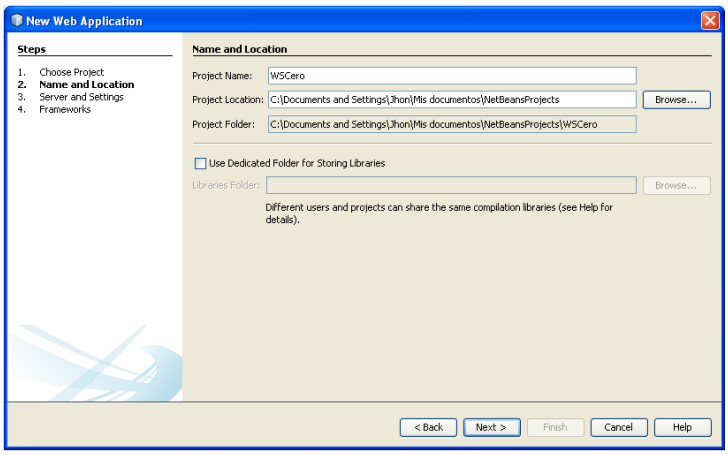
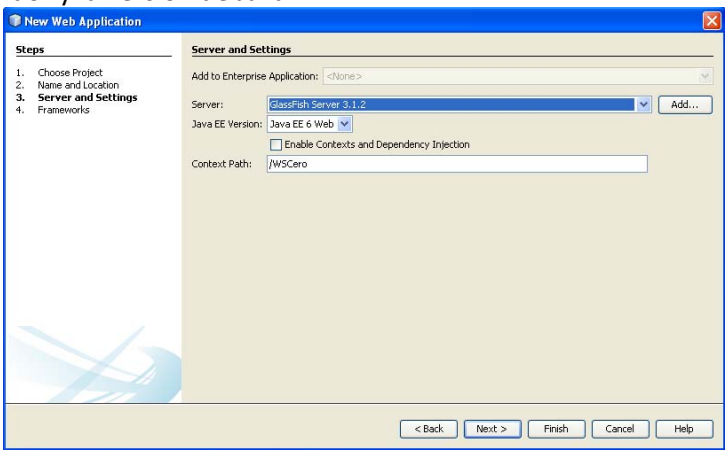
Construcción de un Servicio Web

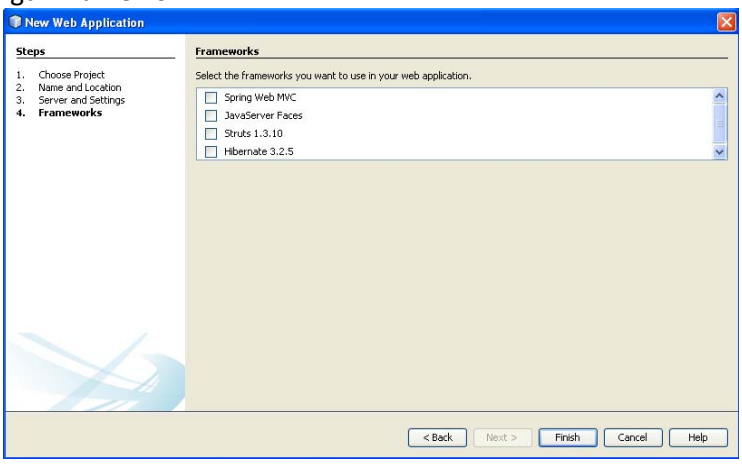
1. Crear un proyecto Java Web → Web Application
2. Adicione un nombre a la aplicación web



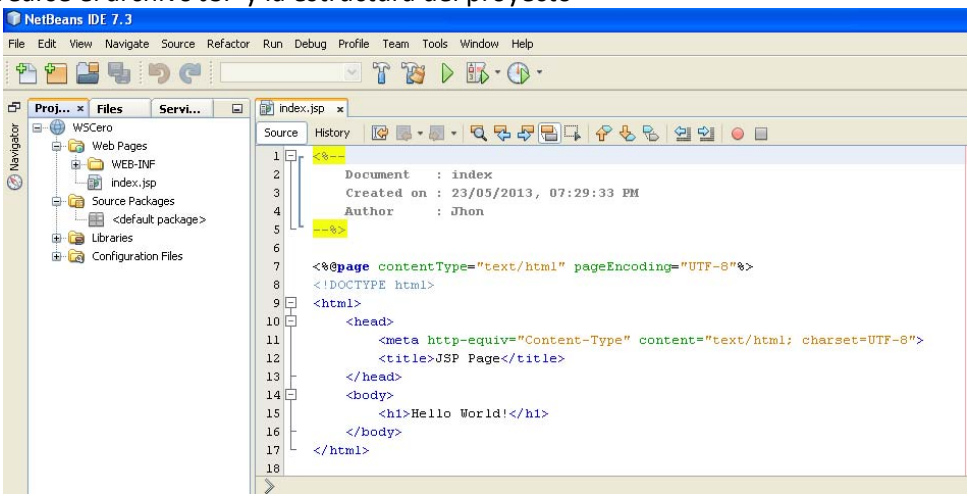
3. Seleccione el servidor y la versión de Java



4. No seleccione ningún framework

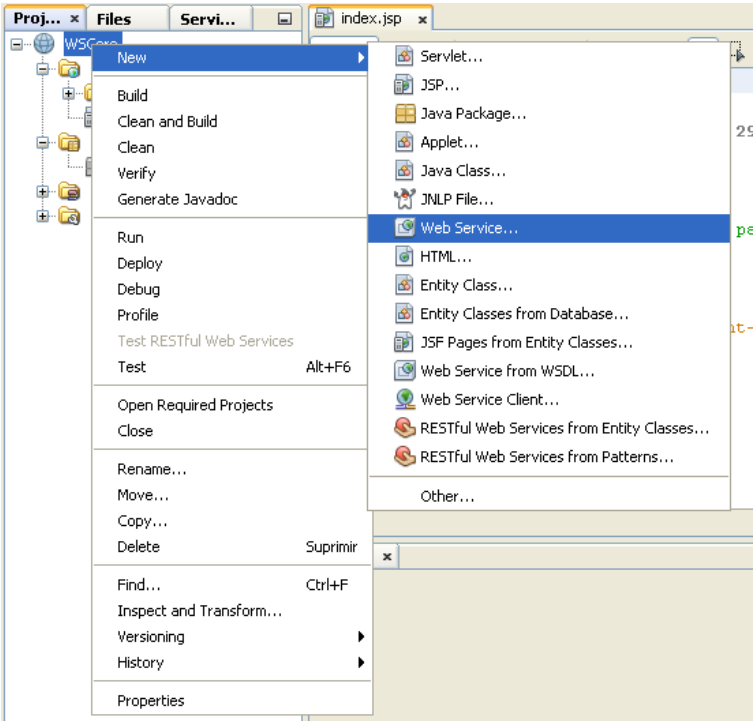


5. Debe crearse el archivo JSP y la estructura del proyecto



6. Se debe crear un paquete donde se va a almacenar el servicio web.

7. A continuación es crear el web services, dando click derecho en el proyecto creado y seleccionando New > Web Service



8. Se le coloca un nombre (CasadeCambio), y para el caso de ejemplo se selecciona el combo de stateless session bean, y se da click en “Finish”

Los beans de sesión se mantienen a lo largo de la sesión del cliente con la finalidad de dar representar flujo de trabajo, estado de aplicación, lógica y utilidades de empresa. No obstante, no es persistente,

puesto que los datos no se almacenan en un repositorio y por tanto, los datos de la sesión no sobrevivirían a una caída del servidor, por ejemplo. Un bean de sesión con estado (stateful) puede mantener datos entre las diferentes peticiones de un cliente, pero no ocurre así con un bean de sesión sin estado (stateless).

Se recomienda el uso de **stateful** cuando se cumpla una de las siguientes situaciones:

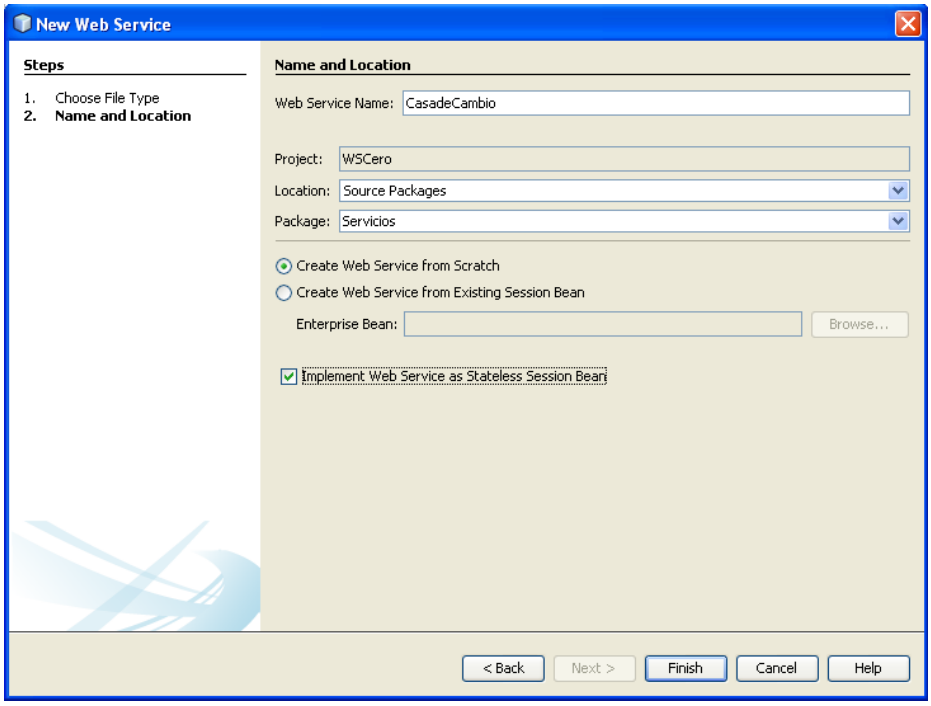
- El estado del bean representa la interacción entre el bean y un cliente específico.
- El bean necesita mantener información acerca del cliente a lo largo de varias invocaciones de métodos.
- El bean es mediador entre el cliente y los otros componentes de la aplicación, presentando una vista simplificada para el cliente.

Internamente, el bean maneja el control de flujo de muchos otros enterprise beans. Para incrementar el rendimiento, se podrían utilizar **stateless** session bean en alguna de las siguientes situaciones:

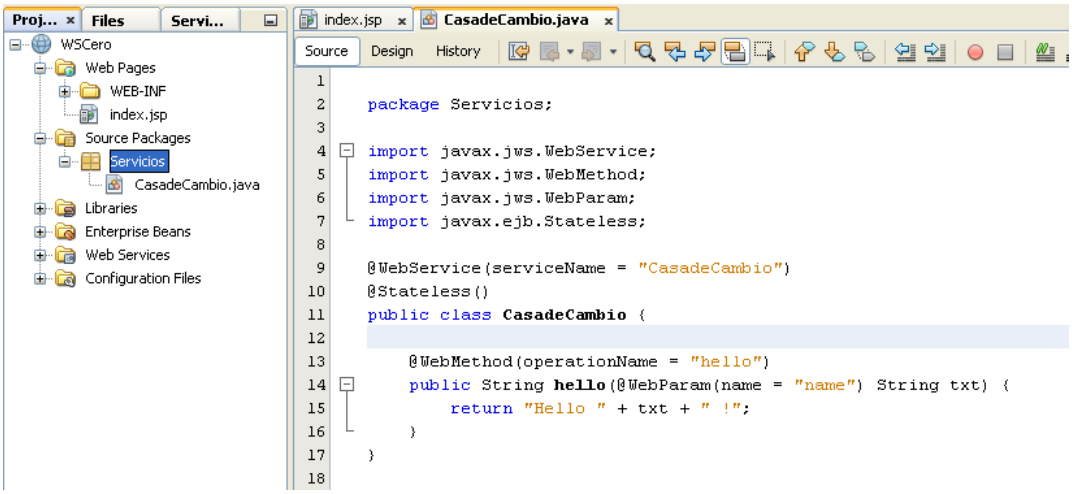
- El estado del bean no tiene datos para un cliente en específico.
- En un único método de invocación, el bean realiza una tarea genérica para todos los clientes. Por ejemplo, se podría utilizar un stateless session bean para mandar un email que confirme una compra online.
- El bean implementa un servicio web.

Los **singleton** session bean son apropiados en las siguientes circunstancias:

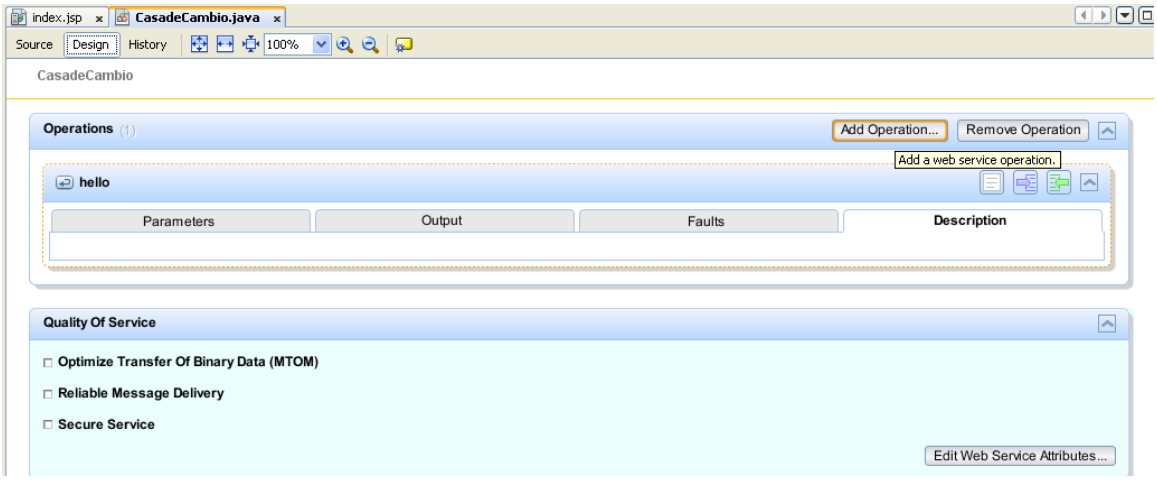
- El estado necesita ser compartido a través de la aplicación.
- Un único enterprise bean necesita ser accedido por múltiples hebras concurrentemente.



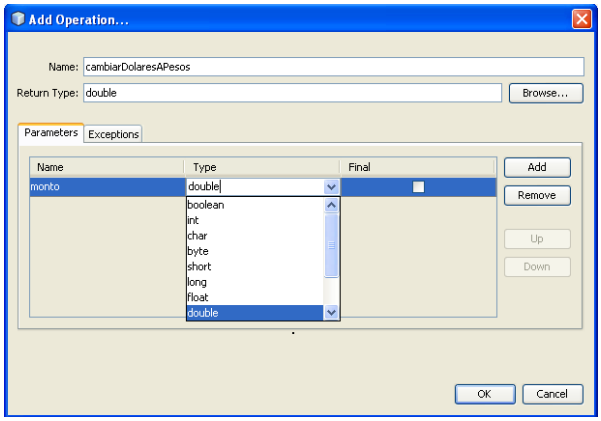
9. Al terminar la operación, adicional al JSP debe aparecer el archivo del Web Service



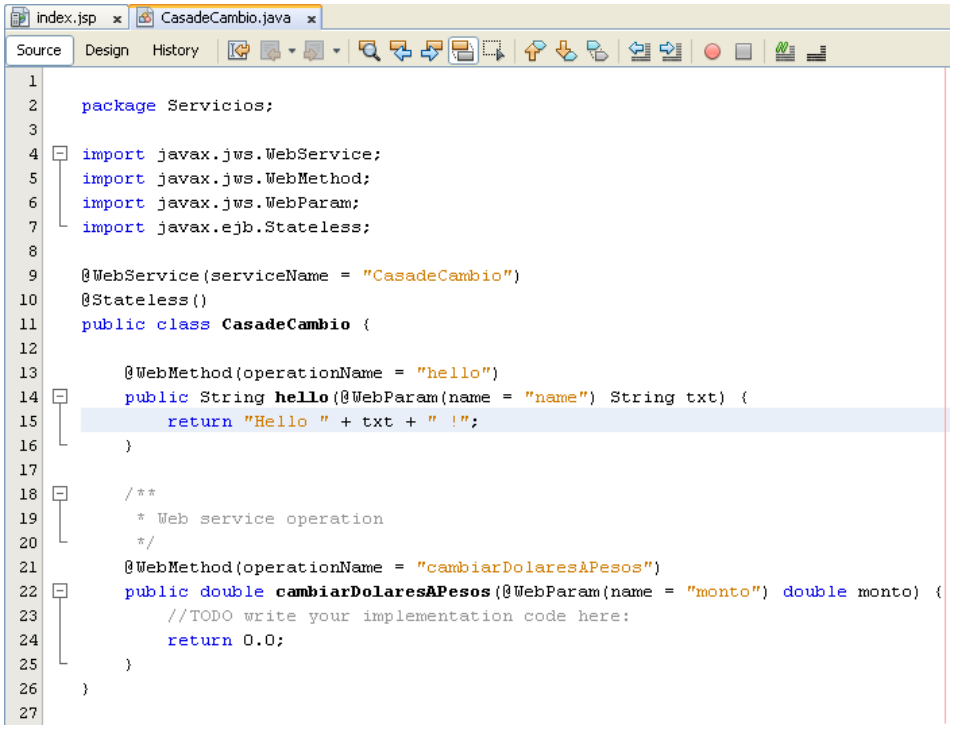
10. Ya creado el web service, se ingresa a la pestaña diseño (Design) y se da click en el “boton add operation”



11. Se selecciona el nombre de la operación (cambiarDolaresAPesos), se define el tipo de retorno de la operación (para este caso double), y se agrega una variable, en este caso monto, de tipo Double

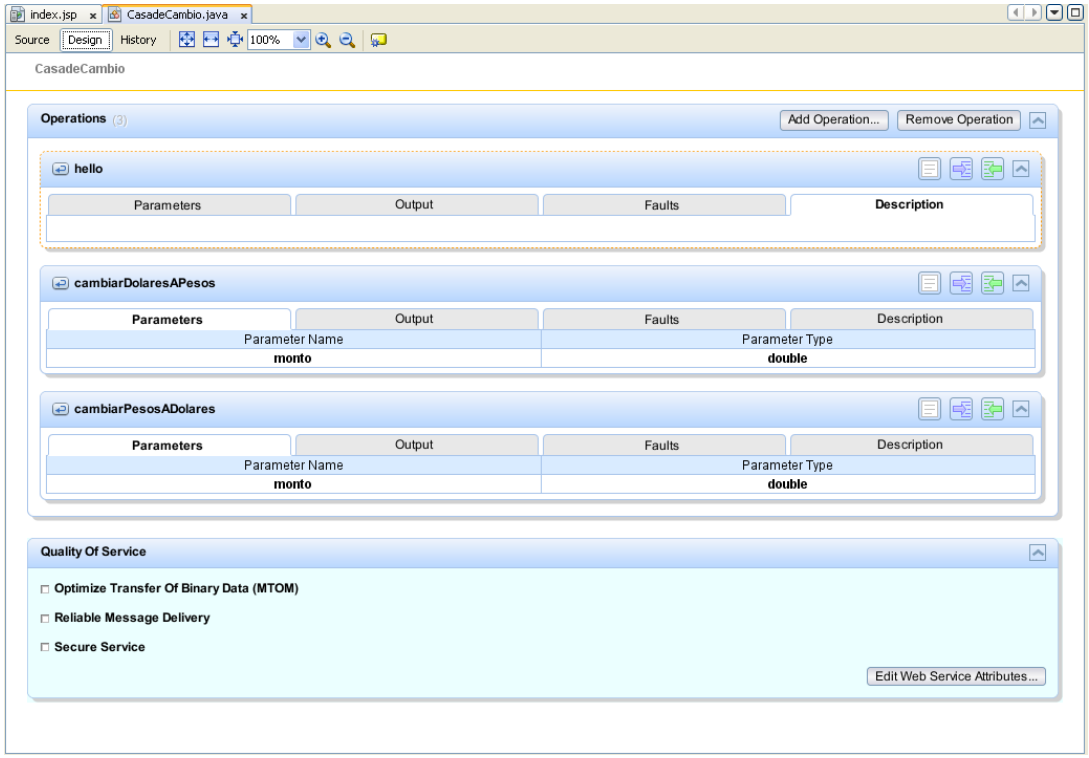


12. En la vista de Código (Source) se verifica que ya se haya agregado la operación

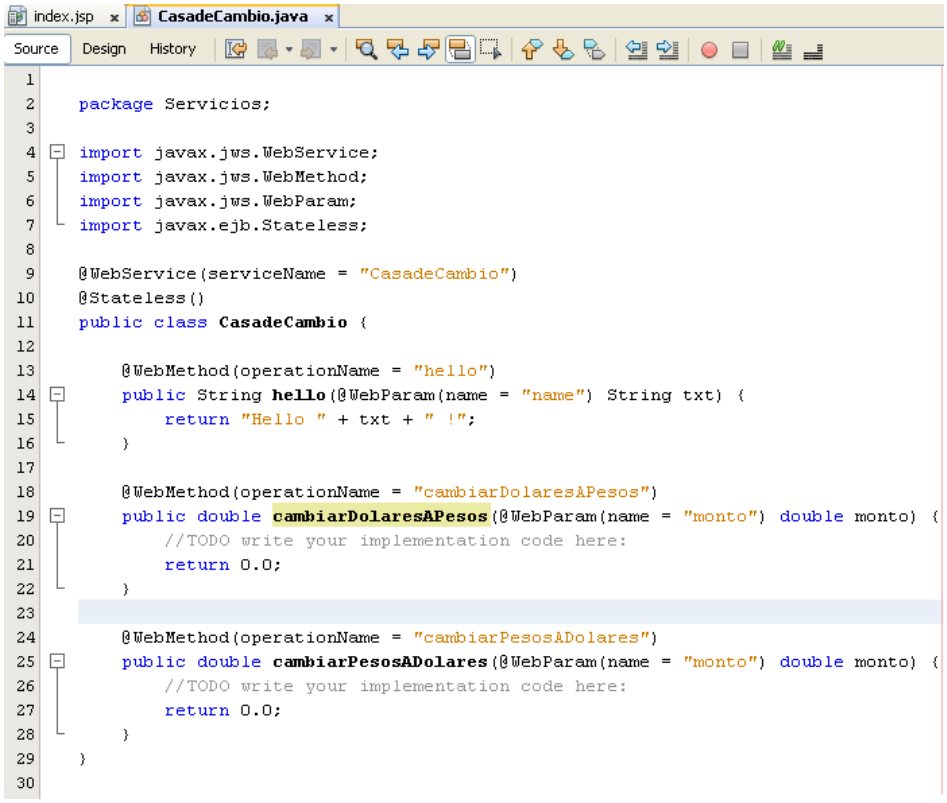


13. De igual forma se hace la adición de la operación de conversión de Pesos a Dólares

En vista de Diseño



En vista de Código



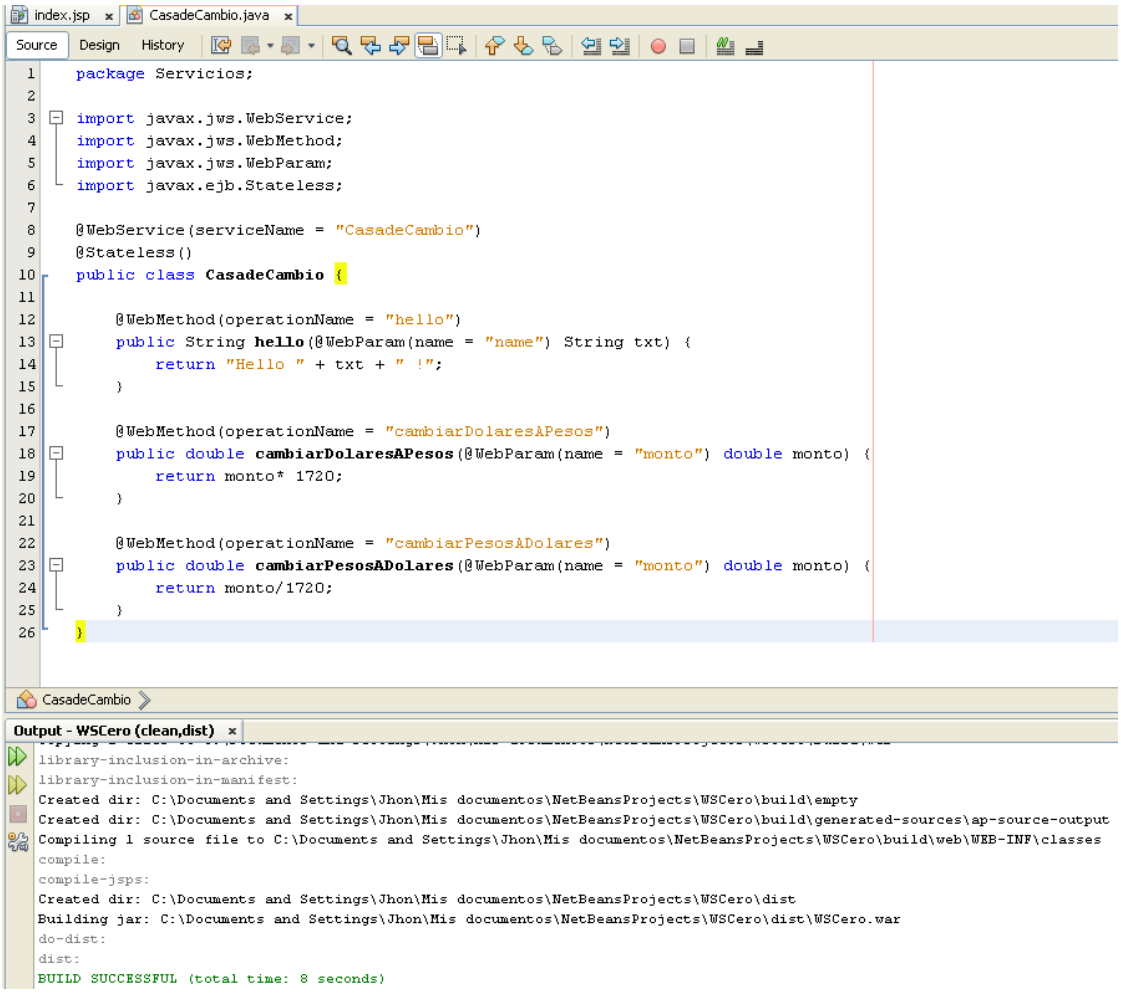

```
1
2   package Servicios;
3
4   import javax.jws.WebService;
5   import javax.jws.WebMethod;
6   import javax.jws.WebParam;
7   import javax.ejb.Stateless;
8
9   @WebService(serviceName = "CasadeCambio")
10  @Stateless()
11  public class CasadeCambio {
12
13      @WebMethod(operationName = "hello")
14      public String hello(@WebParam(name = "name") String txt) {
15          return "Hello " + txt + " !";
16      }
17
18      @WebMethod(operationName = "cambiarDolaresAPesos")
19      public double cambiarDolaresAPesos(@WebParam(name = "monto") double monto) {
20          //TODO write your implementation code here:
21          return 0.0;
22      }
23
24      @WebMethod(operationName = "cambiarPesosADolares")
25      public double cambiarPesosADolares(@WebParam(name = "monto") double monto) {
26          //TODO write your implementation code here:
27          return 0.0;
28      }
29  }
30
```

14. Para probar la operación de Dólares a Pesos, se debe ubicar la formula de conversión, para ello se cambia la sentencia retun, adicionándole "return monto * 2.6"

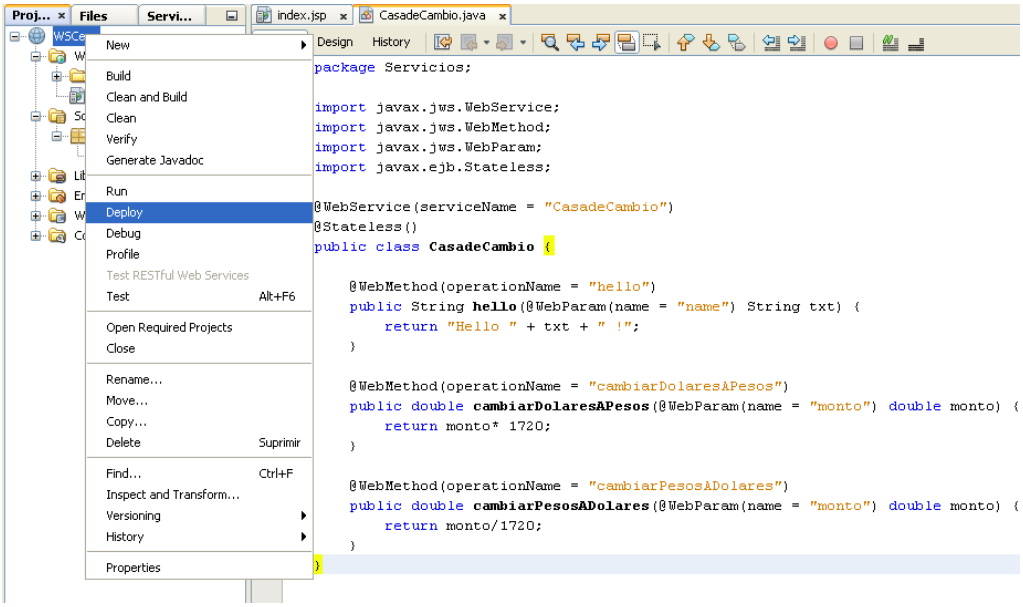
```
@WebMethod(operationName = "cambiarDolaresAPesos")
  public double cambiarDolaresAPesos(@WebParam(name = "monto") double monto) {
    return monto* 1720;
  }

  @WebMethod(operationName = "cambiarPesosADolares")
  public double cambiarPesosADolares(@WebParam(name = "monto") double monto) {
    return monto/1720;
}
```

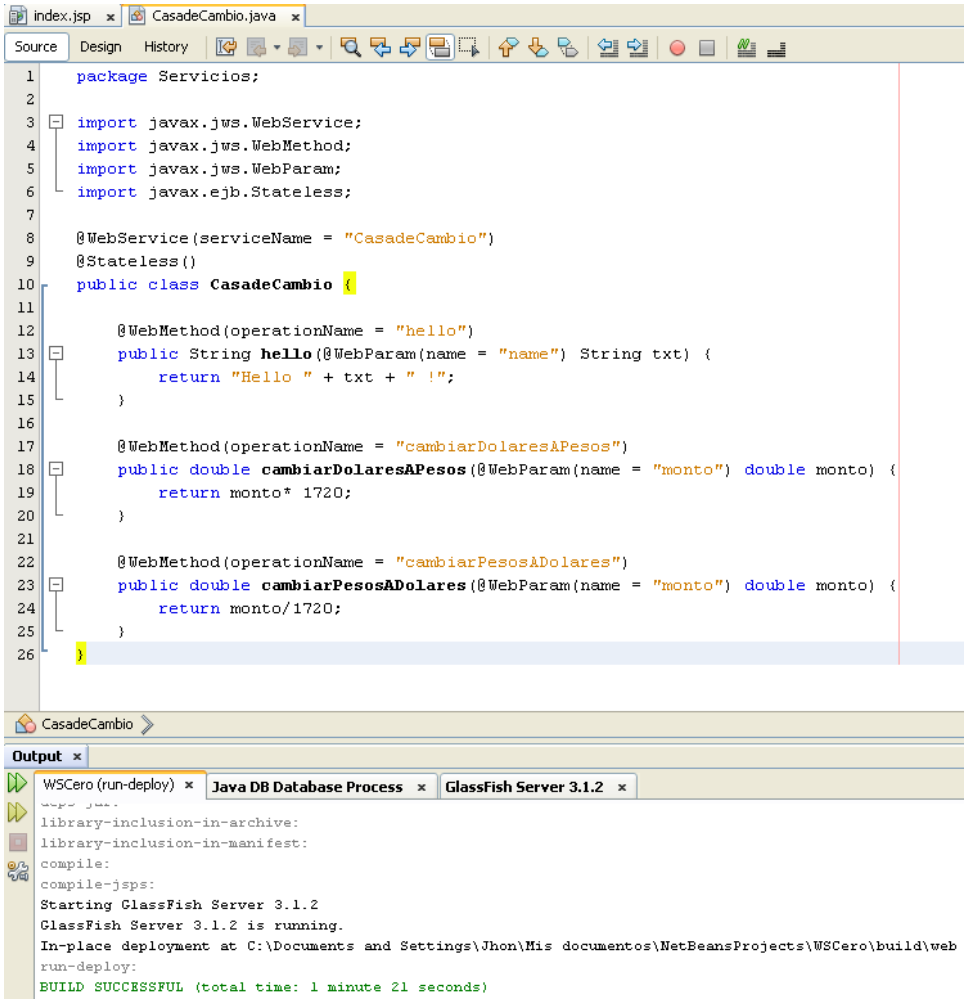
15. Con estas operaciones se completaría el Servicio Web. Para empezar su ejecución se debe hacer click derecho en el proyecto, luego se selecciona “Clean &build”



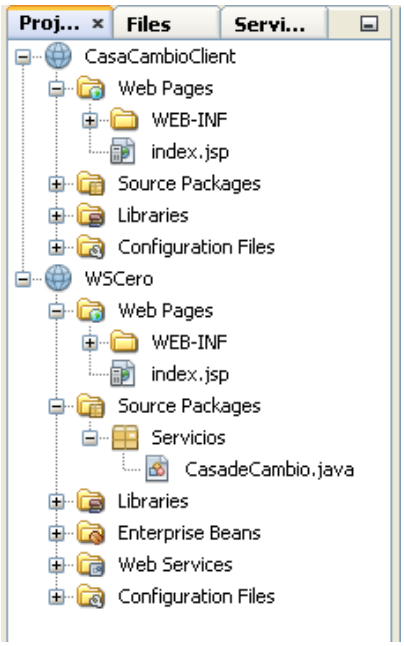
16. Una vez acabado de procesar, se da click derecho en el proyecto y selecciona “deploy”



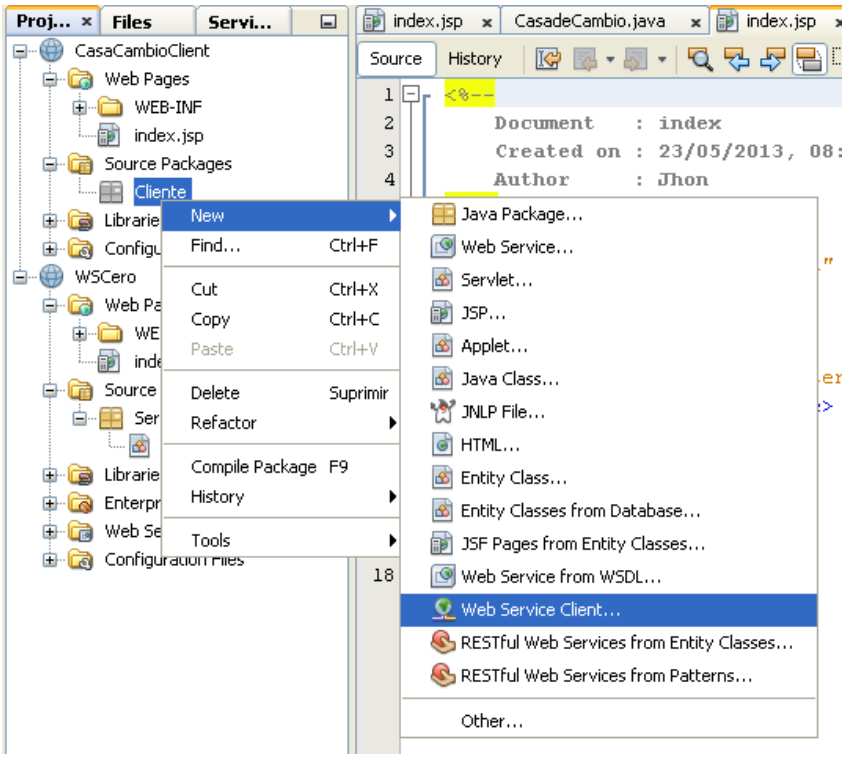
Cuando se hace Deploy, se inician el servidor de Bases de Datos y el Servidor GlassFish (o el que se tenga instalado)



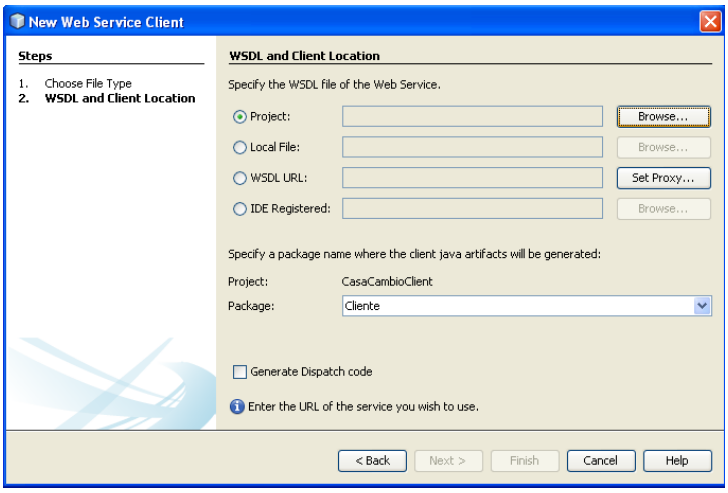
17. Una vez terminado el proceso ya se tiene el servidor listo (No se debe cerrar ningún servidor en ejecución). A continuación se necesita un cliente que consuma este servicio. Para ello se debe crear un nuevo Projec → Web Application, al cual se le coloca el nombre CasaCambioClientApp y se continúa su construcción de la misma forma que en los primeros pasos de la guía.

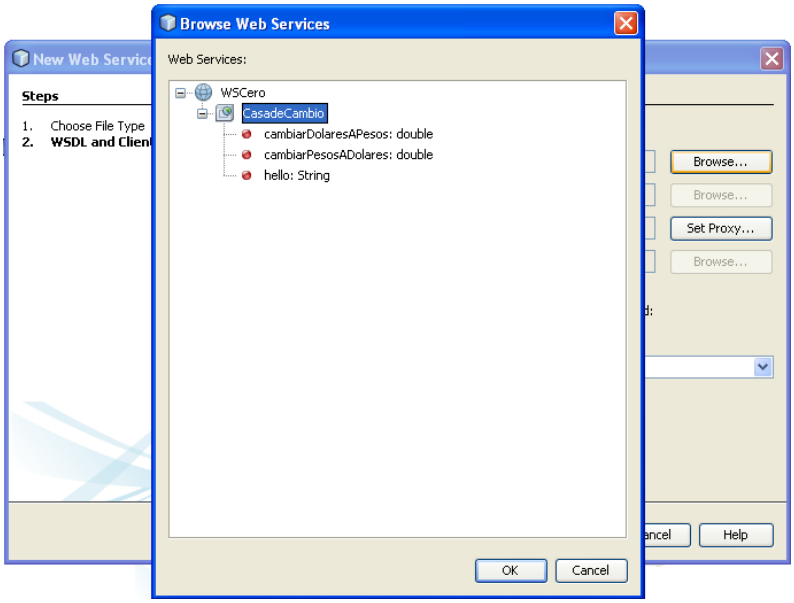


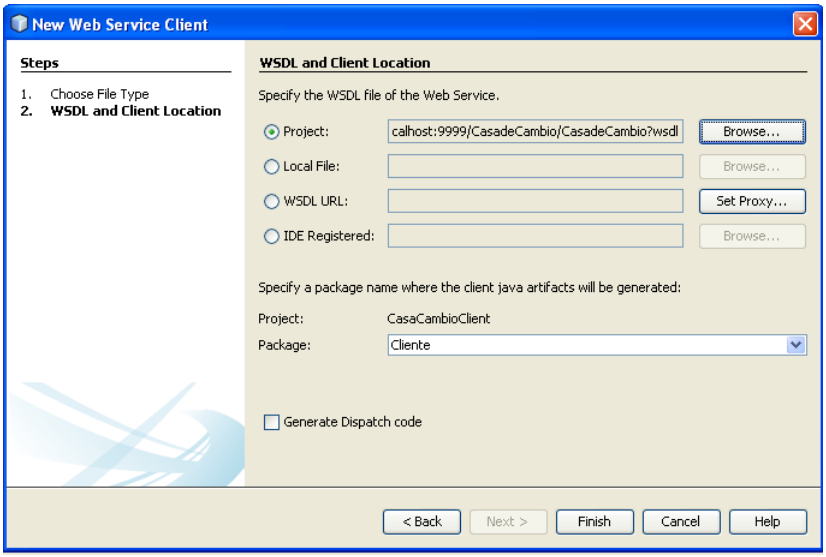
18. Una vez creado el proyecto cliente, se adiciona un paquete denominado Cliente, y se agrega a este nuevo proyecto un nuevo Web Service Client



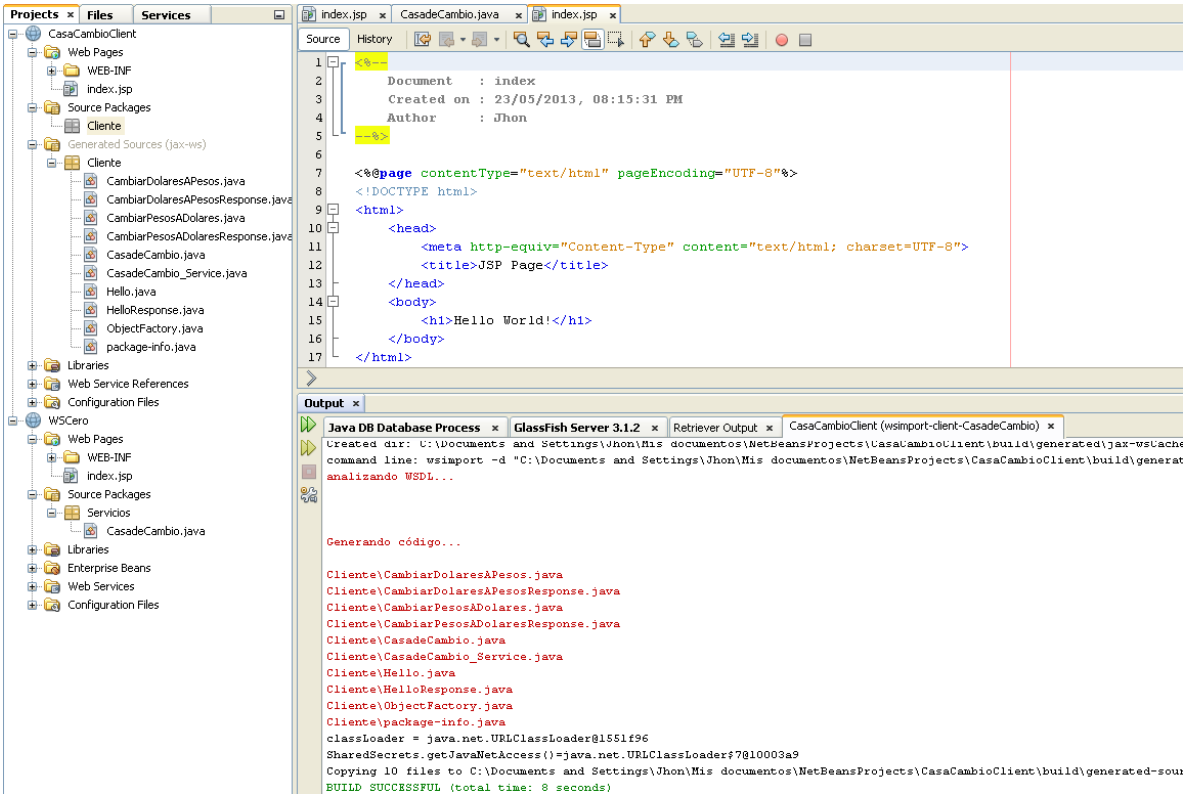
19. En la configuración del Servicio Cliente se agrega la ubicación del servicio que se va a consumir: se selecciona y se da click en “Finish”.


New Web Service Client
Steps
1. Choose File Type
2. WSDL and Client Location
WSDL and Client Location
Specify the WSDL file of the Web Service.
Project:
Local File:
WSDL URL:
IDE Registered:
Browse...
Set Proxy...
Specify a package name where the client java artifacts will be generated:
Project: CasaCambioClient
Package: Cliente
Generate Dispatch code
Enter the URL of the service you wish to use.
< Back
Next >
Finish
Cancel
Help

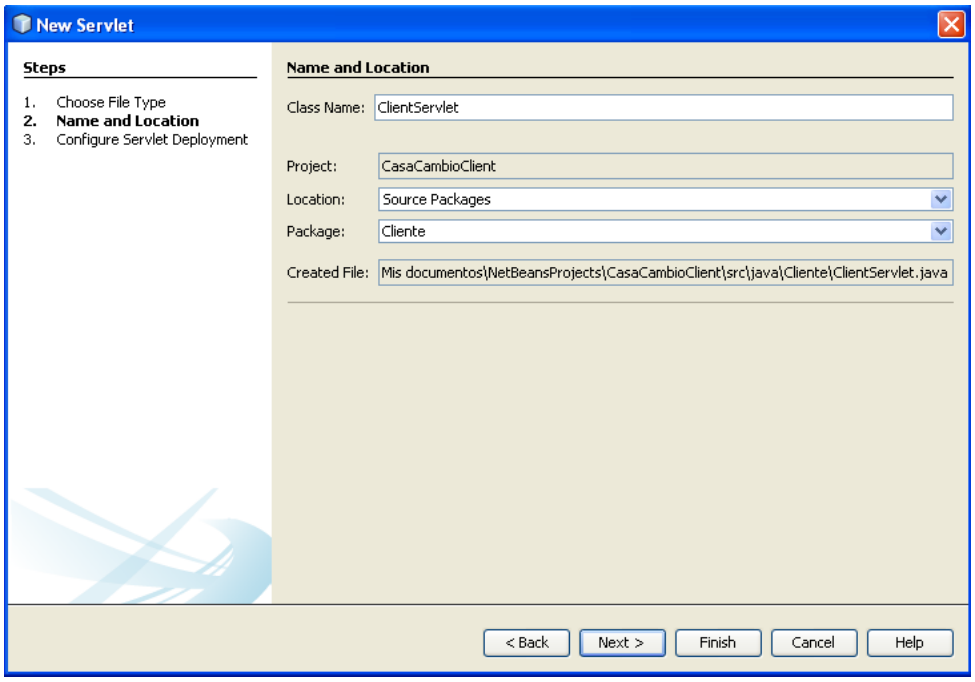

Browse Web Services
Web Services:
WSCero
CasadeCambio
cambiarDolaresAPesos: double
cambiarPesosADolares: double
hello: String
OK
Cancel


New Web Service Client
Steps
1. Choose File Type
2. WSDL and Client Location
WSDL and Client Location
Specify the WSDL file of the Web Service.
Project: calhost:9999/CasadeCambio/CasadeCambio?wsdl
Local File:
WSDL URL:
IDE Registered:
Browse...
Set Proxy...
Specify a package name where the client java artifacts will be generated:
Project: CasaCambioClient
Package: Cliente
Generate Dispatch code
< Back
Next >
Finish
Cancel
Help

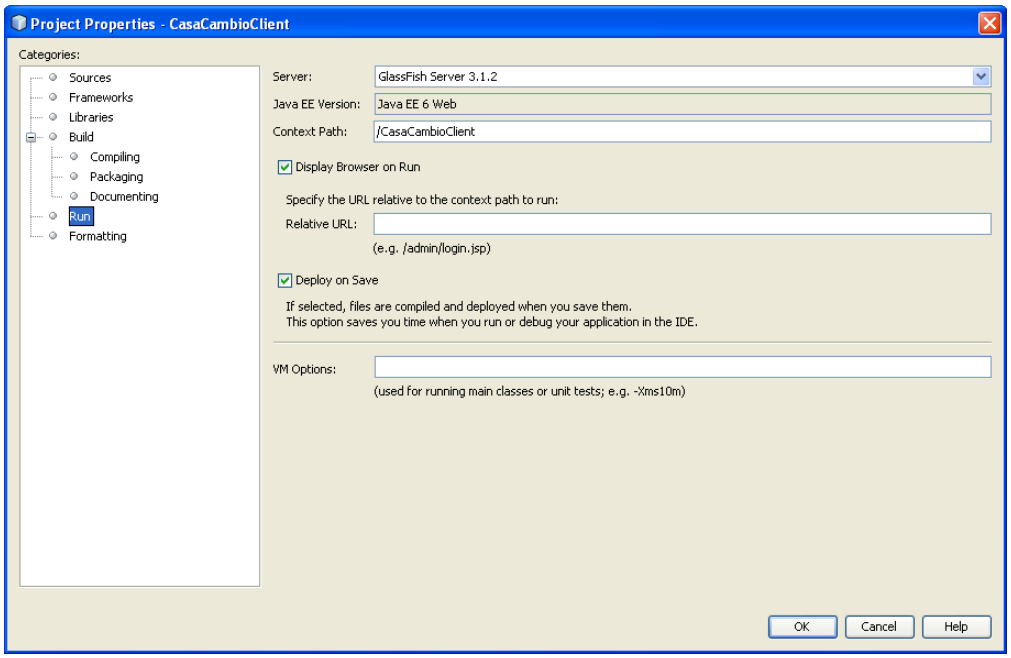
20. Se verifica que en el proyecto cliente se haya creado un nuevo código, el cual indica que ya están asociados.



21. Para iniciar este proceso se necesita de un Servlet en el cliente (ClientServlet).

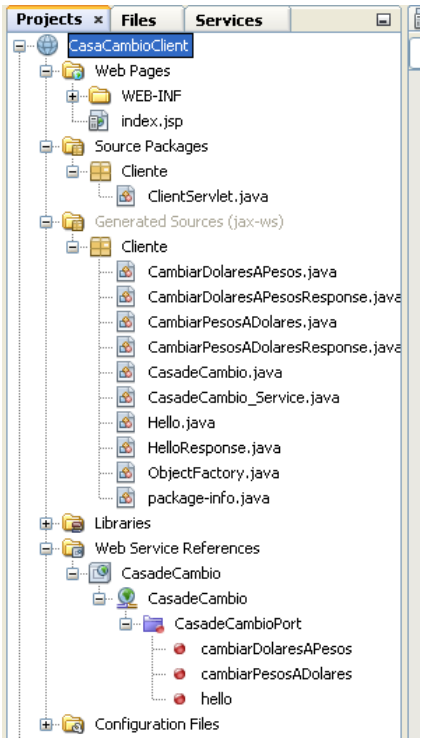


22. Después de crear el Servlet (ClientServlet) se debe configurar para que el servicio levante el servlet y no el hola mundo que viene por defecto. Para ello se debe dar click sobre el nombre del proyecto Cliente, va hasta las propiedades (properties), y en la ventana que se despliega se ubica la característica "RUN".

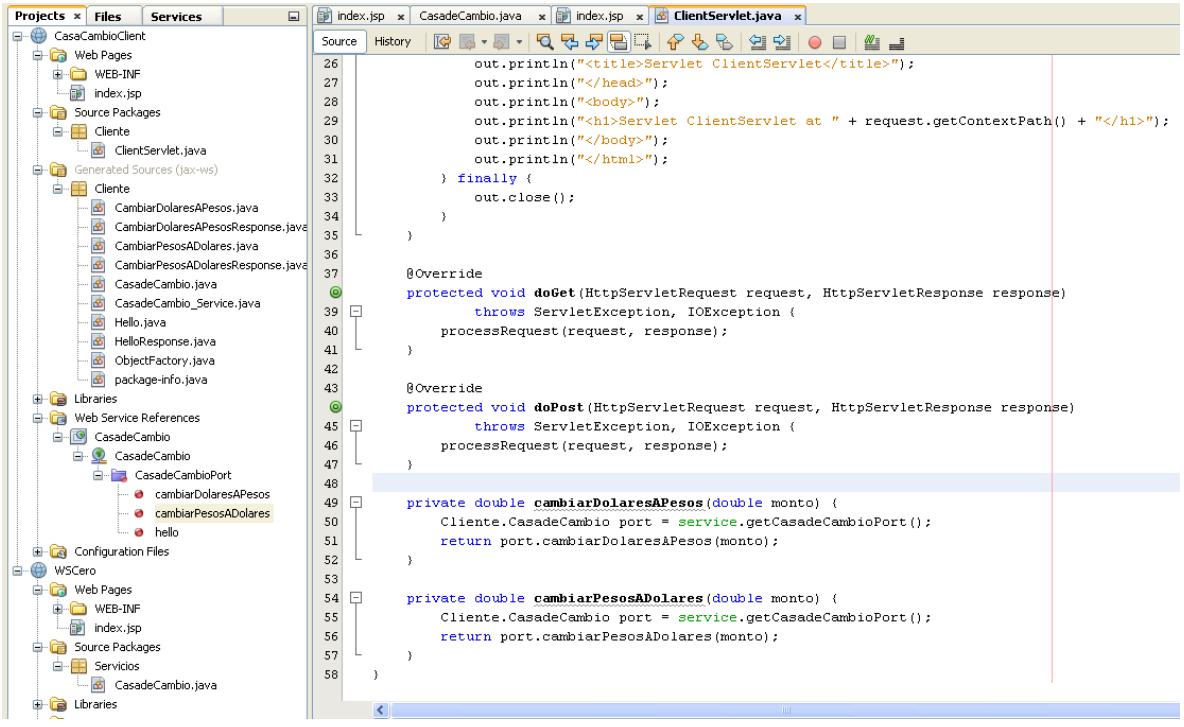


En esta ventana, en la casilla "Relative URL" se coloca el nombre del servlet que se acabo de crear, y se da click en "OK"

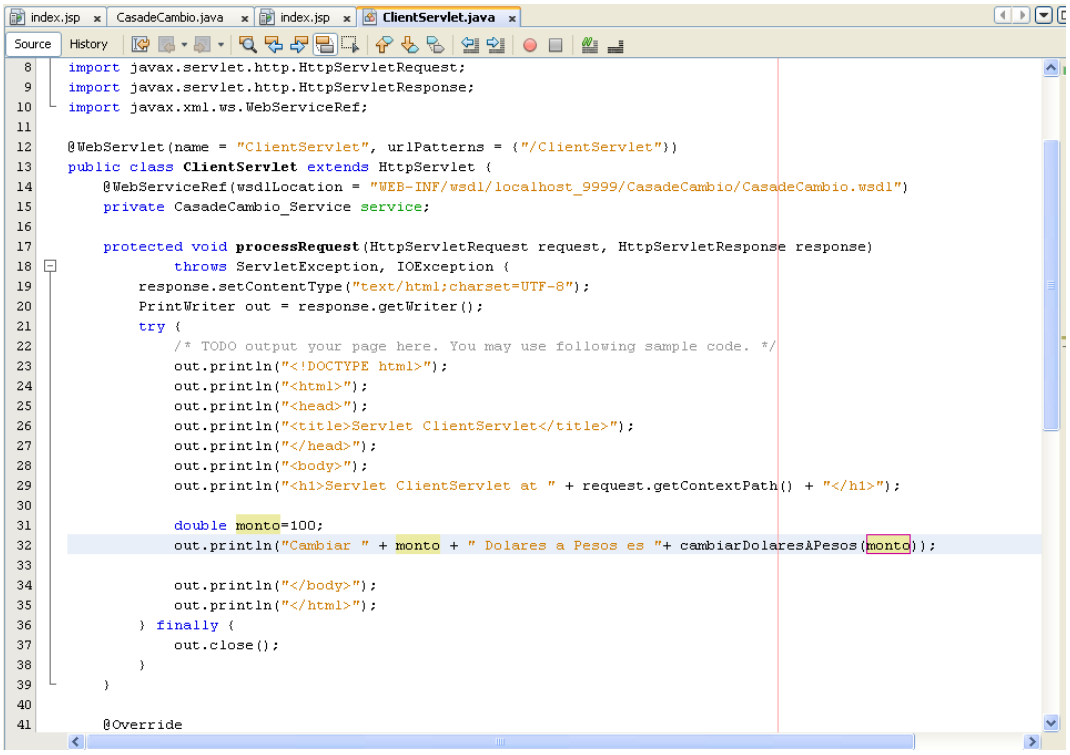
23. A continuación, desde el archivo de la aplicación de cliente, se ubica la carpeta Web Services References



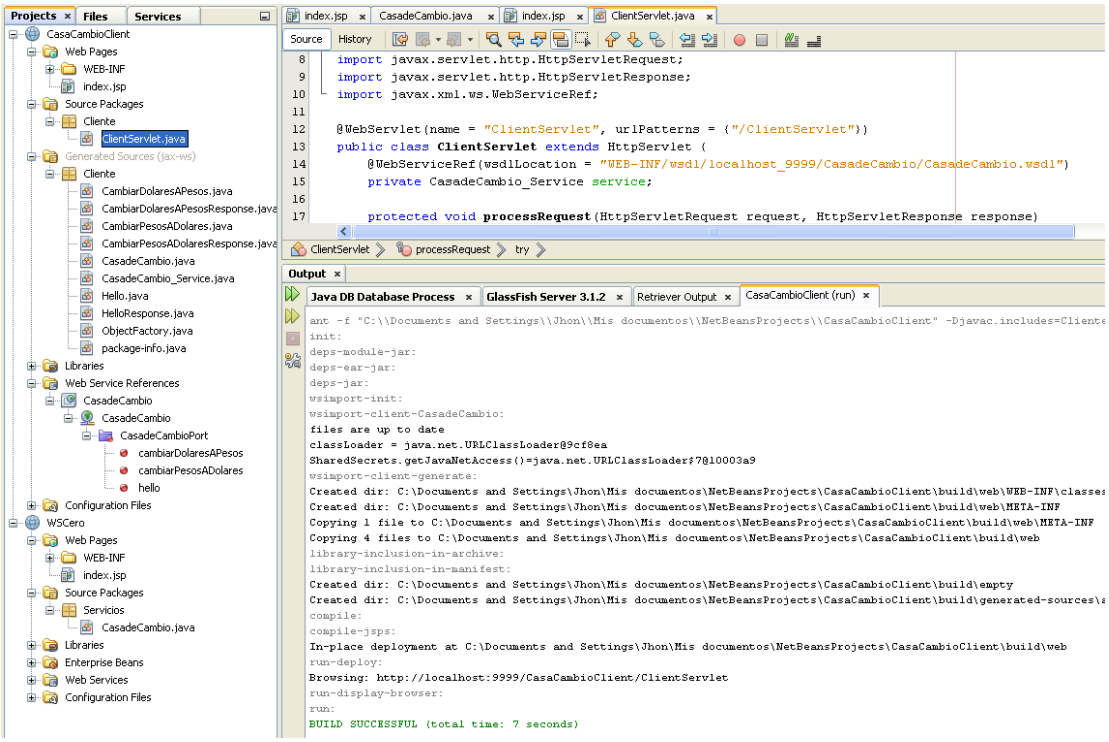
24. Se arrastran los métodos “cambiarDolaresAPesos” y “cambiarPesosADolares” al servlet que se acabo de crear, para poderlos usar. Para ello, de click sostenido sobre el nombre del método (a la izquierda), y arrástrelo hacia el cuerpo del servlet, teniendo en cuenta de soltarlo sobre un espacio que no esté contenido dentro de otro método del servlet.



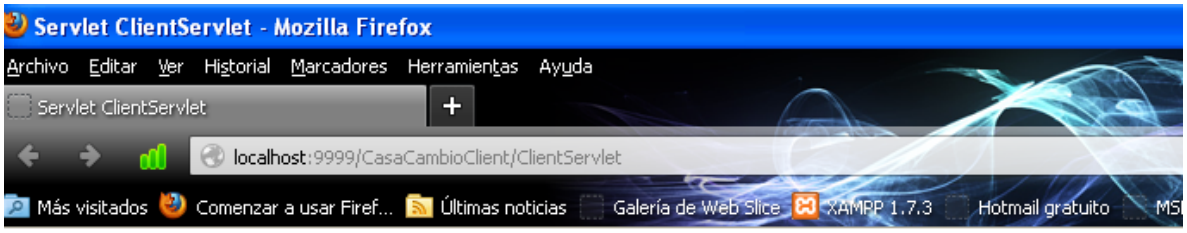
25. En el método **processRequest** del mismo **ServletClient** al que se pasaron los métodos, coloque una llamada al servicio web



26. Terminado este proceso, se da click en el ServLet del Cliente y se inicia la ejecución del cliente



27. En el navegador podrá ver los resultados del proceso



Servlet ClientServlet at /CasaCambioClient

Cambiar 100.0 Dolares a Pesos es 172000.0

Ejercicio

Cree un servicio web que permita convertir una temperatura expresada en Grados Celsius a grados Fahrenheit.

Construye el Cliente Web Service que consuma el WS. Este cliente deberá permitir leer los grados Celsius desde un formulario y consumir el servicio de conversión.