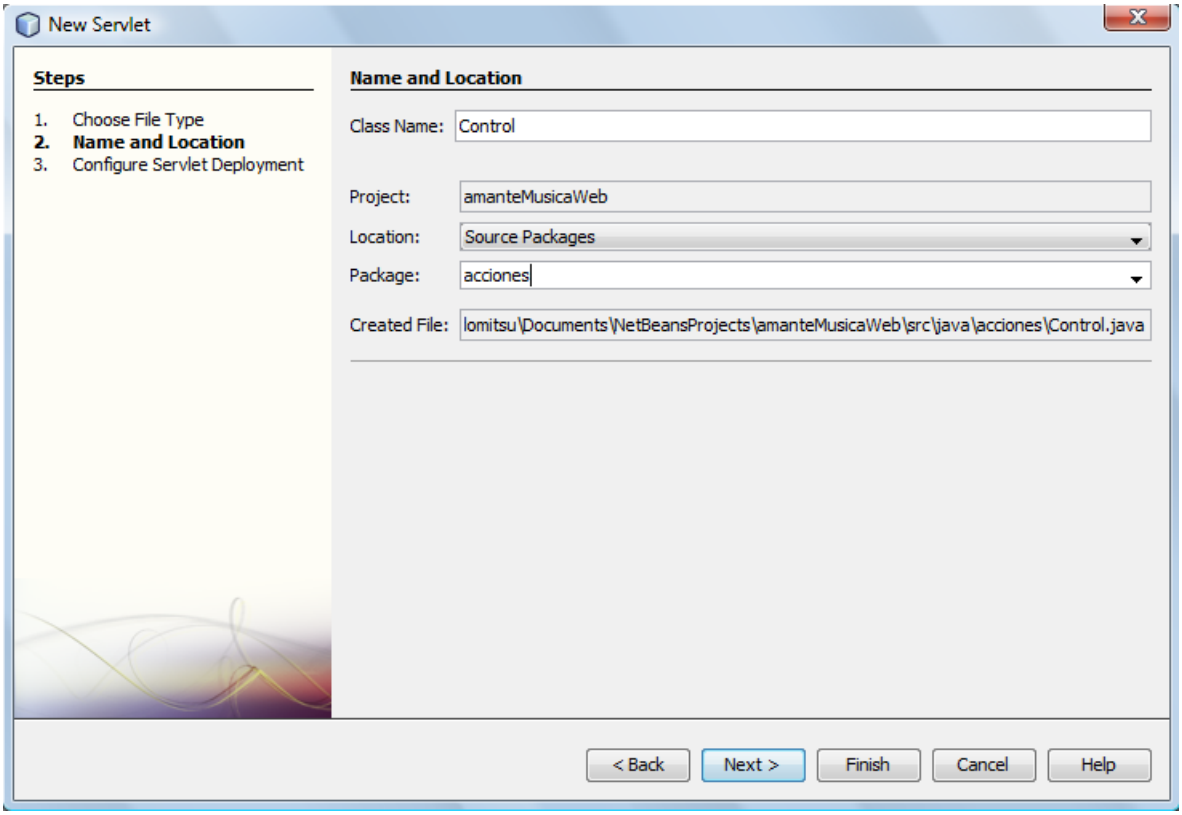

Figura 37

4. Aparecerá el tercer cuadro de diálogo del asistente para crear un archivo, figura 38. En este cuadro de diálogo se creará una entrada para este servlet en el descriptor de despliegue de la aplicación Web, web.xml.

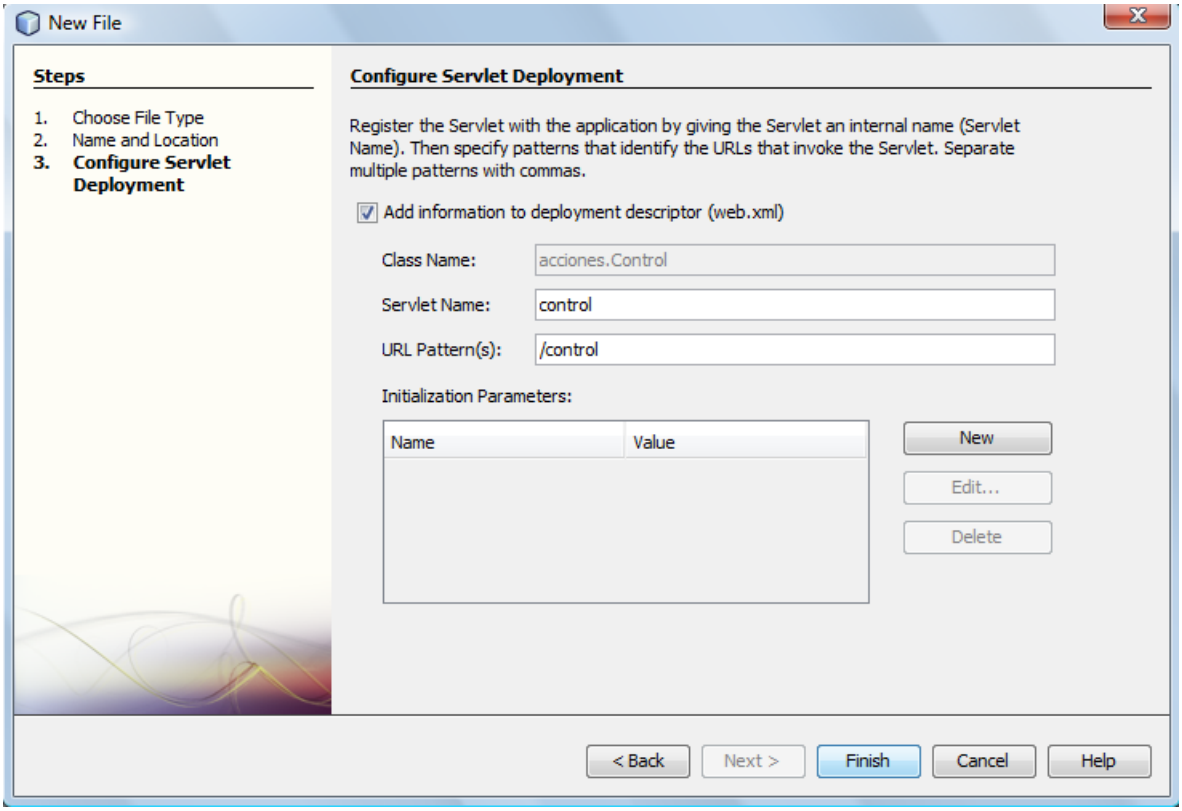

Figura 38

a) Asegúrese que la casilla de verificación **Add information to deployment descriptor** este seleccionada paa que se genere la entada para este servlet.
b) En el campo de texto **Servlet Name**, modifique el nombre del servlet del valor por ausencia, que es el nombre de la clase, cambiando la mayúscula a minúscula para mantener la convención de que los nombres de las variables y referencias van en minúsculas. Por ejemplo, "**control**".
c) En el campo de texto **URL Pattern(s)**, también modifique el URL del valor por ausencia, que es el nombre de la clase, cambiando la mayúscula a minúscula. Por ejemplo "**/control**". No borre la diagonal (/). Este URL es el que se usará para invocar a este servlet.
d) Presione el botón **Finish**.

2. Desaparecerá el asistente para crear un nuevo archivo y aparecerá el esqueleto del servlet creado. También puede verse en el árbol del proyecto que se creo la clase del servlet, figura 39.

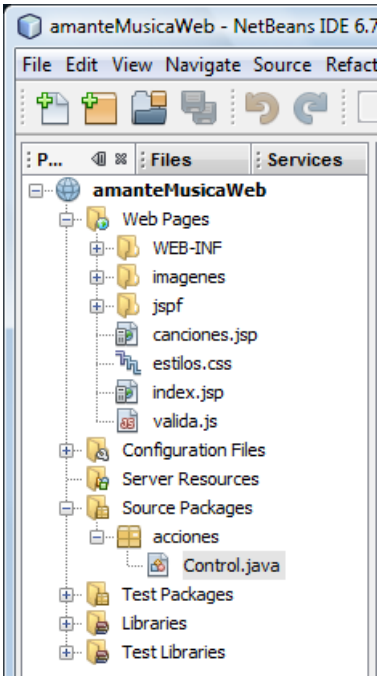

Figura 39

3. Si inspeccionamos el archivo **web.xml** veremos que se creó la siguiente entrada para el servlet:

```
<servlet>
    <servlet-name>control</servlet-name>
    <servlet-class>acciones.Control</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>control</servlet-name>
    <url-pattern>/control</url-pattern>
</servlet-mapping>
```

   Aquí se establece el nombre del servlet, el nombre de su clase y su URL.

4. Edite el servlet **Contol.java** para que su código sea el mostrado en el listado siguiente:

Control.java

```
/*
 * Control.java.
 */
package acciones;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
/**
 * Este servlet selecciona la página JSP o servlet inicial del
 * caso de uso seleccionado de la aplicación AmanteMusica
 * @author mdomitsu
 */
public class Control extends HttpServlet {

    /**
     * Procesa las solicitudes para ambos metodos HTTP:
     * GET y POST.
     * @param request Objeto request del servlet
     * @param response Objeto response del servlet
     * @throws ServletException Si ocuurre un error especifico del
     * servlet.
     * @throws IOException Si ocuurre un error de E/S
     */
    protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        // Obtén de la solicitud, la tarea seleccionada del menú de
        // canciones
        String tareaSel = request.getParameter("tarea");
        String siguiente = null;

        // Guarda la tarea seleccionada como el atributo tareaSel en la
        // variable session que es la que contiene a todas las variables
        // con ámbito de sesion
        HttpSession session = request.getSession();
        session.setAttribute("tareaSel", tareaSel);

        // establece la pagina JSP o servlet que inicia
        // el caso de uso seleccionado
        if (tareaSel.equals("agregarCancion")) {
            siguiente = "capturaClave.jsp";
        } else if (tareaSel.equals("actualizarCancion")) {
            siguiente = "obtenCanciones.jsp";
        } else if (tareaSel.equals("eliminarCanciones")) {
            siguiente = "obtenCanciones.jsp";
        } else if (tareaSel.equals("listarCanciones")) {
            siguiente = "obtenCanciones";
        } else if (tareaSel.equals("listarCancionesGenero")) {
            siguiente = "obtenGenerosCanciones.jsp";
        } else if (tareaSel.equals("listarCancionesPeriodo")) {
            siguiente = "capturaPeriodo.jsp";
        }

        // Redirecciona a la página JSP o servlet
        request.getRequestDispatcher(siguiente).forward(request,
                                                        response);
    }

    /**
     * Maneja el método HTTP: GET.
     * @param request Objeto request del servlet
     * @param response Objeto response del servlet
     * @throws ServletException Si ocuurre un error especifico del
     * servlet.
```

```
     * @throws IOException Si ocuurre un error de E/S
     */
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Maneja el método HTTP: POST.
     * Maneja el método HTTP: GET.
     * @param request Objeto request del servlet
     * @param response Objeto response del servlet
     * @throws ServletException Si ocuurre un error especifico del
     * servlet.
     * @throws IOException Si ocuurre un error de E/S
     */
    @Override
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Regresa una descripcion corta del servlet.
     * @return Descripcion corta del servlet.
     */
    @Override
    public String getServletInfo() {
        return "Servlet de control";
    }// </editor-fold>
}
```

5. Para probar el funcionamiento del servlet cree otra página JSP llamada **capturaClave** y edita su contenido para que su código sea el mostrado en el listado siguiente:

capturaClave.jsp

```
<%--
    Document   : capturaClave
    Created on : 13/04/2012, 01:05:40 PM
    Author     : mdomitsu

    Esta página JSP permite capturar la clave de una canción a agregar
    al catálogo de canciones. Forma parte de la aplicación AmanteMusica
    versión JSP
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!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=UTF-8">
   <title>Amante Música - Versión JSP: Captura Clave</title>
   <link rel="stylesheet" type="text/css" href="estilos.css" />
   <script type="text/javascript" src="valida.js">
   </script>
   <script type="text/javascript">
     var cancelar = false;

     function validaEnviar() {
       var errores=0;
       if (cancelar) {
         location.href="canciones.jsp";
         return true;
       }
       else{
         // Ejecuta las validaciones en orden inverso para que el
         // elemento con el enfoque sea el primero con error
         if(!validaClave(document.forms.capturaClave.clave,
            'msjClave'))
           errores += 1;
           return (errores==0);
        }
      };
  </script>
</head>

<body>
   <%-- Incluye la barra de título --%>
  <%@include file="jspf/titulo.jspf"%>

  <%-- Incluye el menú --%>
  <%@include file="jspf/menuCanciones.jspf"%>

  <div class="principal">
    <div class="contenido">
      <h1>Canción a agregar</h1>
      <%-- Formulario para capturar la clave y enviarla a la página
           para obtener la canción --%>
      <form action="obtenCancion.jsp" method="post" name="capturaClave"
            onSubmit="return validaEnviar()">
        <table class="centrada">
          <tr>
            <td>Clave *</td>
            <td><input type="text" name="clave" size="7"
                       onchange="validaClave(this, 'msjClave');" /></td>
            <td id="msjClave"> </td>
          </tr>
        </table>
        <br />

        <table class="centrada" width=50%>
          <tr>
            <%-- Botones enviar --%>
            <td><input type="submit" name="boton" value="Continuar"
                       onclick="cancelar=false"/></td>
            <td><input type="submit" name="boton" value="Cancelar"
```

```
                        onclick="cancelar=true" /></td>
              <%-- Botón limpiar --%>
              <td><input type="reset" value="Limpiar" /></td>
            </tr>
          </table>
        </form>
      </div>
    </div>
  </body>
</html>
```

6. Guarde la página JSP y ejecute la aplicación. Aparecerá la página inicial de la aplicación, figura 40.

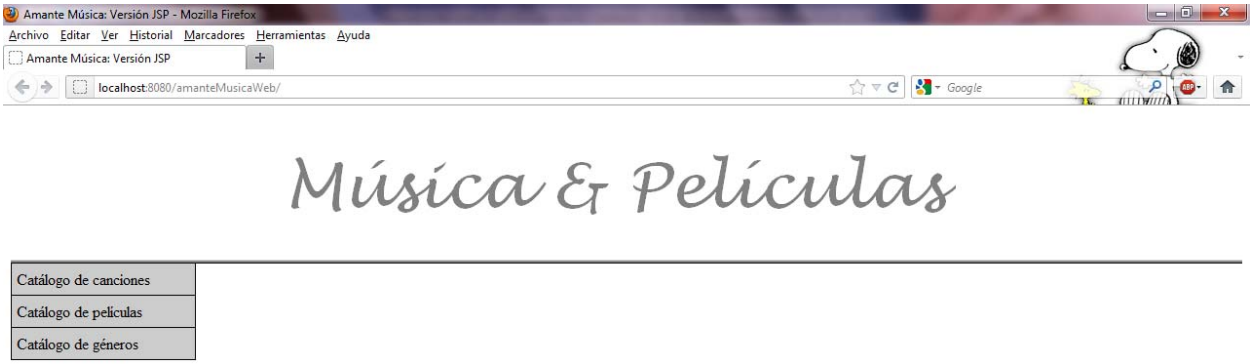

Figura 40

7. Si hacemos clic en la opción **Catálogo de canciones**, se invocará a la página JSP **canciones.jsp** la cual desplegará un menú con las operaciones que se pueden hacer con las canciones, figura 41.

8. Si hacemos clic en la opción **Agregar Canción**, se invocará al servlet Control.java pasándole como parámetro "**tarea=agregarCancion**". El servlet utiliza el valor del parámetro para determinar a qué página JSP o servlet se invocará. En este caso a la página JSP capturarClave.jsp, figura 42.

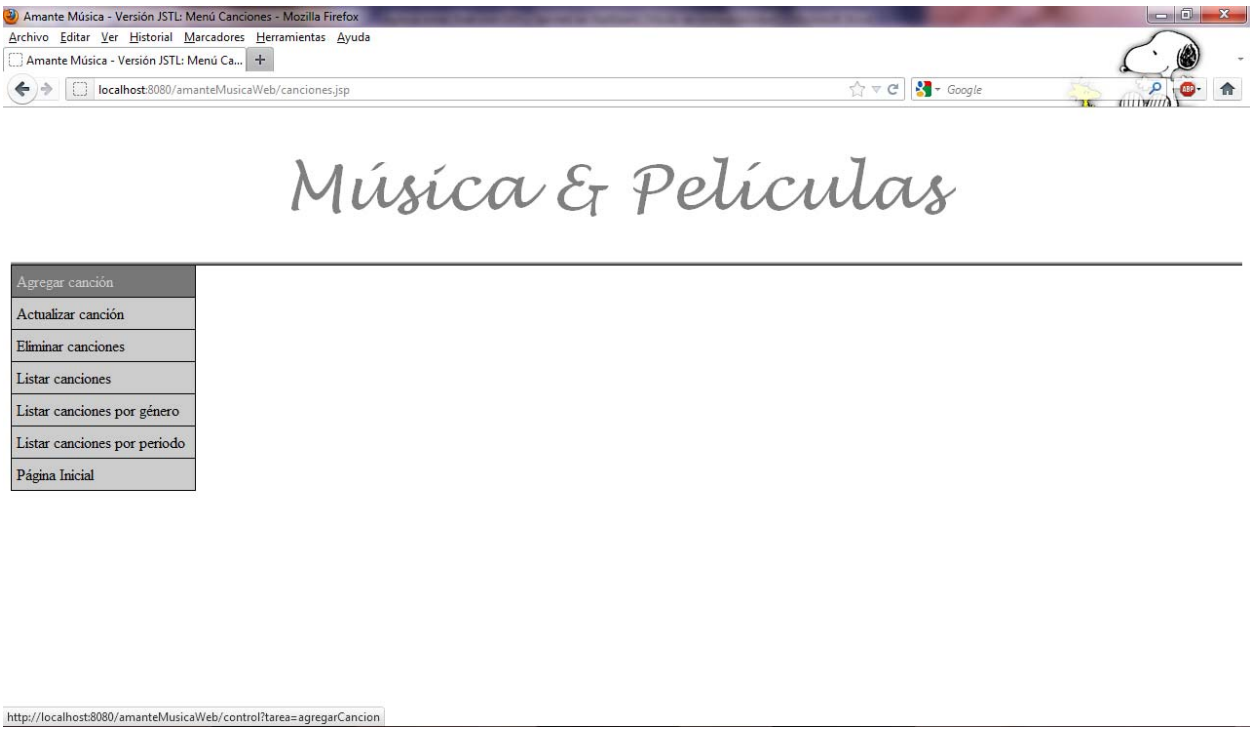

Figura 41

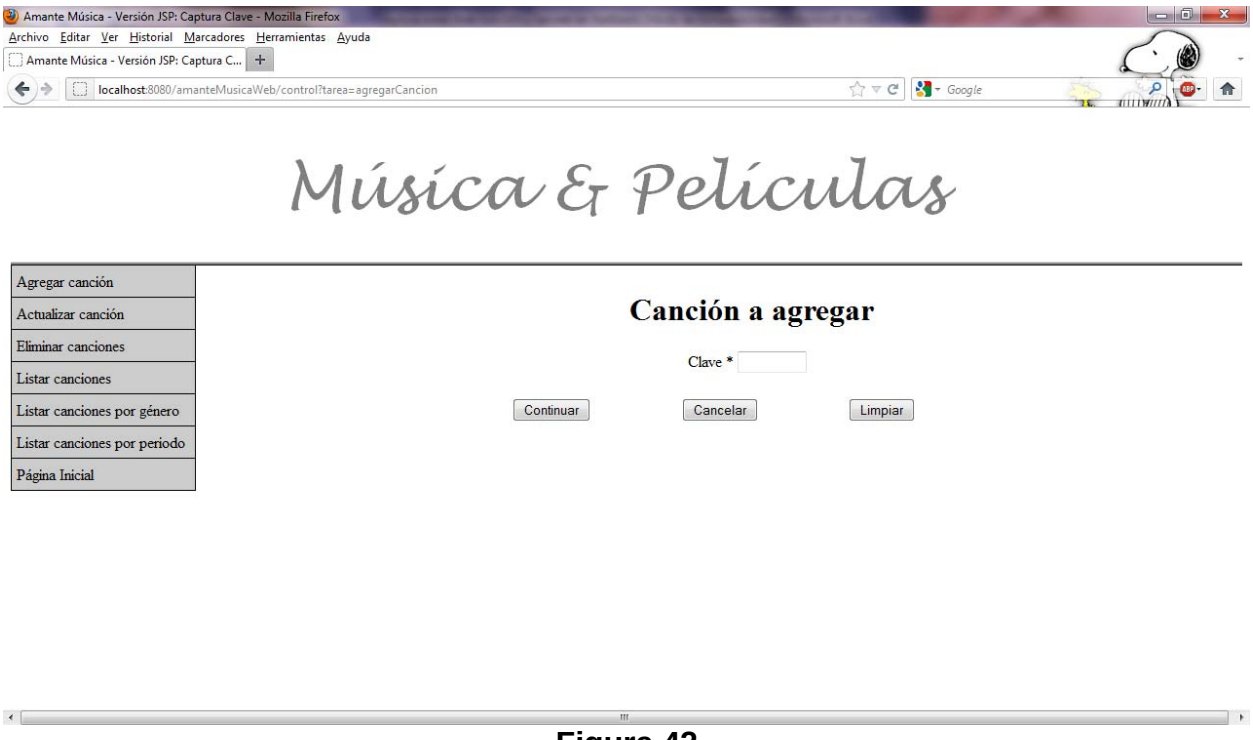

Figura 42

Configuración de una Página JSP de Error para una Aplicación Web

En una aplicación Web, podemos establecer una o más páginas JSP de error. Cada página será invocada cuando en la aplicación ocurra una excepción de un tipo o de sus derivadas.

El nombre de la página JSP de error y la excepción que la invoca se establece en el Descriptor de Despliegue de una Aplicación Web, el archivo web.xml. Para configurar una página JSP de error se sigue el siguiente procedimiento:

1. Haga clic en el nodo del archivo web.xml en el árbol del proyecto, figura 10. Aparecerá el asistente que nos permite inspeccionar/modificar el contenido de ese archivo, figura 11.

2. Haga clic en la pestaña **Pages** para acceder al asistente que permite configurar el nombre del archivo inicial de la aplicación, las páginas de error y los Grupos de Propiedades JSP, figura 43. Haga clic en el nodo Error Pages para expandirlo.

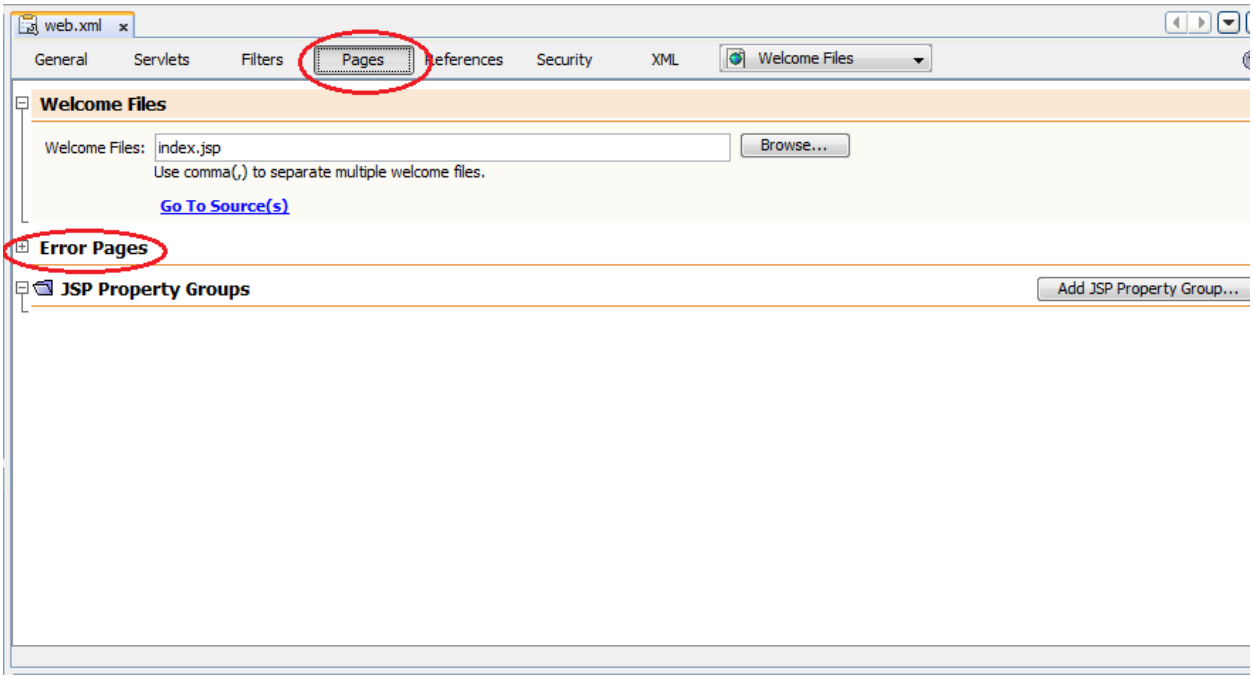

Figura 43

3. Aparecerá un recuadro que nos permite agregar, editar y eliminar una página JSP de error y la excepción que la invoca, figura 44.

4. Para agregar una página de error haga clic en el botón **Add**. Aparecerá el cuadro de diálogo de la figura 45.

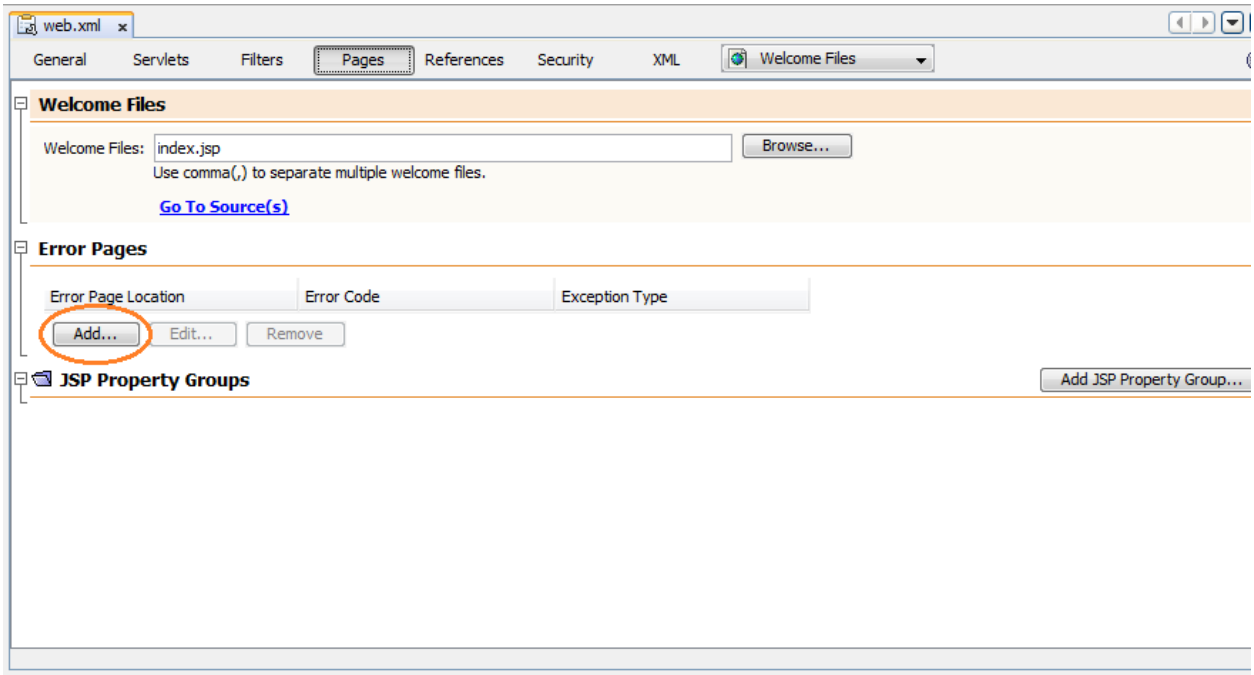

Figura 44

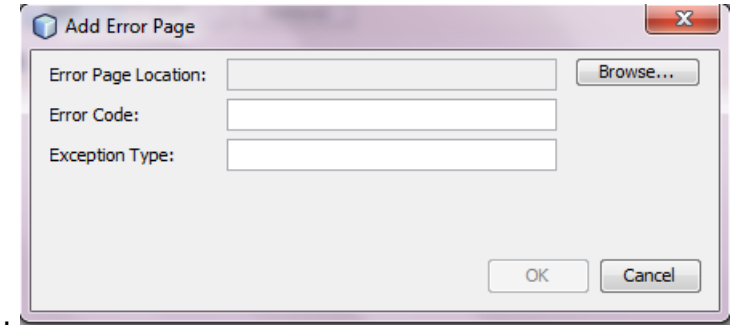

Figura 45

5. Para establecer la página JSP de error haga clic en el botón **Browse…**. Aparecerá el cuadro de diálogo para seleccionar la página JSP, figura 46. Navegue el árbol esta localizar el nodo de la página deseada. Selecciónela y haga clic en el botón **Select File**.

6. Reaparecerá el cuadro de diálogo de la figura 44, con un mensaje indicando que falta el código de error o la excepción que invocará a la página JSP de error, figura 47.

7. Escriba el nombre completamente calificado (con el nombre del paquete en que se encuentra) en el campo de texto **Exception Type**, `java.lang.RuntimeException`, en este ejemplo, y haga clic en el botón **OK**.

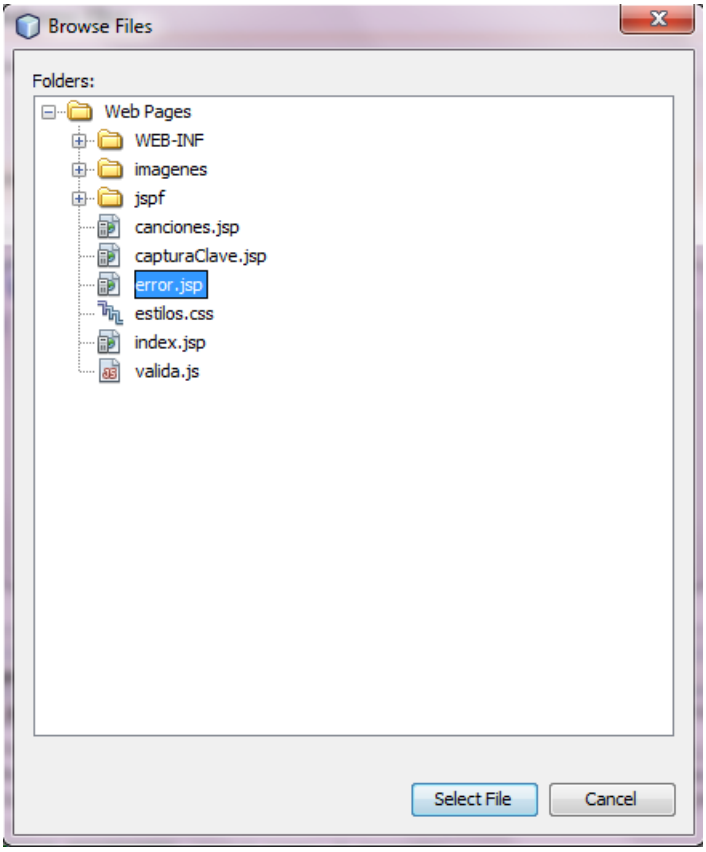

Figura 46

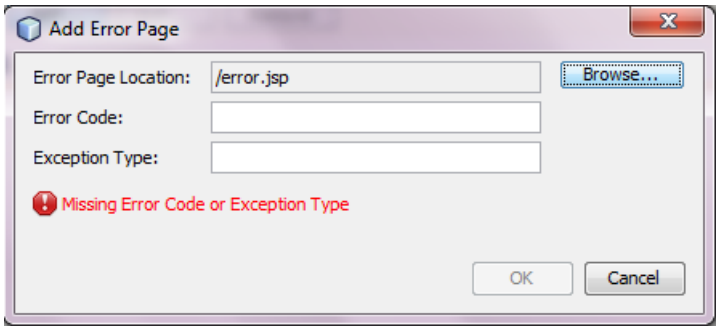

Figura 47

8. Reaparecerá el cuadro de diálogo de la figura 43 con una tabla con los datos de la página de error y excepción configurados, figura 48. Guarde el documento **web.xml**.

9. Podemos inspeccionar la entrada que el asistente hizo en el archivo haciendo clic en la pestaña **XML**, figura 12, veremos lo siguiente:

```
<error-page>
    <exception-type>java.lang.RuntimeException</exception-type>
    <location>/error.jsp</location>
</error-page>
```

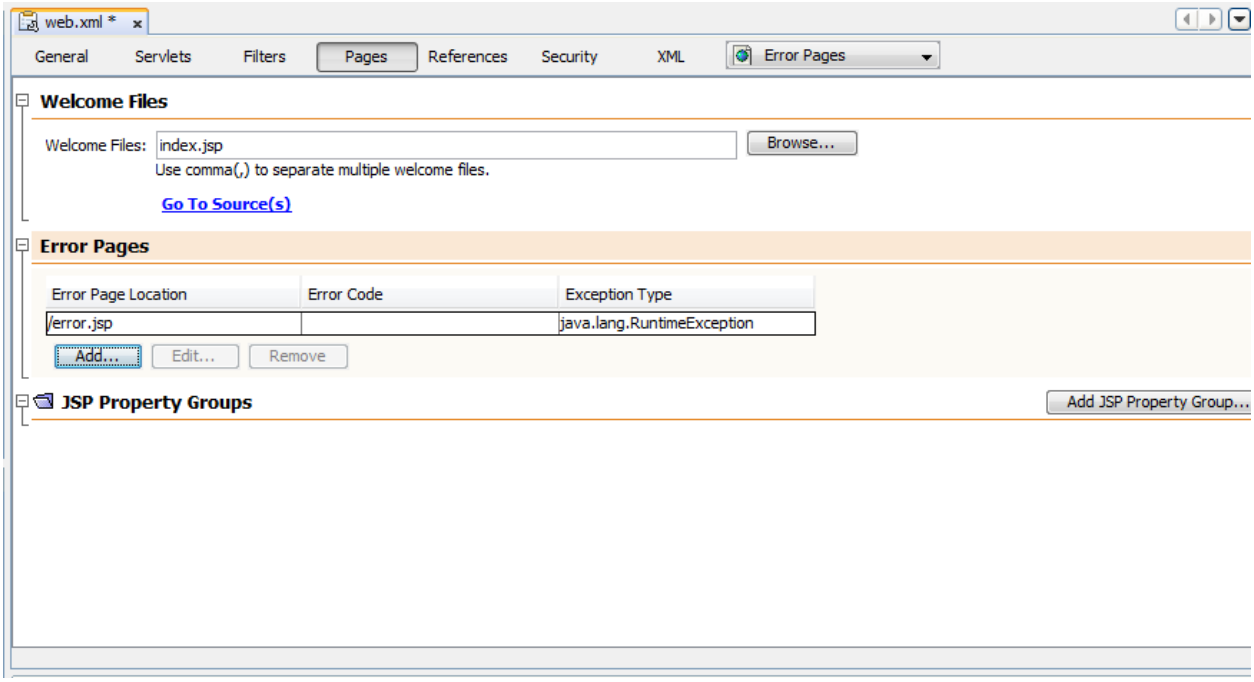
web.xml *
General
Servlets
Filters
Pages
References
Security
XML
Error Pages
Welcome Files
Welcome Files:
index.jsp
Browse...
Use comma(,) to separate multiple welcome files.
Go To Source(s)
Error Pages
Error Page Location
Error Code
Exception Type
/error.jsp
java.lang.RuntimeException
Add...
Edit...
Remove
JSP Property Groups
Add JSP Property Group...

Figura 48