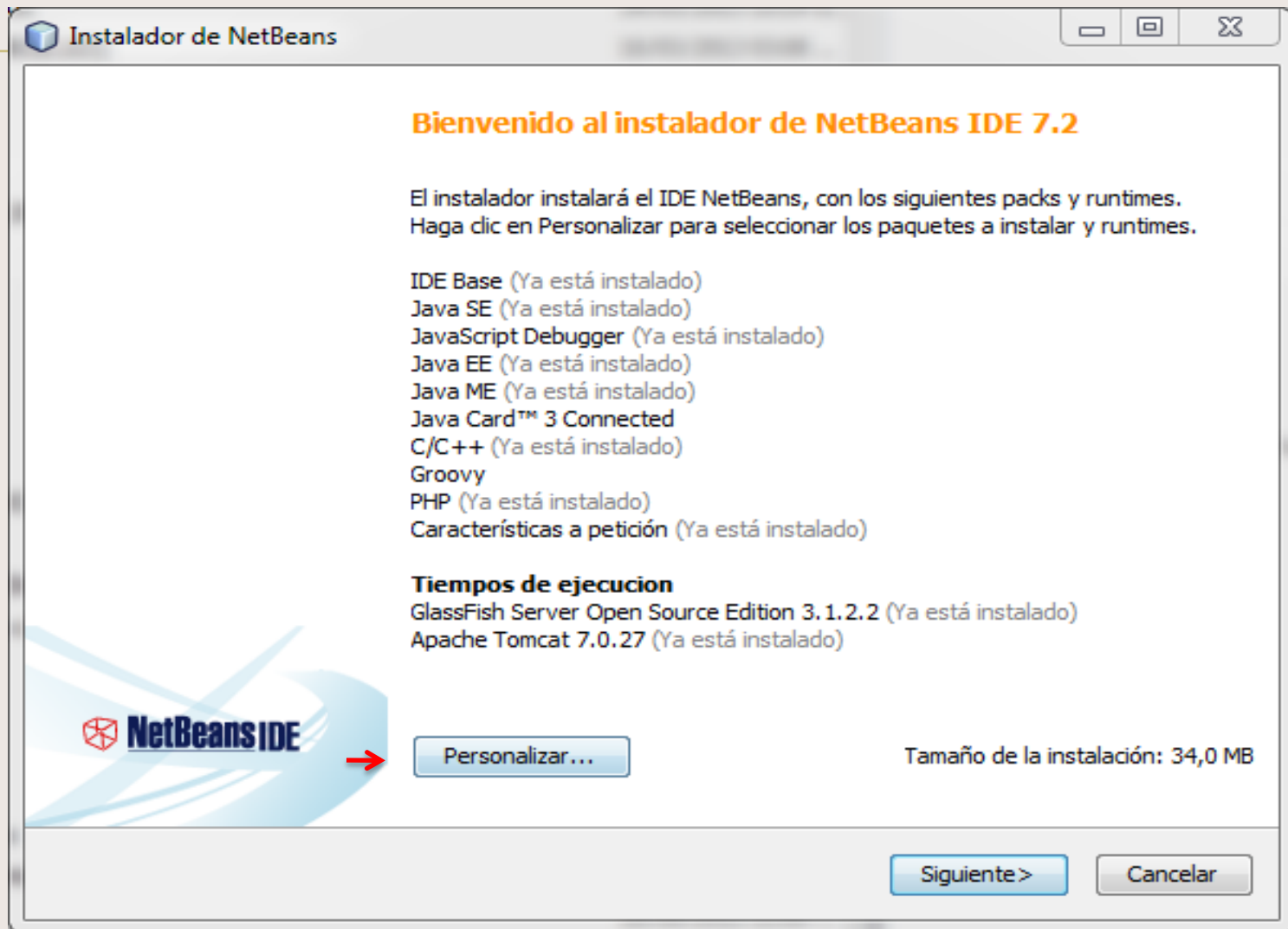


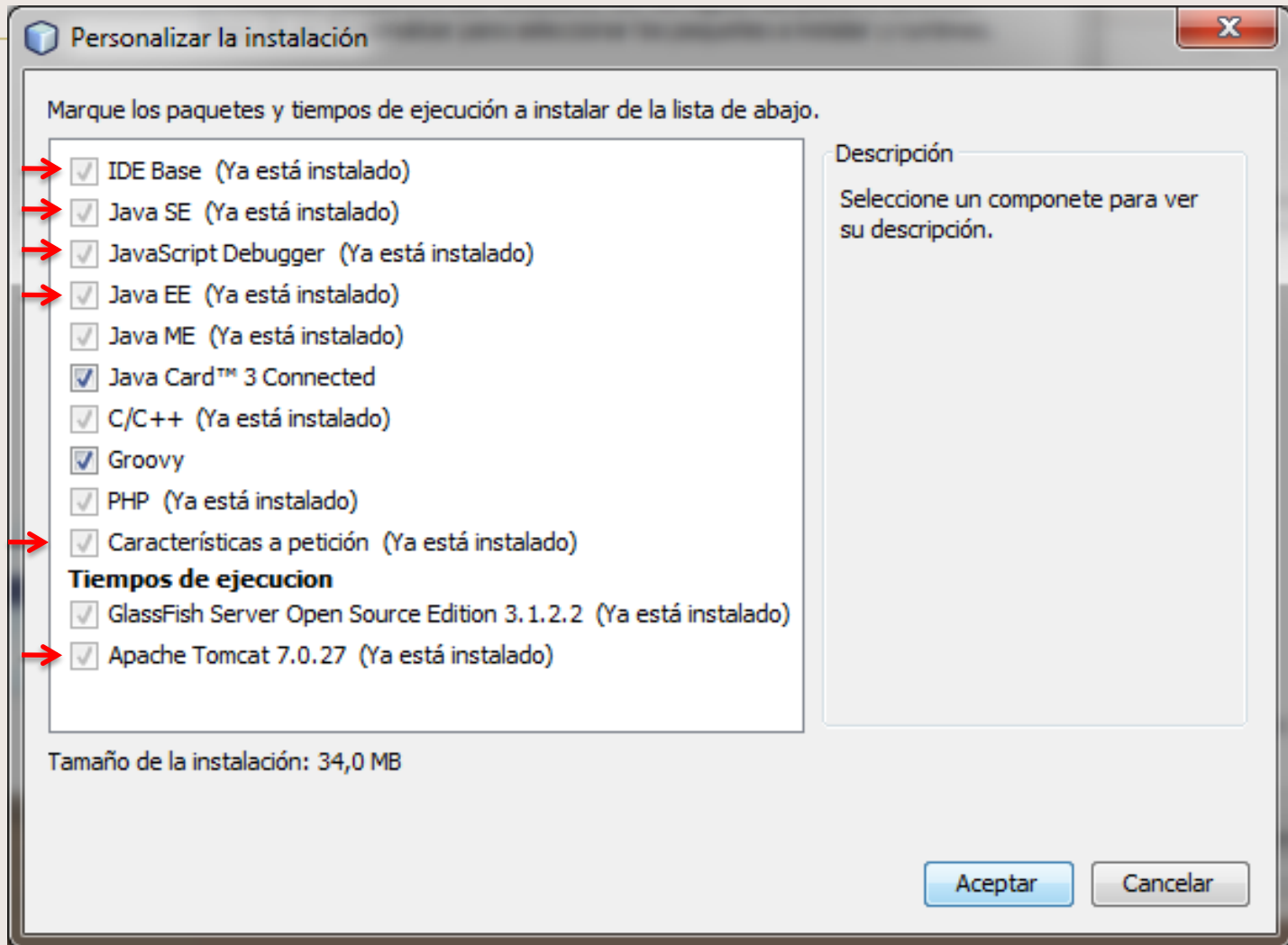
JSP y SERVLET JAVA

John Carlos Arrieta Arrieta

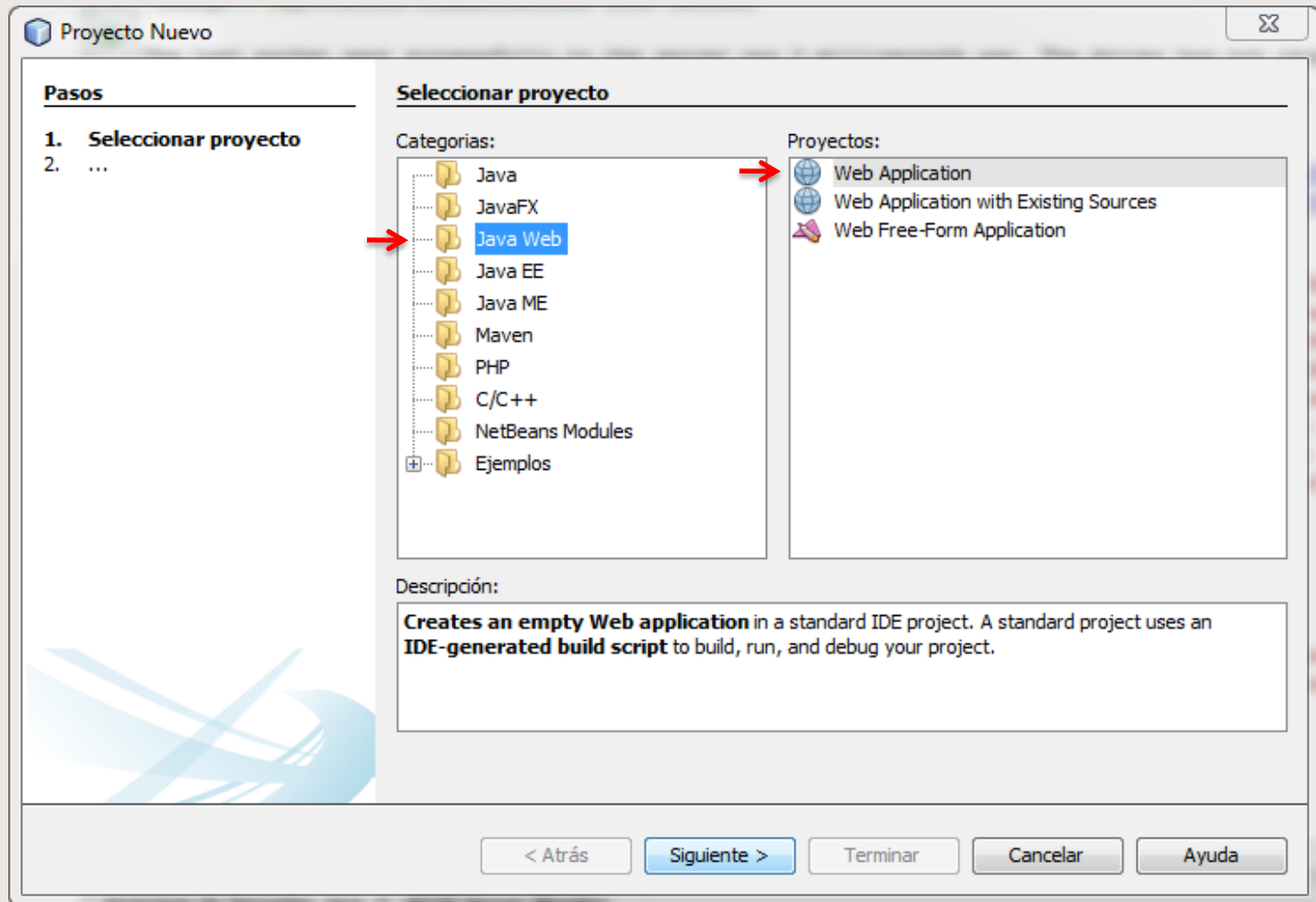
Instalar IDE Netbeans y el Tomcat



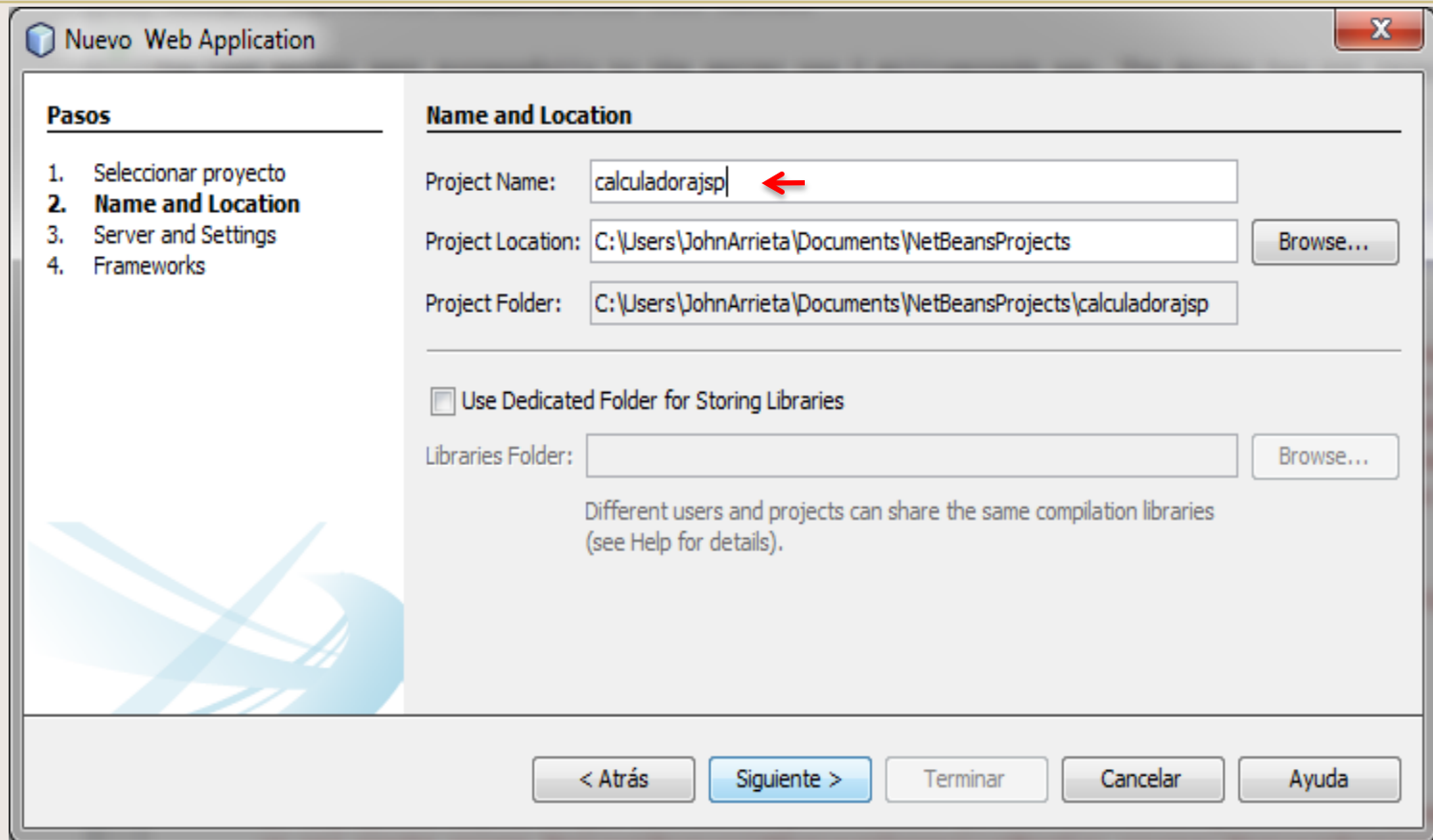
Marcar mínimo estas opciones



Crear un Proyecto Web en Netbeans



Crear un Proyecto Web en Netbeans



Nuevo Web Application

Pasos

1. Seleccionar proyecto
- 2. Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name: ←

Project Location: Browse...

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

< Atrás Siguiete > Terminar Cancelar Ayuda

Crear un Proyecto Web en Netbeans

Nuevo Web Application

Pasos

1. Seleccionar proyecto
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server: → Apache Tomcat 7.0.27.0 Add...

Java EE Version: Java EE 6 Web

☐ Enable Contexts and Dependency Injection

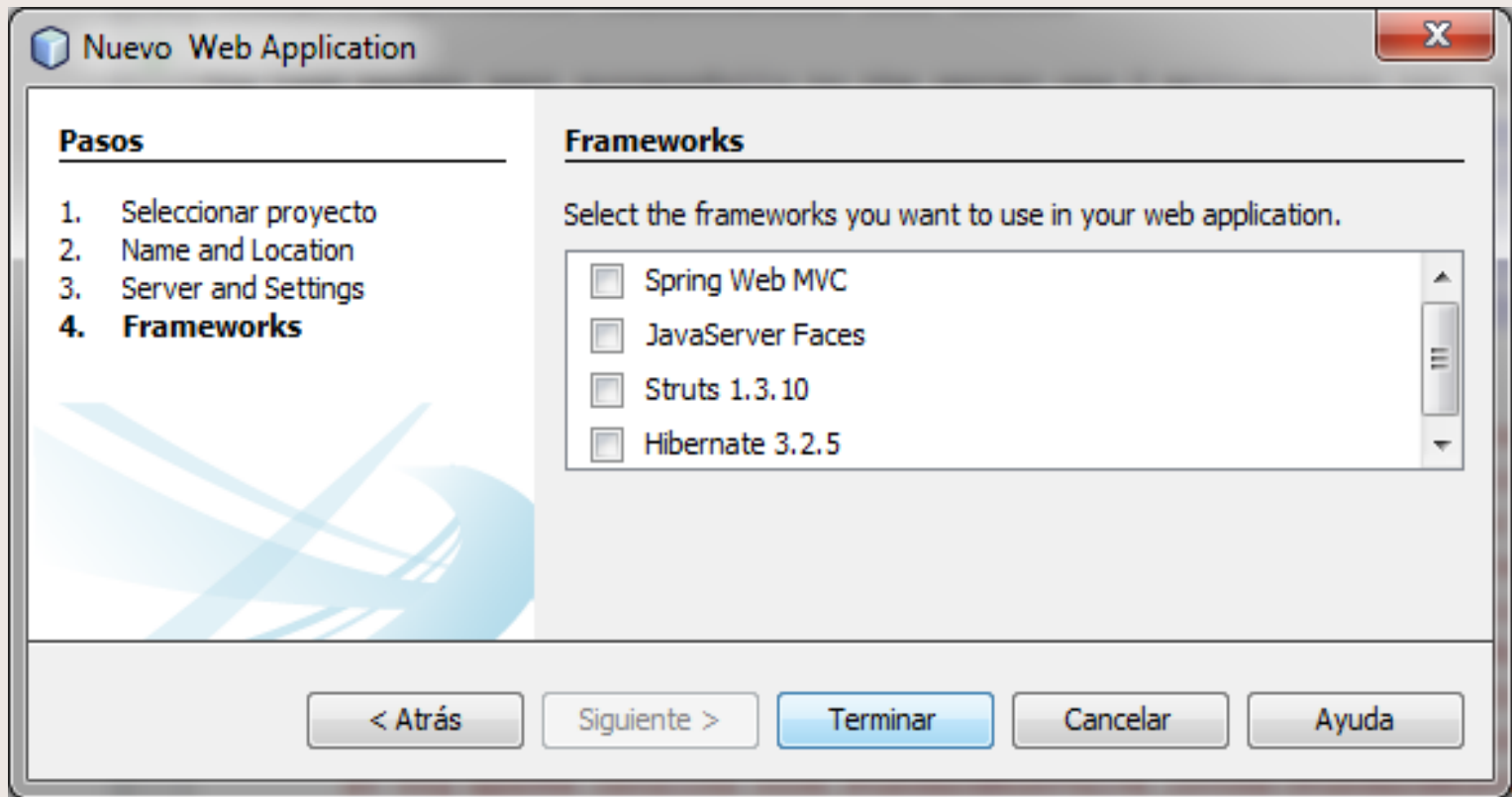
☒ Set Source Level to 6

Recommendation: Source Level 6 should be used in Java EE 6 projects.

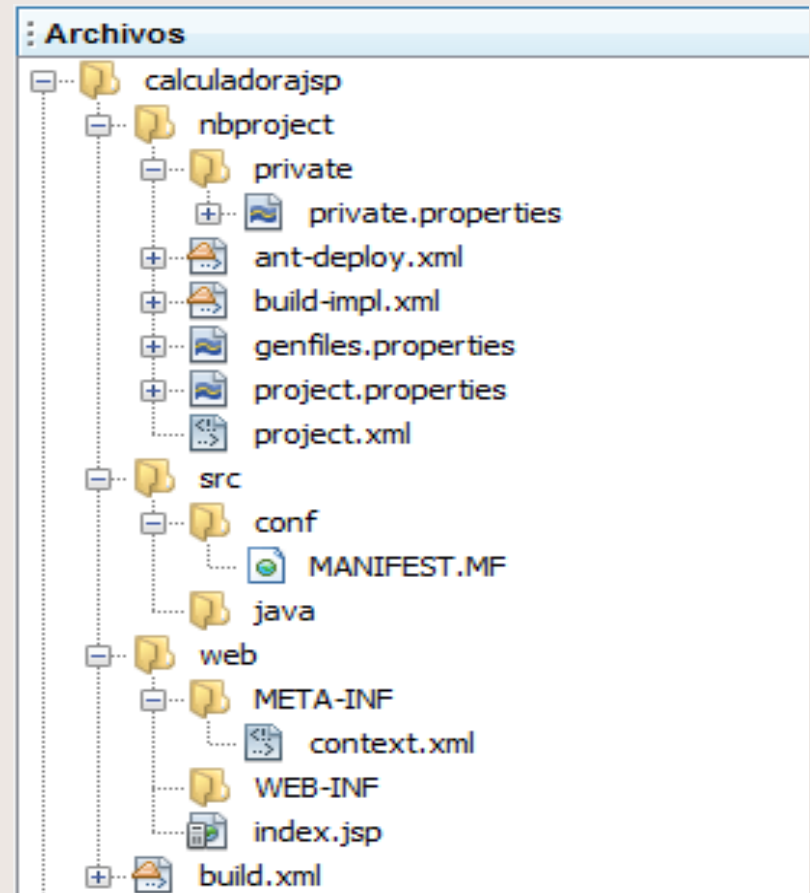
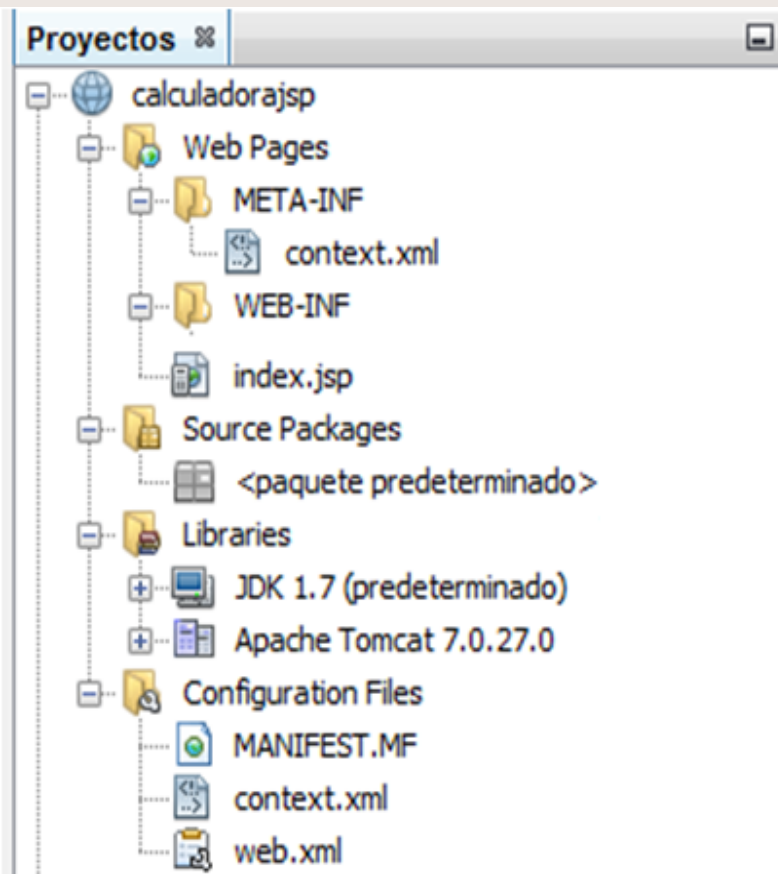
Context Path: /calculadorajsp ←

< Atrás Siguiete > Terminar Cancelar Ayuda

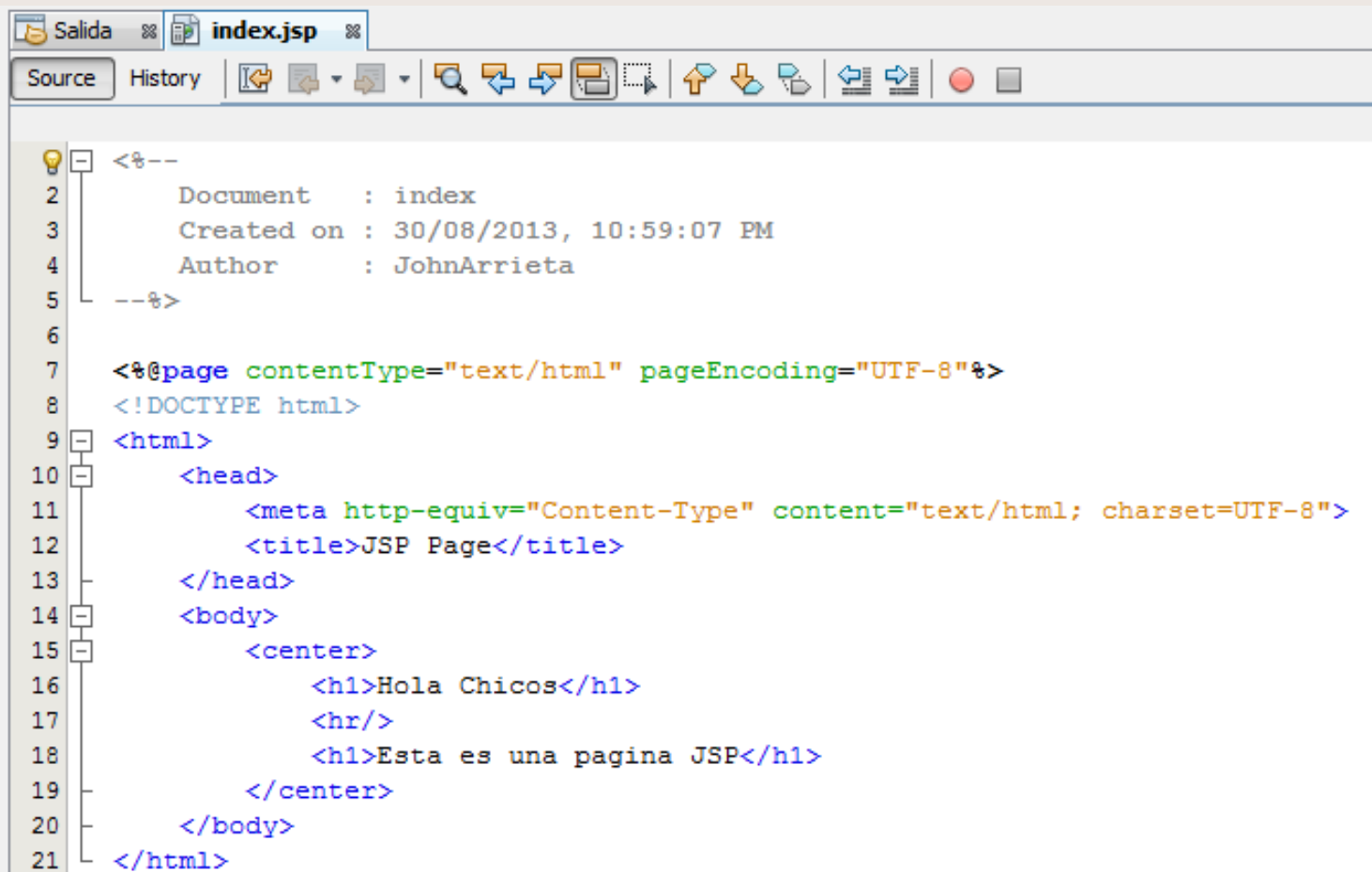
Crear un Proyecto Web en Netbeans



Estructura del Proyecto y sus archivos básicos



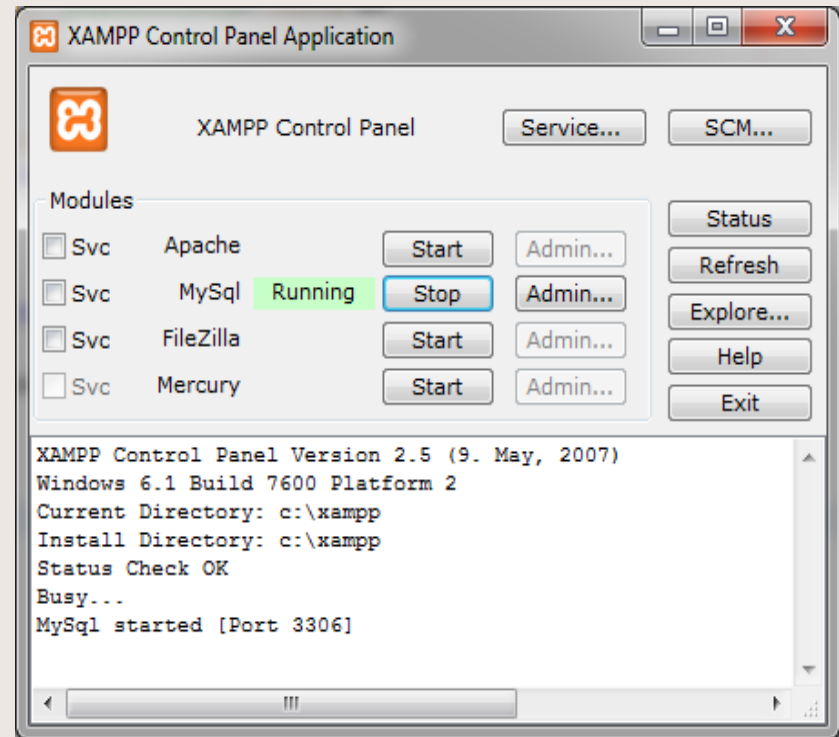
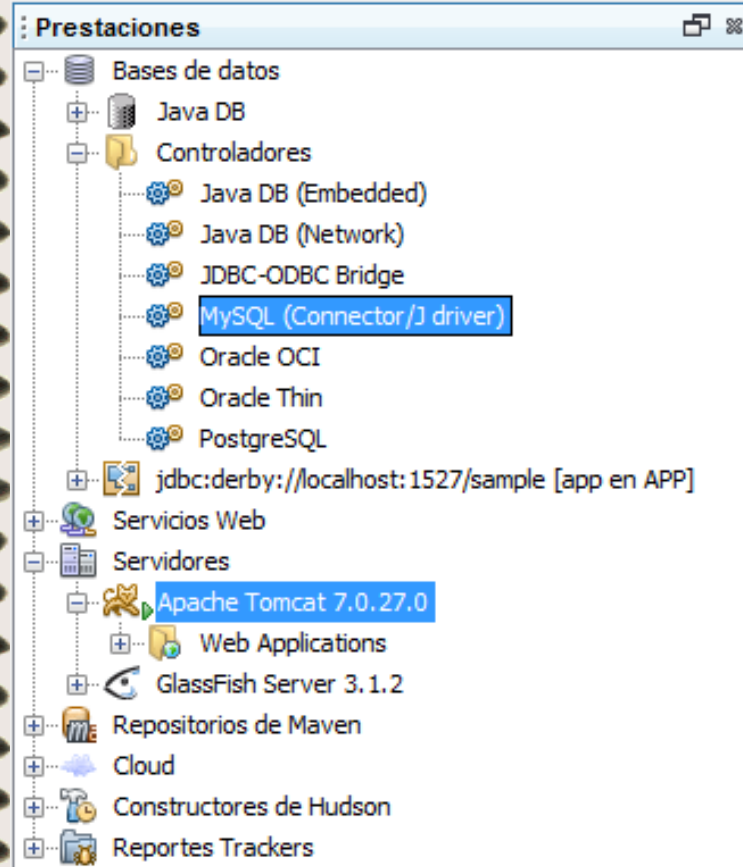
Contenido básico de la pagina web inicial

A screenshot of a web development IDE window titled 'index.jsp'. The window has a menu bar with 'Salida' and a toolbar with various icons. Below the toolbar is a tab labeled 'Source'. The main area displays the source code of a JSP file. The code includes a comment block with document information, a page directive for UTF-8 encoding, and an HTML document structure with a head section containing a meta tag and a title, and a body section with centered text.

```
1 <%--  
2     Document    : index  
3     Created on  : 30/08/2013, 10:59:07 PM  
4     Author     : JohnArrieta  
5 --%>  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10     <head>  
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12         <title>JSP Page</title>  
13     </head>  
14     <body>  
15         <center>  
16             <h1>Hola Chicos</h1>  
17             <hr/>  
18             <h1>Esta es una pagina JSP</h1>  
19         </center>  
20     </body>  
21 </html>
```

John Carlos Arrieta Arrieta

Iniciar los servicios necesarios para compilar y ejecutar el proyecto



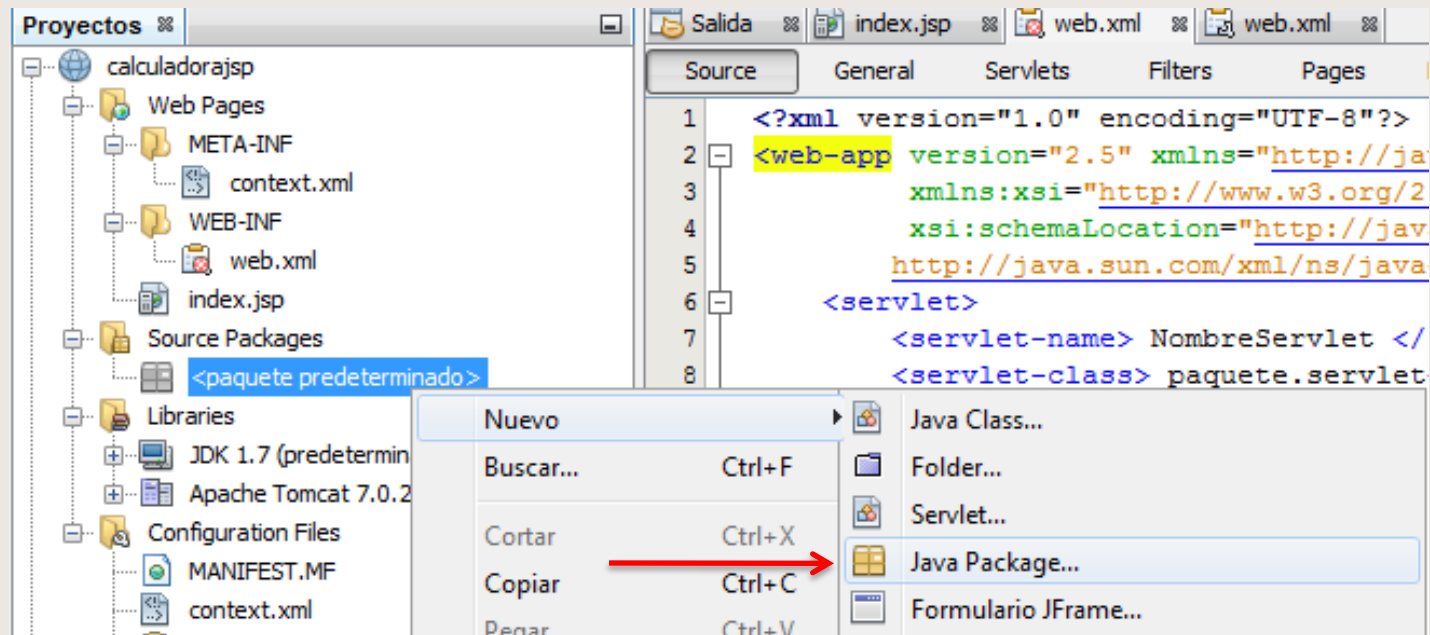
John Carlos Arrieta Arrieta

El proyecto basico ejecutandose



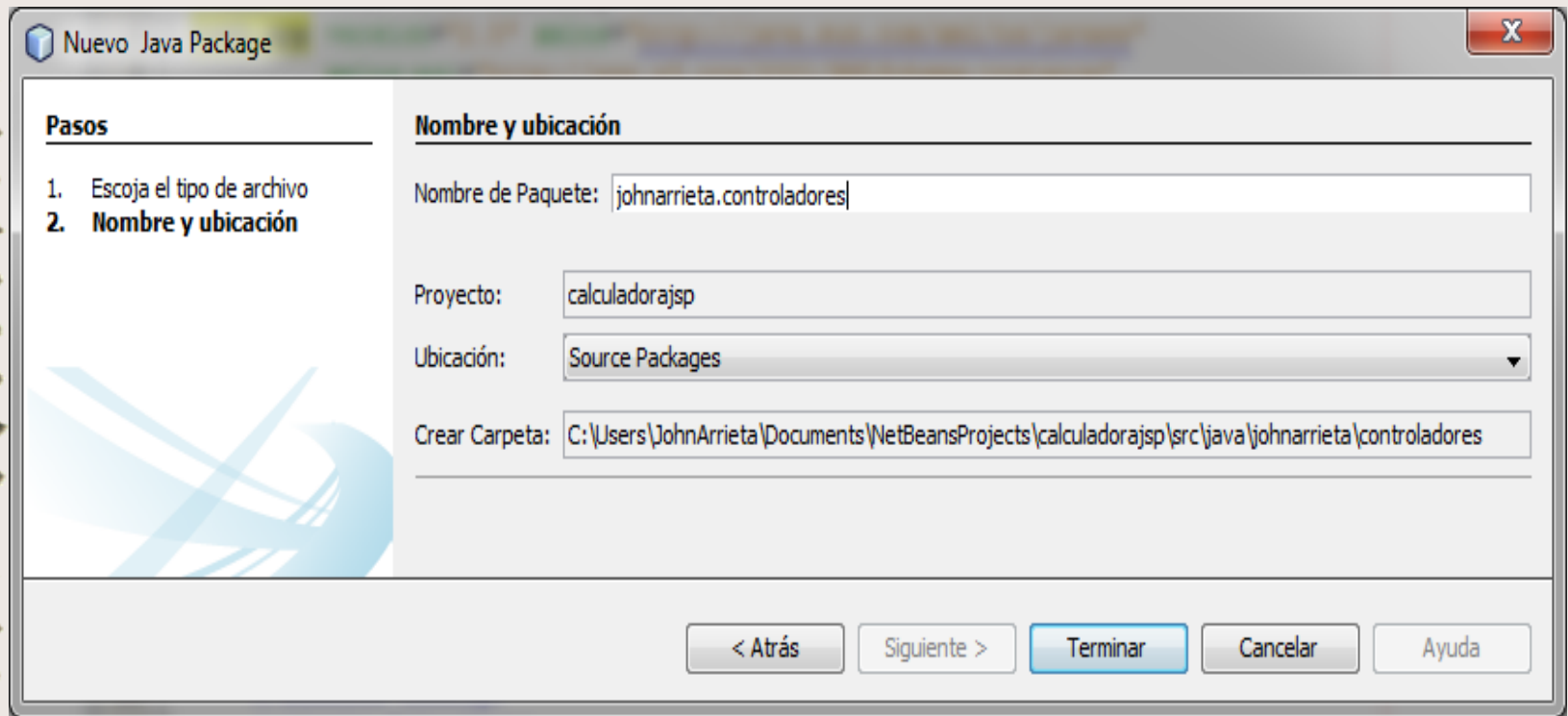
John Carlos Arrieta Arrieta

Crear un paquete



John Carlos Arrieta Arrieta

Crear un paquete



Nuevo Java Package

Pasos

1. Escoja el tipo de archivo
2. **Nombre y ubicación**

Nombre y ubicación

Nombre de Paquete:

Proyecto:

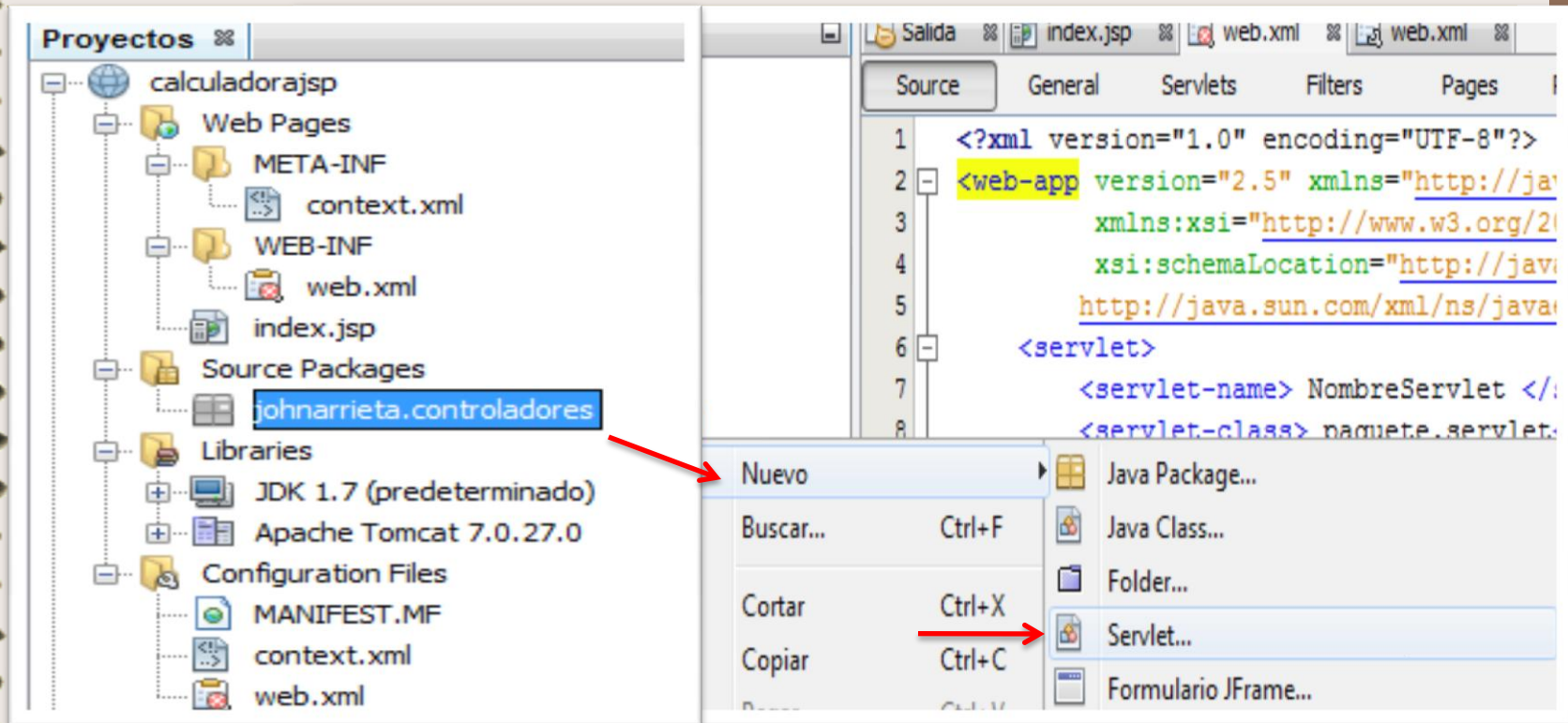
Ubicación:

Crear Carpeta:

< Atrás Siguiente > **Terminar** Cancelar Ayuda

John Carlos Arrieta Arrieta

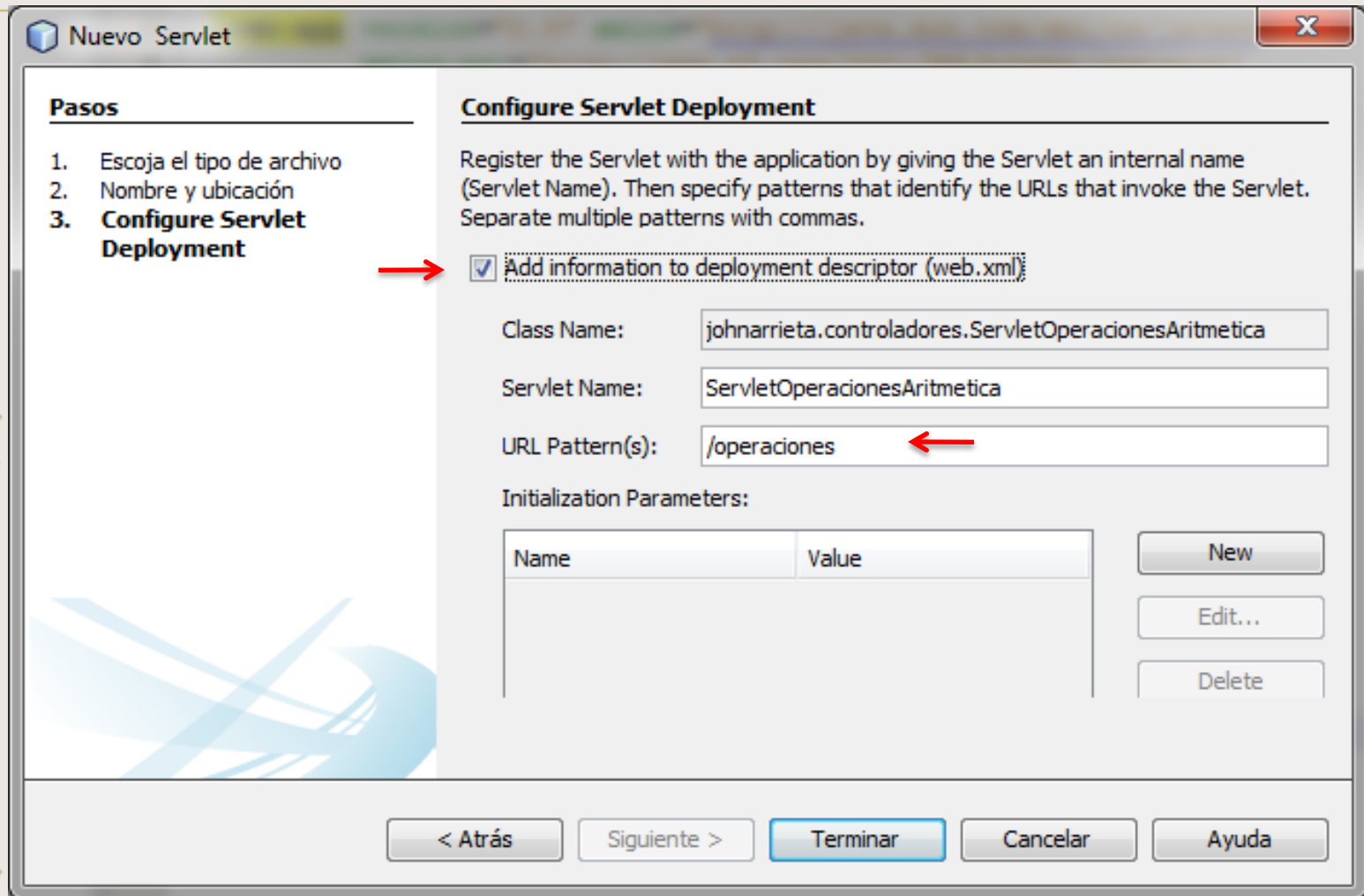
Crear un Servlet en el nuevo paquete



Crear un Servlet en el nuevo paquete

John Carlos Arrieta Arrieta

Crear un Servlet en el nuevo paquete indicar la URI del Servlet



Nuevo Servlet

Pasos

1. Escoja el tipo de archivo
2. Nombre y ubicación
3. **Configure Servlet Deployment**

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☒ Add information to deployment descriptor (web.xml)

Class Name:

Servlet Name:

URL Pattern(s):

Initialization Parameters:

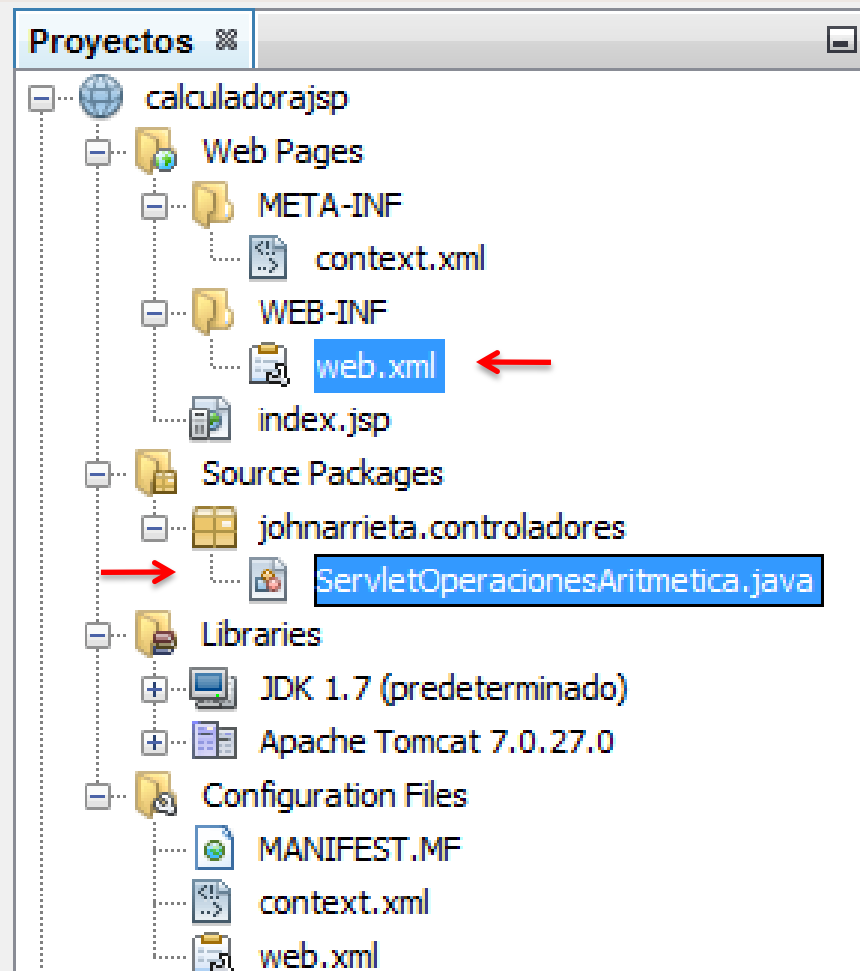
Name	Value
------	-------

New Edit... Delete

< Atrás Siguiente > **Terminar** Cancelar Ayuda

John Carlos Arrieta Arrieta

En Servlet agregado a la estructura del proyecto



John Carlos Arrieta Arrieta

El archivo XML que registra al Servlet en el Proyecto

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5     http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6
7     <servlet>
8         <servlet-name>ServletOperacionesAritmetica</servlet-name>
9         <servlet-class>johnarrieta.controladores.ServletOperacionesAritmetica</servlet-class>
10    </servlet>
11    <servlet-mapping>
12        <servlet-name>ServletOperacionesAritmetica</servlet-name>
13        <url-pattern>/operaciones</url-pattern>
14    </servlet-mapping>
15    <session-config>
16        <session-timeout>
17            30
18        </session-timeout>
19    </session-config>
20    <welcome-file-list>
21        <welcome-file>index.html</welcome-file>
22    </welcome-file-list>
23 </web-app>
```

John Carlos Arrieta Arrieta

El contenido basico de la clase ServletOperacionesAritmetica

```
1  /*
2  * Clase ServletOperacionesAritmeticas
3  */
4  package johnarrieta.controladores;
```

Paquete donde se guardara la clase

```
5
6  import java.io.IOException;
7  import java.io.PrintWriter;
8
9  import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
```

Importando las clases necesarias

```
13 public class ServletOperacionesAritmetica extends HttpServlet {
```

Los Servlet Heredan de HttpServlet

```
14
15     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
16         throws ServletException, IOException {
17         response.setContentType("text/html;charset=UTF-8");
18         PrintWriter out = response.getWriter();
19         try {
20             // el codigo de la respuesta a la peticion va a aqui
21         } finally {
22             out.close();
23         }
24     }
```

Los Servlet Heredan de HttpServlet

```
25
26     @Override
```

```
27     protected void doGet(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29         processRequest(request, response);
30     }
```

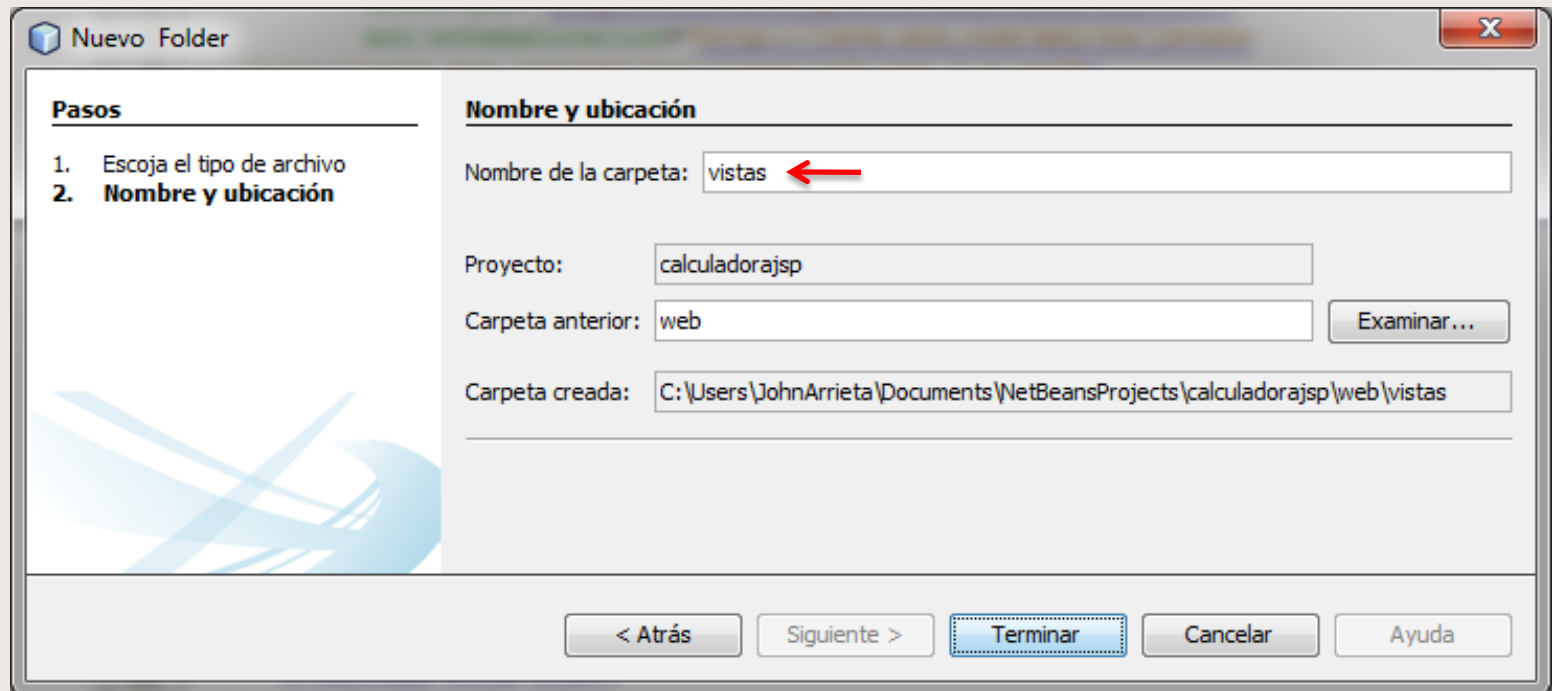
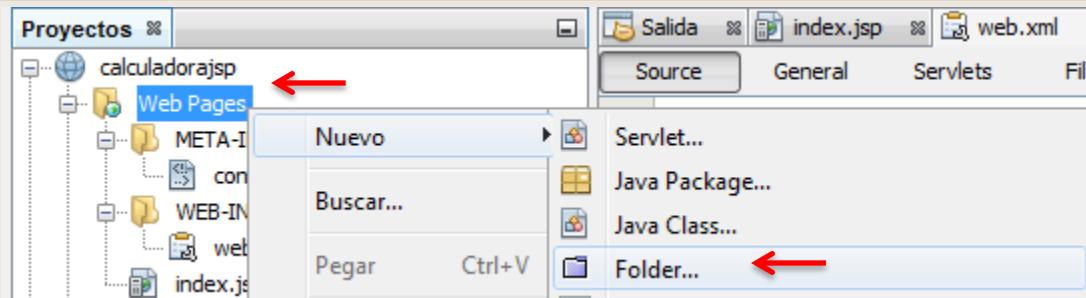
Método para atender peticiones GET

```
31
32     @Override
```

```
33     protected void doPost(HttpServletRequest request, HttpServletResponse response)
34         throws ServletException, IOException {
35         processRequest(request, response);
36     }
37 }
```

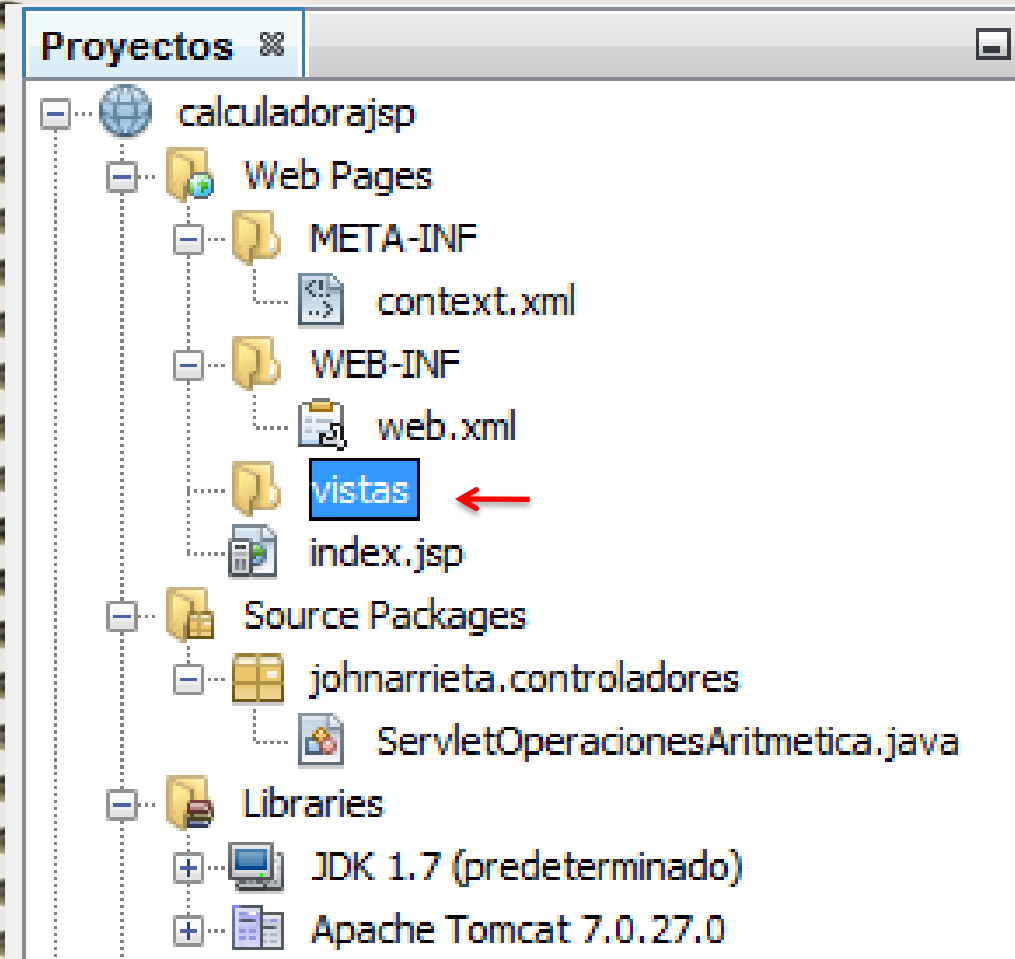
Método para atender peticiones POST

Crear una carpeta para las paginas JSP



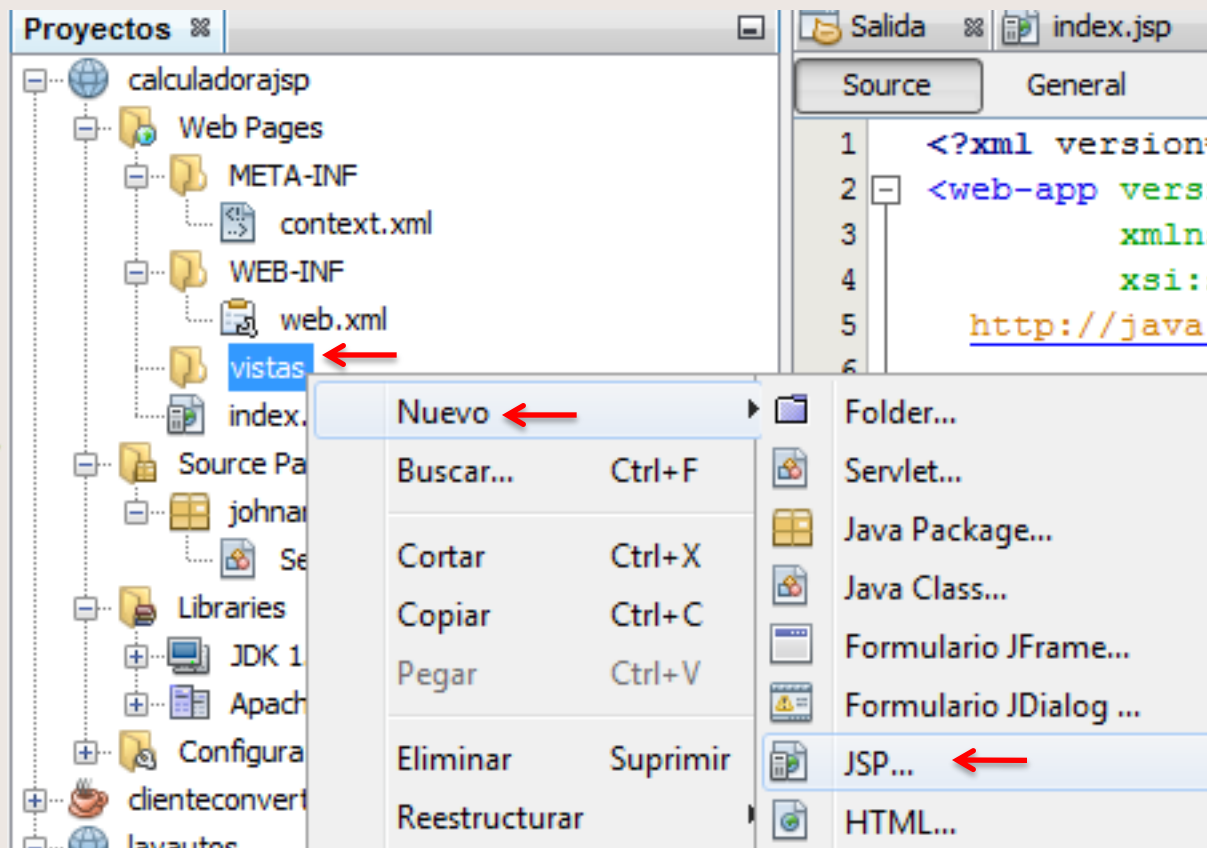
John Carlos Arrieta Arrieta

La nueva carpeta dentro de la estructura del proyecto



John Carlos Arrieta Arrieta

Crear una pagina JSP dentro del nueva capeta



John Carlos Arrieta Arrieta

Crear una Pagina JSP dentro de la carpeta Vista

Nuevo JSP

Pasos

1. Escoja el tipo de archivo
2. **Nombre y ubicación**

Nombre y ubicación

File Name:

Proyecto:

Ubicación:

Carpeta:

Examinar...

Archivo creado:

Options:

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

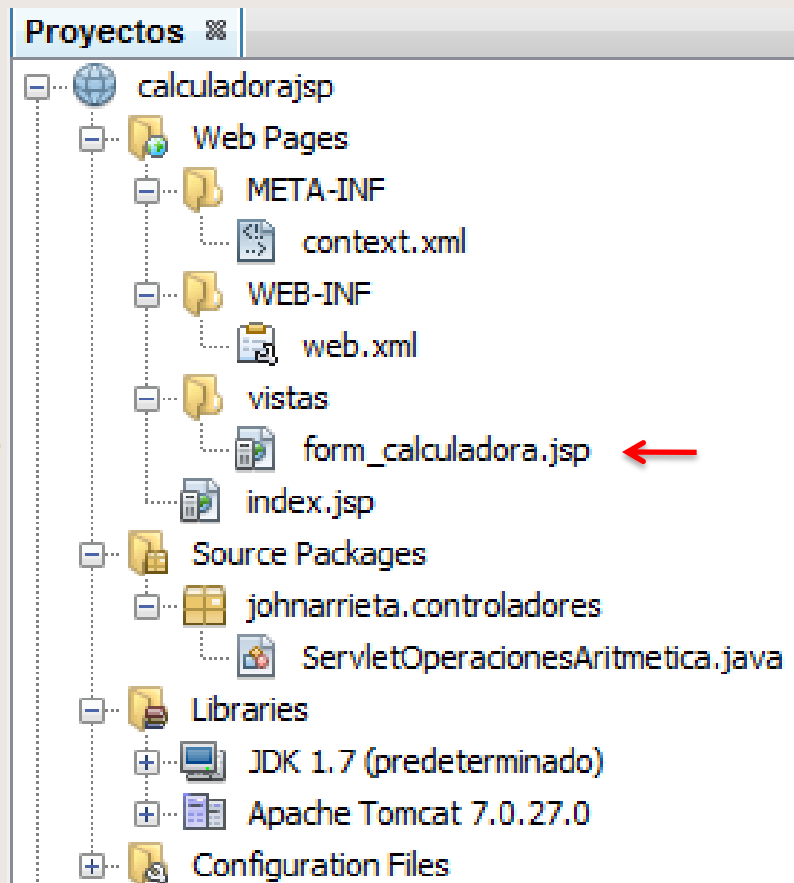
☐ JSP Document (XML Syntax)

Description:

< Atrás Siguiente > Terminar Cancelar Ayuda

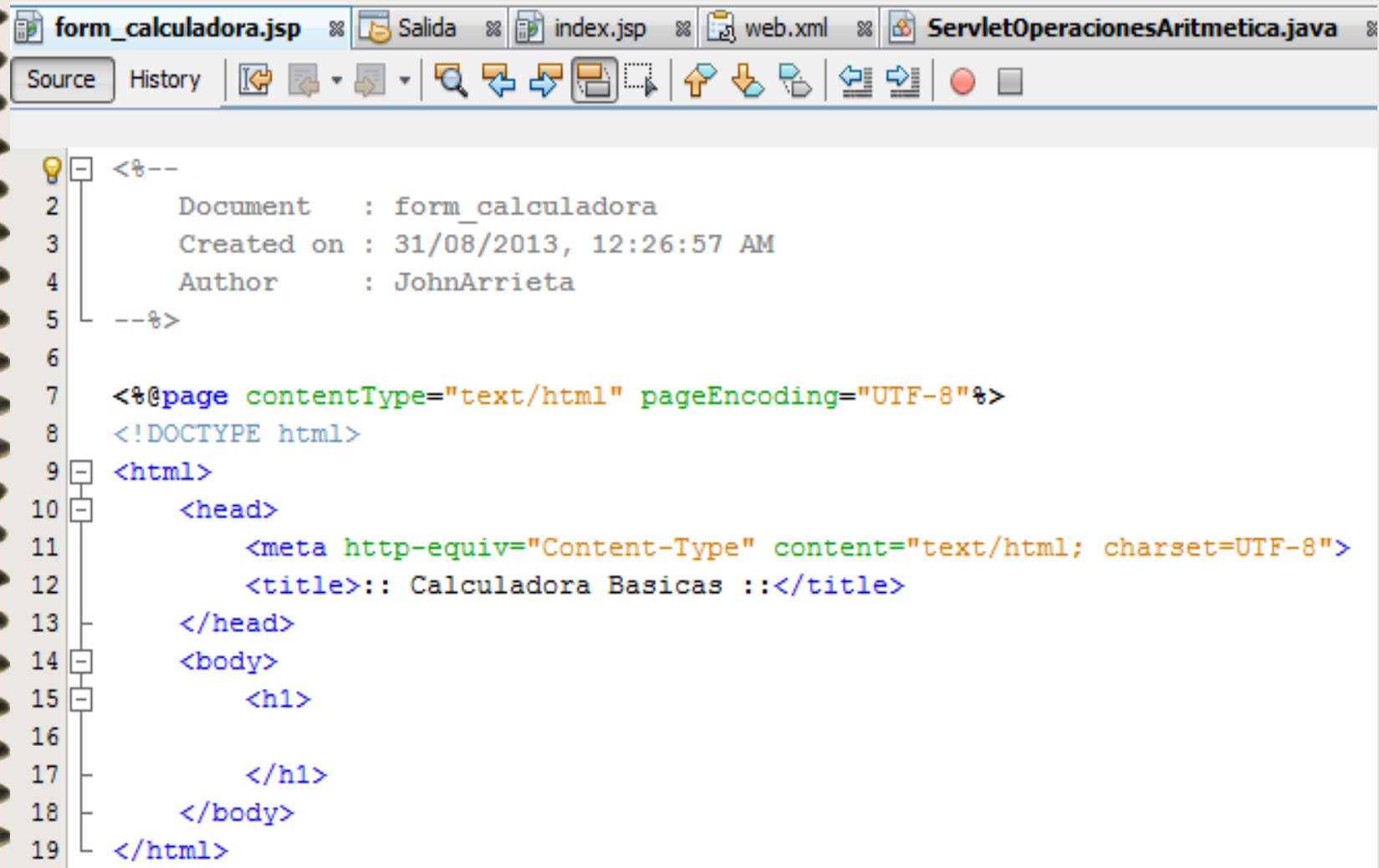
John Carlos Arrieta Arrieta

La Nueva Pagina JSP dentro de la estructura del proyecto



John Carlos Arrieta Arrieta

El contenido por defecto de la nueva Pagina JSP

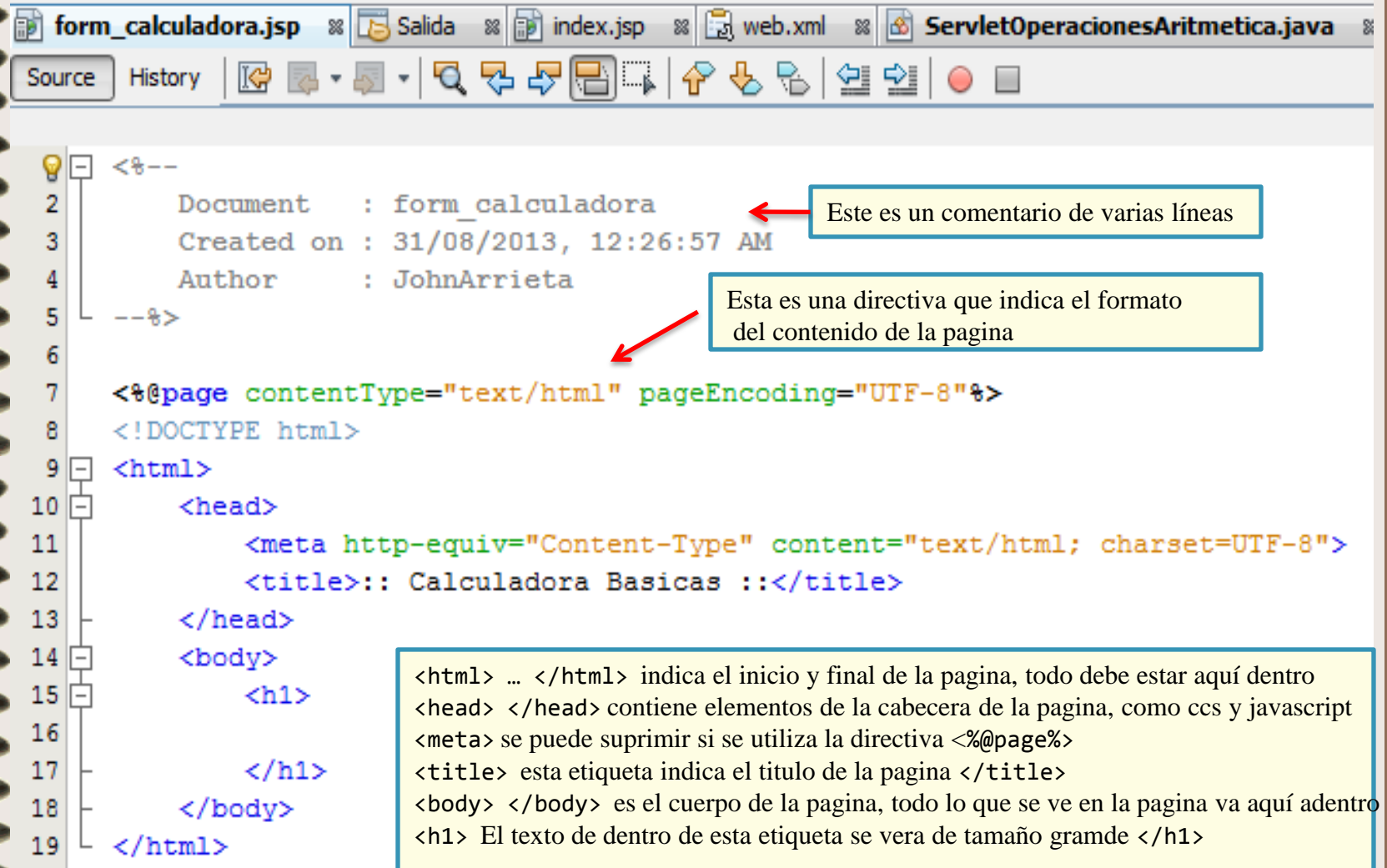


The screenshot shows an IDE window with several tabs: `form_calculadora.jsp`, `Salida`, `index.jsp`, `web.xml`, and `ServletOperacionesAritmetica.java`. The `form_calculadora.jsp` tab is active, showing the source code of a new JSP page. The code is as follows:

```
1 <%--
2     Document      : form_calculadora
3     Created on    : 31/08/2013, 12:26:57 AM
4     Author       : JohnArrieta
5 --%>
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>:: Calculadora Basicas ::</title>
13 </head>
14 <body>
15     <h1>
16
17     </h1>
18 </body>
19 </html>
```

John Carlos Arrieta Arrieta

El contenido por defecto de la nueva Pagina JSP



```
1  <%--
2      Document      : form_calculadora
3      Created on    : 31/08/2013, 12:26:57 AM
4      Author       : JohnArrieta
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>:: Calculadora Basicas ::</title>
13     </head>
14     <body>
15         <h1>
16
17         </h1>
18     </body>
19 </html>
```

Este es un comentario de varias líneas

Esta es una directiva que indica el formato del contenido de la pagina

<html> ... </html> indica el inicio y final de la pagina, todo debe estar aquí dentro
<head> </head> contiene elementos de la cabecera de la pagina, como ccs y javascript
<meta> se puede suprimir si se utiliza la directiva <%@page%>
<title> esta etiqueta indica el titulo de la pagina </title>
<body> </body> es el cuerpo de la pagina, todo lo que se ve en la pagina va aquí adentro
<h1> El texto de dentro de esta etiqueta se vera de tamaño grande </h1>

John Carlos Arrieta Arrieta

Utilizar la Paleta de elementos HTML para agregar un Formulario

The image shows a web editor interface with a file named `form_calculadora.jsp`. The editor displays the following HTML code:

```
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10   <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title:: Calculadora Basicas ::</title>
13   </head>
14   <body>
15     <h1>
16       <hr style="size:50%"/>
17     <hr/>
18   </h1>
19 </body>
20 </html>
```

The **Paleta** (palette) on the right side of the editor shows the following categories and elements:

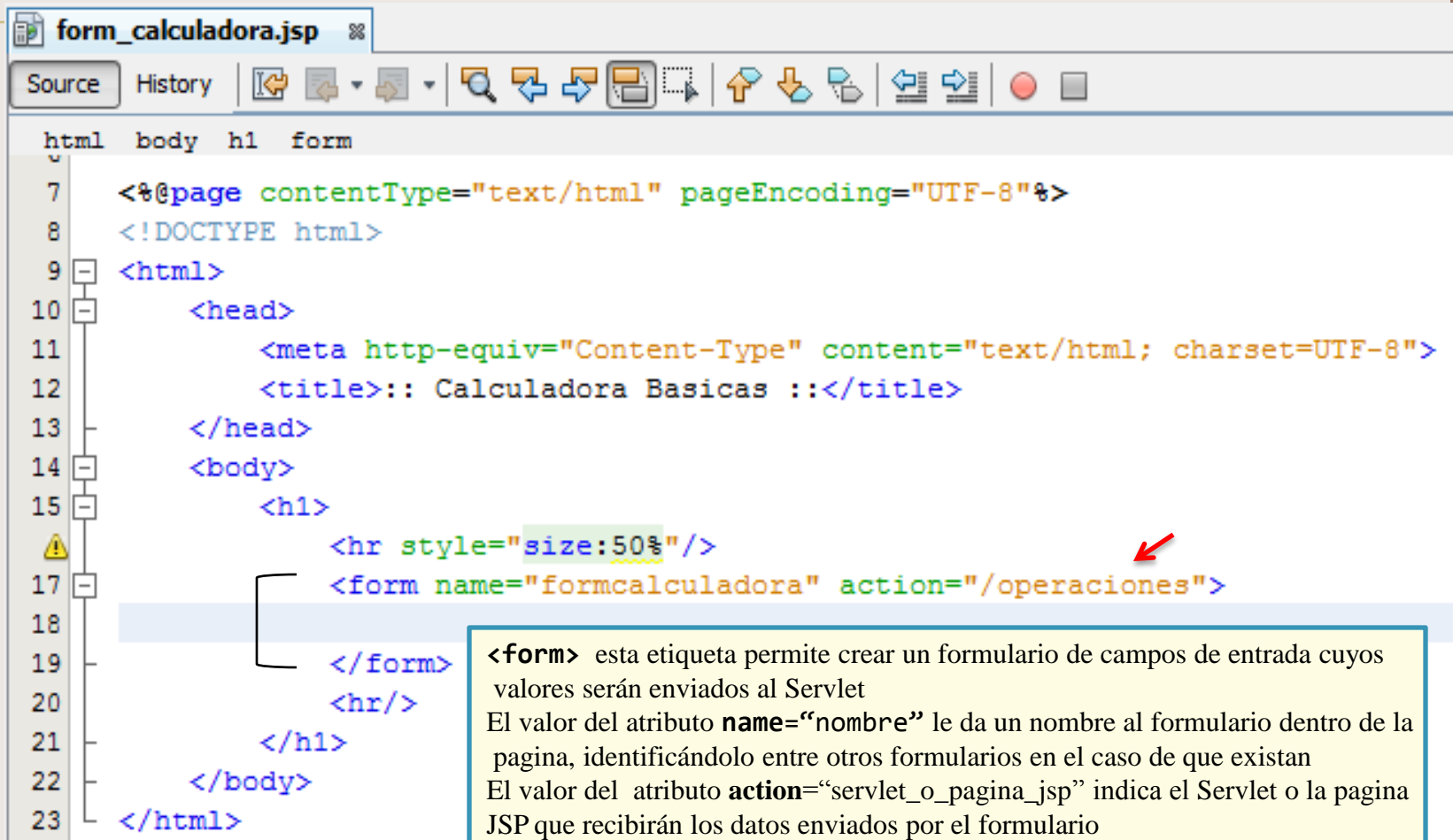
- HTML**
 - Tabla
 - Lista ordenada
 - Lista sin ordenar
 - Imagen
 - Enlace
 - Meta data
- Formularios HTML**
 - Formulario** (selected)
 - Entrada de texto
 - Entrada multilinea
 - desplegable
 - e activación
 - e opción
 - ar archivo

The **Insertar Formulario** (Insert Form) dialog box is open, showing the following configuration:

- Acción:** `/operaciones`
- Método:** ☒ GET ☐ POST
- Codificación:** ☒ application/x-www-form-urlencoded ☐ multipart/form-data
- Nombre:** `formcalculadora`

The **Aceptar** (Accept) button is highlighted.

La pagina JSP con el formulario



```
html body h1 form
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10 <head>
11   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12   <title:: Calculadora Basicas ::</title>
13 </head>
14 <body>
15   <h1>
16     <hr style="size:50%"/>
17     <form name="formcalculadora" action="/operaciones">
18
19   </form>
20   <hr/>
21 </h1>
22 </body>
23 </html>
```

<form> esta etiqueta permite crear un formulario de campos de entrada cuyos valores serán enviados al Servlet
El valor del atributo **name**="nombre" le da un nombre al formulario dentro de la pagina, identificándolo entre otros formularios en el caso de que existan
El valor del atributo **action**="servlet_o_pagina_jsp" indica el Servlet o la pagina JSP que recibirán los datos enviados por el formulario
El valor del atributo **method**="metodo_de_envio" indica si el envío de los datos se hace realiza usando el formato **GET** o el formato **POST**

html body h1

9 <html>

10 <head>

11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

12 <title>:: Calculadora Basicas ::</title>

13 </head>

14 <body>

15 <h1>

16 <hr style="size:50%;" />

17 <form name="formcalculadora" action="/operaciones">

18 <center>

19 <table border="0" >

20 <tr>

21 <th>Numero:</th>

22 <th><input type="text" name="numero1" value="" /></th>

23 </tr>

24 <tr>

25 <th>Numero:</th>

26 <th><input type="text" name="numero2" value="" /></th>

27 </tr>

28 <tr>

29 <td colspan="2">Resultado: </td>

30 </tr>

31 <tr>

32 <td><input type="submit" value="Calcular" /></td>

33 <td><input type="reset" value="Limpiar" /></td>

34 </tr>

35 </table>

36 </center>

37 </form>

38 <hr/>

39 </h1>

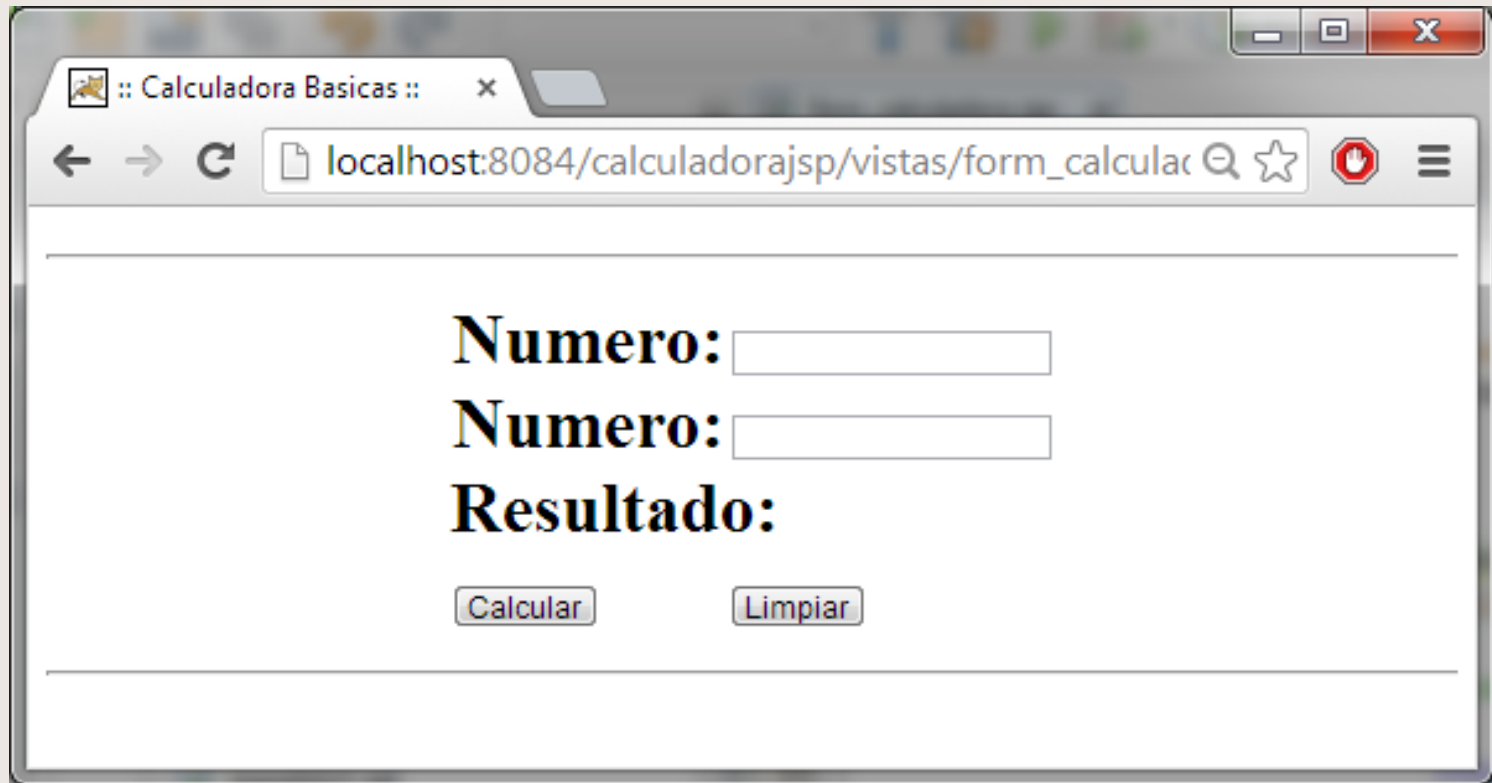
40 </body>

41 </html>

<table> </table> se utiliza para construir una tabla con filas <tr> </tr> y columnas <td> </td>

<input> se utiliza para colocar un campo de entrada de datos en el Formulario, los tipos de campos son **text** (campo de texto), **button** (botón), **radio** (botón circular de selección), **checkbox** (caja de Selección), **file** (subida de archivo), **hidden** (oculto o no visible), ... El atributo **name** es necesario para que el servlet o pagina JSP puedan capturar o recuperar el valor del campo .

Vista de la pagina JSP con formulario



The screenshot shows a web browser window with the title "Calculadora Basicas ::". The address bar displays "localhost:8084/calculadorajsp/vistas/form_calculac". The main content area contains the following text and form elements:

Numero:

Numero:

Resultado:

Así se ve el formulario mostrado por la pagina JSP `form_calculadora.jsp` explicado anteriormente. Los datos ingresados en los campos `numero1` y `numero2`, se envían al Servlet gracias a que están dentro de la etiqueta `<form>` y al menos uno de los campos es de tipo `<input type="submit">`, el cual inicia el envío de los datos al Servlet

John Carlos Arrieta Arrieta

```
<body>
  <h1>
    <hr style="size:50%;" />
    <form name="formcalculadora" action="/calculadorajsp/operaciones">
      <center>
        <table border="0" >
          <tr>
            <th>Numero:</th>
            <th colspan="3"><input type="text" name="numero1" value="" /></th>
          </tr>
          <tr>
            <th>Numero:</th>
            <th colspan="3"><input type="text" name="numero2" value="" /></th>
          </tr>
          <tr>
            <td colspan="2">Resultado: <%= (request.getAttribute("resultado") != null) ?
              request.getAttribute("resultado") :
              request.getAttribute("mensaje") %> </td>
          </tr>
          <tr>
            <td><input name="operacion" type="submit" value="+" /></td>
            <td><input name="operacion" type="submit" value="-" /></td>
            <td><input name="operacion" type="submit" value="*" /></td>
            <td><input name="operacion" type="submit" value="/" /></td>
            <td><input type="reset" value="Limpiar" /></td>
          </tr>
        </table>
      </center>
    </form>
  </h1>
</body>
```

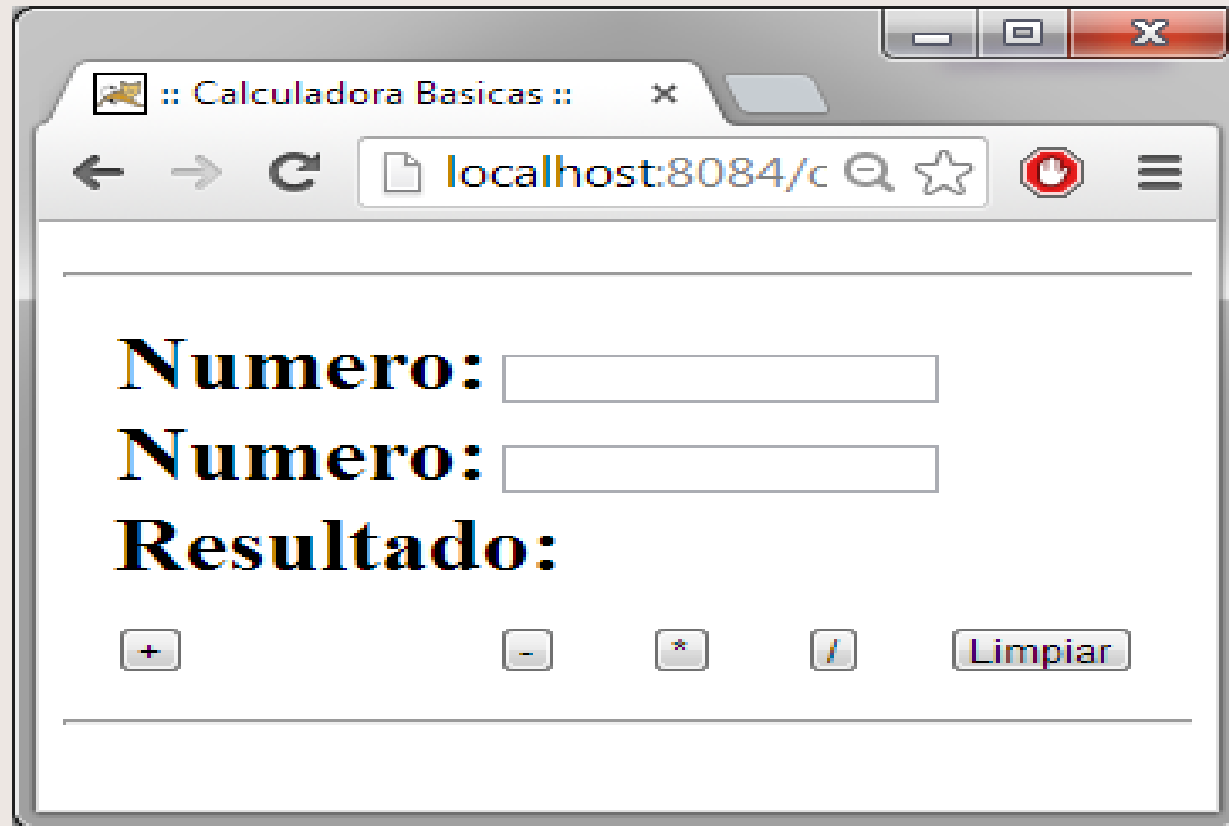
La variable **request** esta implícita dentro de toda pagina JSP, esta variable es en realidad un objeto de tipo **HttpServletRequest**, clase que contiene los métodos necesarios para recuperar **atributos** y **parámetros** de una petición, ya sea que esta provenga de una pagina HTML o una pagina JSP, con o sin formulario. En este caso recuperamos el atributo resultado y el atributo mensaje, si resultado es diferente de **null**, entonces su valor es concatenado o pegado al texto Resultado. De lo contrario, si resultado es igual a **null** se le asigna el valor del atributo mensaje.

John Carlos Arrieta Arrieta

```
<body>
  <h1>
    <hr style="size:50%;" />
    <form name="formcalculadora" action="/calculadorajsp/operaciones">
      <center>
        <table border="0" >
          <tr>
            <th>Numero:</th>
            <th colspan="3"><input type="text" name="numero1" value="" /></th>
          </tr>
          <tr>
            <th>Numero:</th>
            <th colspan="3"><input type="text" name="numero2" value="" /></th>
          </tr>
          <tr>
            <td colspan="2">Resultado: <%= (request.getAttribute("resultado") != null) ?
              request.getAttribute("resultado") :
              request.getAttribute("mensaje") %> </td>
          </tr>
          <tr >
            <td><input name="operacion" type="submit" value="+" /></td>
            <td><input name="operacion" type="submit" value="-" /></td>
            <td><input name="operacion" type="submit" value="*" /></td>
            <td><input name="operacion" type="submit" value="/" /></td>
            <td><input type="reset" value="Limpiar" /></td>
          </tr>
        </table>
      </center>
    </form>
  </h1>
</body>
```

Hemos modificado la pagina JSP `form_calculadora.jsp` para agregarle 4 campos. Estos campos son de tipo `<input type="submit">` y comparten el mismo nombre. Pero diferente valor, esto permite que al ser pulsado cualquiera de ellos, el formulario Enviara su nombre y su valor, en este caso enviara **operación=+** si el usuario pulsa El botón +, pero también enviara los campos `numero1` y `numero2` con sus respectivos Valores, por ejemplo: **numero1=7&numero2=4&operacion=+**. El campo de tipo **reset** se utiliza para volver el formulario a sus valores iniciales.

Vista de la pagina JSP modificada



The screenshot shows a web browser window with the title bar 'Calculadora Basicas ::'. The address bar displays 'localhost:8084/c'. The page content includes:

- Two input fields, each preceded by the label 'Numero:'.
- A label 'Resultado:' followed by a large empty space for the result.
- A row of buttons: a '+' button, a '-' button, a '*' button, a '/' button, and a 'Limpiar' button.

Al ejecutar la aplicación así se vera el la pagina JSP **form_calculadora.jsp** con las modificaciones realizadas.

```

form_calculadora.jsp x ServletOperacionesAritmeticas.java x
Source History
2  * El Servlet Controlador, pues recupera los datos de la vista determina que
3  * operacion se debe realizar con dichos datos y redirecciona la respuesta a la
4  * vista, en este caso la misma vista que realizo la peticion al Servlet
5  */
6  package johnarrieta.controladores;
7
8  import java.io.IOException;
9  import java.io.PrintWriter;
10 import javax.servlet.ServletException;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14
15 /**
16  * @author JohnArrieta
17  */
18 public class ServletOperacionesAritmeticas extends HttpServlet {
19
20     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
21         throws ServletException, IOException {...}
22
23
24     public float operacion(float numero1, String operador, float numero2) {...}
25
26
27     @Override
28     protected void doGet(HttpServletRequest request, HttpServletResponse response)
29         throws ServletException, IOException {...}
30
31
32     @Override
33     protected void doPost(HttpServletRequest request, HttpServletResponse response)
34         throws ServletException, IOException {...}
35
36
37     @Override
38     public String getServletInfo() {...} // </editor-fold>
39
40 }

```

Este es el código del **ServletOperacionesAritmetica** con las clases necesarias importadas, los métodos **processRequest**, **doGet** y **doPost**.

Si la petición es hecha mediante GET, entonces se invoca automáticamente el método **doGet**, la información incluida en la petición, es recuperada automáticamente por la variable **request**, mientras que la variable **response** posee las operaciones necesarias para enviar información de Respuesta, si por el contrario la petición se hace con POST Entonces se ejecuta automáticamente el método **doPost**.

John Carlos Arrieta Arrieta

Detalles del

ServletOperacionesAritmeticas.java

```
1  /*
2   * El Servlet Controlador, pues recupera los datos de la vista determina que
3   * operacion se debe realizar con dichos datos y redirecciona la respuesta a la
4   * vista, en este caso la misma vista que realizo la peticion al Servlet
5   */
6  package johnarrieta.controladores;
7
8  import java.io.IOException;
9  import java.io.PrintWriter;
10 import javax.servlet.ServletException;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
```

Línea 6: se indica que la clase ServletOperacionesAritmetica.java será almacenada en el paquete controladores
Líneas 8 a 13: se importan las clases que necesita el **Servlet** para poder funcionar adecuadamente.
La clase **IOException**: se usa para una excepción de entrada y de salida
La clase **PrintWriter**: Se utiliza para enviar datos como respuesta de la petición realizada por el usuario.
La clase **HttpServlet**: Es la clase padre o **super** clase de todos los **Servlet**, contiene métodos **doGet** y **doPost**
La clase **HttpServletRequest**: se utiliza para capturar información incluida en la petición
La clase **HttpServletResponse**: Se utiliza junto con **PrintWriter** para enviar datos como respuesta a la peticion

John Carlos Arrieta Arrieta

Detalles del

ServletOperacionesAritmeticas.java

```
20 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
21     throws ServletException, IOException {
22     response.setContentType("text/html;charset=UTF-8");
23     PrintWriter out = response.getWriter();
24     try {
25         /* TODO output your page here. You may use following
26         String numero1 = request.getParameter("numero1");
27         String numero2 = request.getParameter("numero2");
28         String operando = request.getParameter("operacion");
29         float num1 = Float.parseFloat(numero1);
30         float num2 = Float.parseFloat(numero2);
31         float resultado = this.operacion(num1, operando, num2);
32         request.setAttribute("resultado", String.valueOf(resultado));
33         getServletContext().getRequestDispatcher("/vistas/form_calculadora.jsp").forward(request, respon
34
35     } catch (Exception error) {
36         request.setAttribute("error", error.getMessage());
37         getServletContext().getRequestDispatcher("/vistas/form_calculadora.jsp").forward(request, respon
38     }
39     finally {
40
41     }
42 }
```

Este método es invocado por **doGet** o por **doPost** cada vez que se realiza una petición ya sea Usando **GET** o **POST**. Lo primero que realiza este método en su cuerpo es indicar invocar el método **setContentType** de la clase **HttpServletResponse** con el objetivo de indicar que la salida Sera en formato **texto/html/codificada** en **UTF-8**

Luego invoca el método **getParameter** para recuperar los campos o parámetros enviados en la Petición, dos de ellos son convertidos a decimal, luego son pasados como argumentos al invocar El método **operacion** declarado en esta misma clase, operación retorna un **resultado**, el cual es Colocado como atributo de la petición y por ultimo la petición es disparada hacia la pagina **form_calculadora.jsp** invocando el mentado forward de la clase **RequestDispatcher** Si llegase a ocurrir una excepción o error, este es tratado en el bloque **catch**

Detalles del ServletOperacionesAritmeticas.java

```
44 public float operacion(float numero1, String operador, float numero2) {  
45     if (operador.equals("+")) {  
46         return numero1 + numero2;  
47     } else if (operador.equals("-")) {  
48         return numero1 - numero2;  
49     } else if (operador.equals("*")) {  
50         return numero1 * numero2;  
51     } else if (operador.equals("/")) {  
52         return numero1 / numero2;  
53     } else {  
54         return 0;  
55     }  
56 }
```

Este método es invocado por el método `processRequest`, el cual a su vez es invocado por los métodos `doGet` y `doPost`, Que a su vez son invocados automáticamente cada vez que se realiza una operación usando GET o usando POST. Este método operación recibe 3 argumentos, 2 de tipo decimal y 1 de tipo cadena de texto. Internamente compara el valor del argumento llamado operador con los símbolos de las operaciones aritméticas +, -, * y /. Si encuentra una coincidencia entonces realiza la operación correspondiente retornando el resultado de dicha operación. Si no encuentra coincidencia alguna retorna 0.

Detalles del ServletOperacionesAritmeticas.java

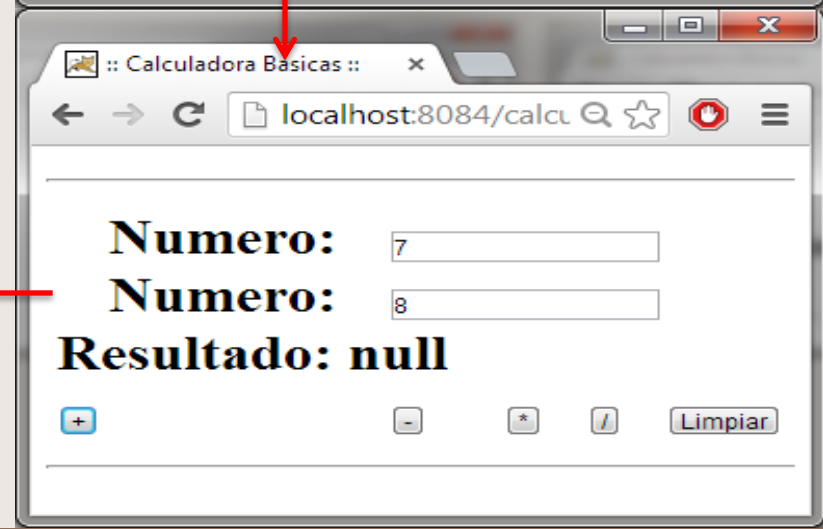
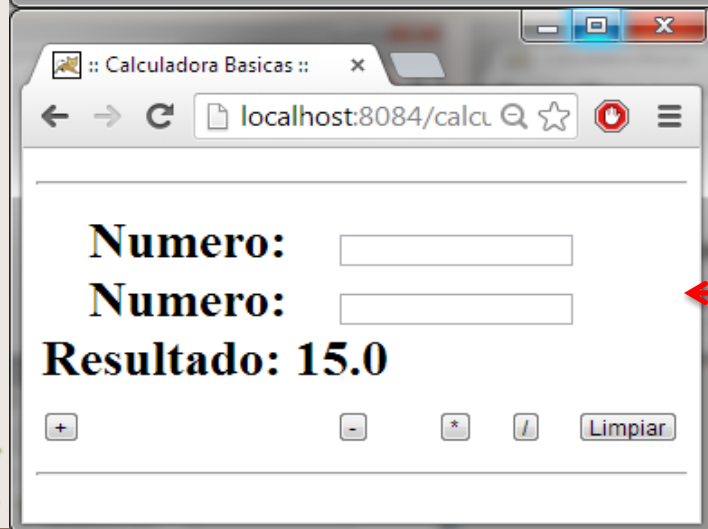
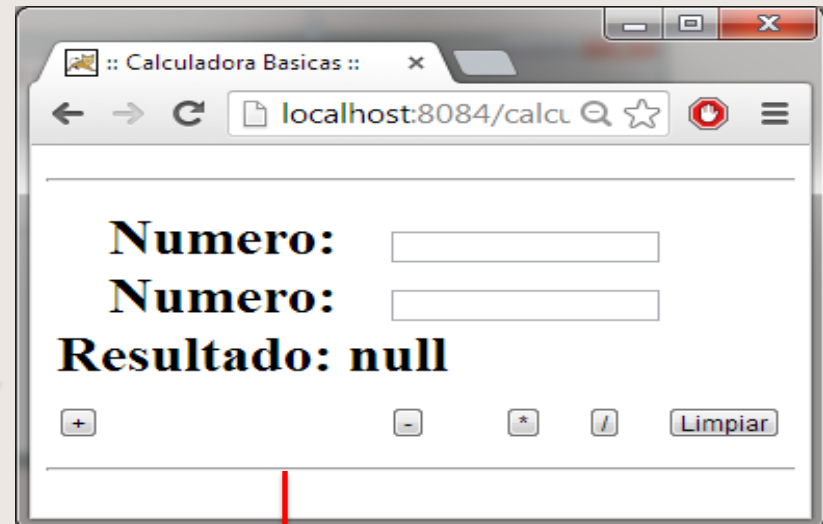
```
@Override
59 protected void doGet(HttpServletRequest request, HttpServletResponse response)
60     throws ServletException, IOException {
61     processRequest(request, response);
62 }

@Override
65 protected void doPost(HttpServletRequest request, HttpServletResponse response)
66     throws ServletException, IOException {
67     processRequest(request, response);
68 }

@Override
71 public String getServletInfo() {
72     return "Short description";
73 } // </editor-fold>
```

El método **doGet** se invoca y ejecuta automáticamente cada vez que se realiza una petición usando **GET**. Este a su vez carga y recibe como argumentos un objeto de la clase **HttpServletRequest** llamado **request** y otro de la clase **HttpServletResponse** llamado **response**. Este método invoca al método **processRequest**, el cual es el encargado de procesar la petición. El método **doPost** es similar a **doGet**, con la diferencia que este se invoca y ejecuta automáticamente cada vez que se realiza una petición usando **POST**.

Probando el funcionamiento del Proyecto



Actividades a desarrollar

EJERCICIOS Y PREGUNTAS A RESOLVER:

- Realizar un conversor de moneda
- Realizar un Traductor de días, meses y colores
- Realizar una tienda básica, de compra y venta de artículos sin bases de datos, solo utilizando estructura de datos y sesiones.
- Consultar, estudiar y practicar cuales otras clases y sus métodos se utilizan en aplicaciones JSP y Servlet.
- Atributos a nivel de aplicación, de sesión, de petición y de respuesta
- Como se pasan parámetros entre un Servlet y otro Servlet
- Como se pasan parámetros entre una JSP y otra JSP
- De que manera se puede importar una clase dentro de un JSP
- De que manera se pueden recuperar automáticamente todos los campos de un formulario dentro de un servlet o una JSP
- Como se puede manejar los errores de peticiones de forma automática, Por ejemplo personalizando una pagina especialmente para mostrar errores