Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 2

Servlets

Alumno: Hernández Salinas Octavio Iván

Profesor: Montes Casiano Hermes Franciso.

Grupo: 3CV3

Fecha de entrega
26 de febrero de 2019

Índice

1. Introducción. **3**

2. Desarrollo. **3**
2.1. Creación del proyecto maven . . . 3
2.2. Configuración del proyecto maven . . . 6
2.2.1. Dependencia para maven . . . 7
2.2.2. Configuración de Jetty . . . 7
2.2.3. Recursos . . . 8
2.2.4. Java Build Path . . . 9
2.3. Servlet 1: Hola Mundo . . . 11
2.4. Servlet 2: Headers del Protocolo HTTP . . . 13
2.5. Servlet 3: Contador . . . 13
2.6. Servlet 4: Contador con la variable como propiedad del servlet . . . 14

3. Conclusiones. **14**

4. Bibliografías y Referencias. **15**

1. Introducción.

Los Java Servlets son una solución eficiente y poderosa para crear contenido dinámico para la Web. En los últimos años, los Servlets se han convertido en el componente fundamental de la corriente principal del lado del servidor Java y es que el poder detrás de Servlets proviene del uso de Java como plataforma y de la interacción con un contenedor Servlet.
Un servlet es una extensión pequeña y conectable a un servidor que mejora la funcionalidad del mismo. Permiten a los desarrolladores extender y personalizar cualquier servidor web o de aplicaciones habilitado para Java con un grado de portabilidad, flexibilidad y facilidad hasta ahora desconocidos. Cuando utilizamos un servlet para crear contenido dinámico para una página web o ampliar la funcionalidad de un servidor web, estamos creando una aplicación web.
Mientras que una página web simplemente muestra contenido estático y permite al usuario navegar a través de ese contenido, una aplicación web provee una experiencia más interactiva. Un servlet es similar a una extensión de servidor propietaria, excepto que se ejecuta dentro de Java Virtual Machine (JVM) en el servidor, por lo que es seguro y portable. Los servlets operan únicamente dentro del dominio del servidor: a diferencia de los applets, no requieren soporte para Java en el navegador web. La plataforma Java proporciona un desarrollador Servlet con una API robusta, programación orientada a objetos, neutralidad de plataforma, tipos estrictos, recolección de basura y todas las características de seguridad de JVM. Para complementar esto, un contenedor Servlet proporciona administración del ciclo de vida, un único proceso para compartir y administrar recursos en toda la aplicación e interacción con un servidor web. En conjunto, esta funcionalidad hace que Servlets sea una tecnología deseable para los desarrolladores de Java del lado del servidor.

2. Desarrollo.

El desarrollo de ésta práctica consiste en realizar cuatro servlets que realicen las siguientes acciones:

1. Programar un servlet que muestre en pantalla el mensaje "Hola mundo".

2. Implementar un servlet que muestre en el navegador los headers del protocolo HTTP que se están enviando desde el cliente al servidor.

3. Implemente un servlet que atienda una peticion GET y en el método service de atención implemente una variable contador inicializada en 0, y posteriormente muestre su valor en pantalla y en cada petición incrementar su valor.

4. Con base en el punto anterior mueva la variable implementada en el método service como propiedad del servlet. ¿Qué observa? ¿Qué diferencia existe entre la salida de éste ejercicio y el anterior?, justifique sus respuestas.

2.1. Creación del proyecto maven

Cuando utilizamos por primera vez *Eclipse IDE* es necesario que configuremos ciertas características para que éste pueda utilizar los componentes que queremos que utilice ya que de otra forma nos marcará algunos errores de configuración o de que no detecta algunos componentes. Éstos errores aparecen después de que se haya creado el proyecto ***Maven***, mismo que se crea como se muestra en la figura 1

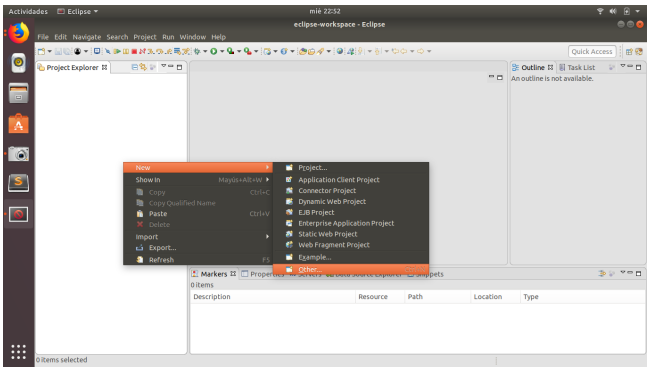

Figura 1: Creación de un nuevo proyecto

Cuando seleccionamos la opción “Other” nos aparece la figura 2 en la cual tenemos que seleccionar la opción “Maven Project” y continuar con el boton “Next” que nos lleva a la figura 3

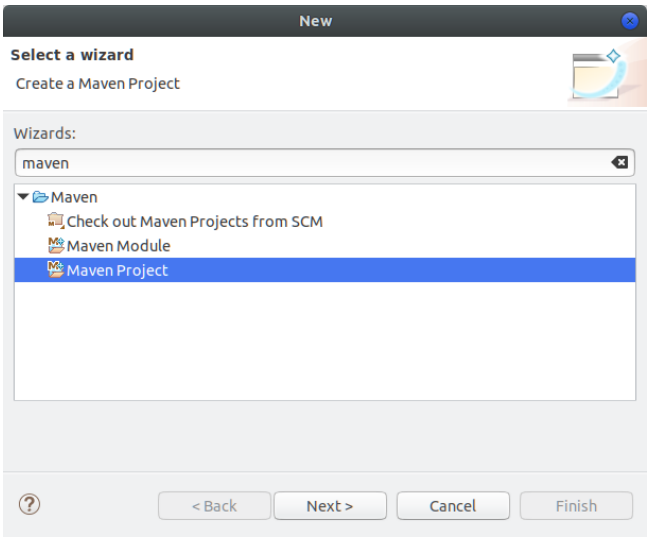

Figura 2: Creación de un proyecto Maven

En esta ventana seleccionamos la opción del Workspace que en la instalación del IDE se definió como predeterminada como se muestra en la figura 3 y presionamos el botón “Next”.

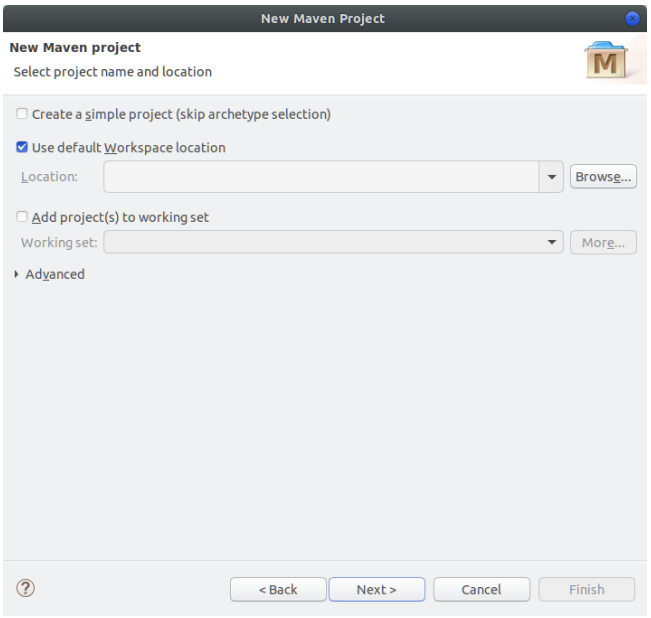

Figura 3: Escogiendo la ubicación donde se alojará el proyecto

Para que podamos desarrollar una aplicación web es necesario que escojamos el arquetipo de *maven-webapp* como se muestra en la figura 4

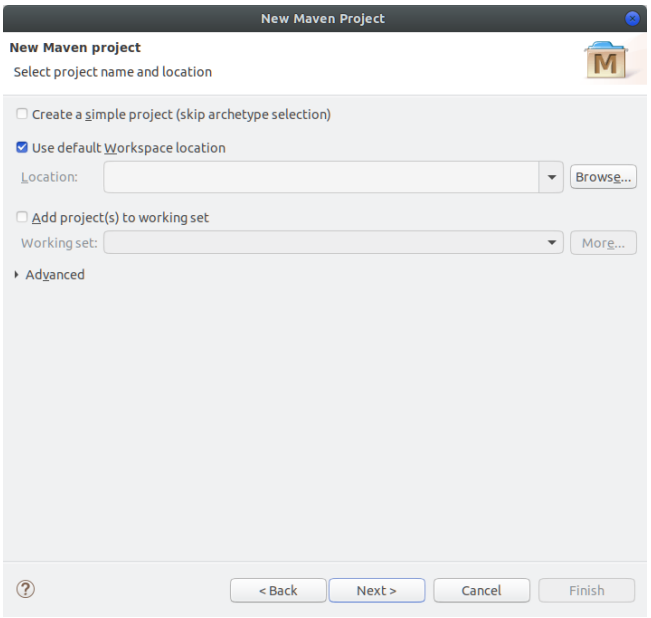

Figura 4: Arquetipo para aplicaciones web

Al seleccionar este arquetipo, se nos pedirán los parámetros para el mismo los cuales se muestran en la figura 5 finalizamos con la creación del proyecto maven y nos apareace la figura 6

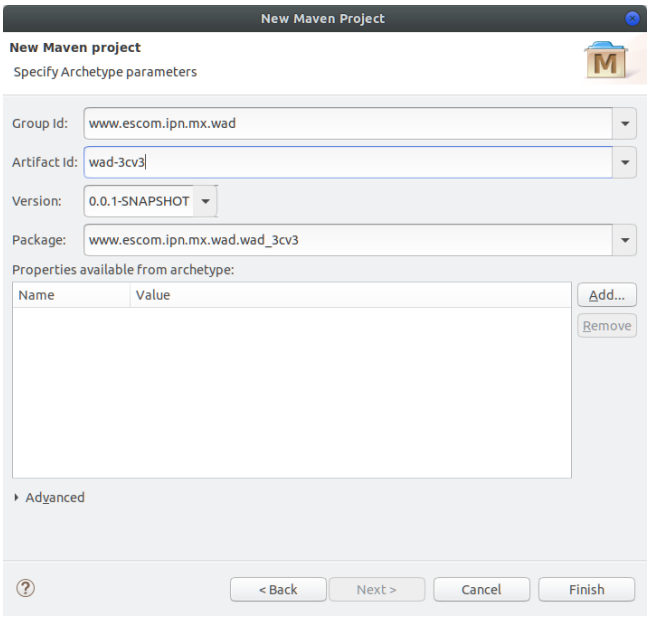

Figura 5: Parámetros para el arquetipo

Al finalizar la creación de nuestro proyecto maven, nos dice que la biblioteca nativa de Subversion no está disponible, cuando presionemos el botón "OK" nos debe abrir el proyecto como fue creado.

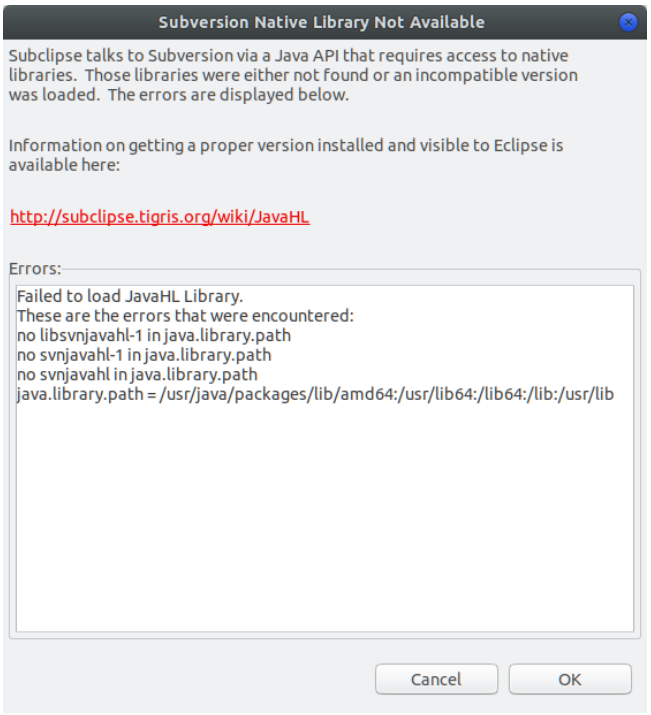

Figura 6: Errores al terminar la creación del proyecto maven

2.2. Configuración del proyecto maven

Para corregir los errores que nos marcar el IDE y podamos deployar nuestra aplicación es necesario realizar varias cosas, mismas que se explican más adelante, y es al ser un proyecto maven, éste necesita

ciertas dependencias con las que no cuenta al momento de crearce el proyecto, por lo tanto, ésto será lo primero que se realice.

2.2.1. Dependencia para maven

En el navegador tenemos que ingresar a la página de los repositorios de maven y buscar especificamente el Java Servlet API en su versión 4.1. Deberemos copiar la dependencia para maven como se muestra en la figura 7

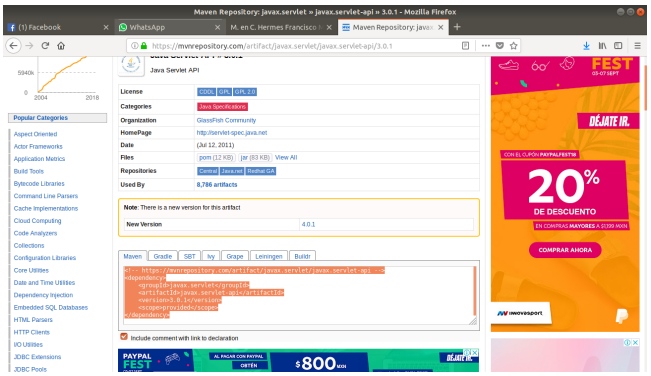

Figura 7: Repositorio de Maven

La dependencia que hemos copiado la tenemos que agregar al archivo pom.xml que es donde se encuentran declaradas las dependencias que necesita nuestro proyecto para funcionar correctamente 8.

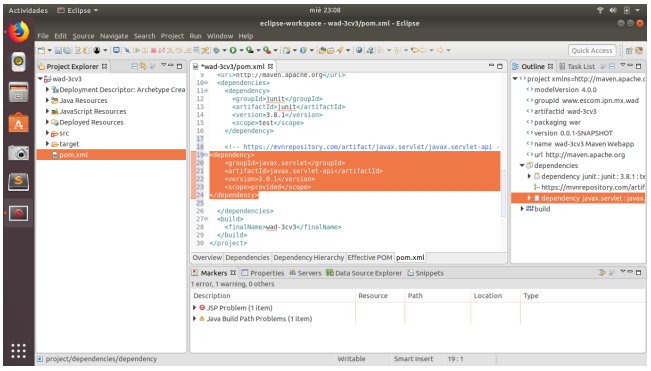

Figura 8: Agregando la dependencia de maven

2.2.2. Configuración de Jetty

Para que podamos deployar nuestra aplicación en el contenedor de Jetty es necesario que le configuremos el puerto con el que viene por default, ya que ese puerto es el mismo que utiliza nuestro contenedor Apache Tomcat 9.

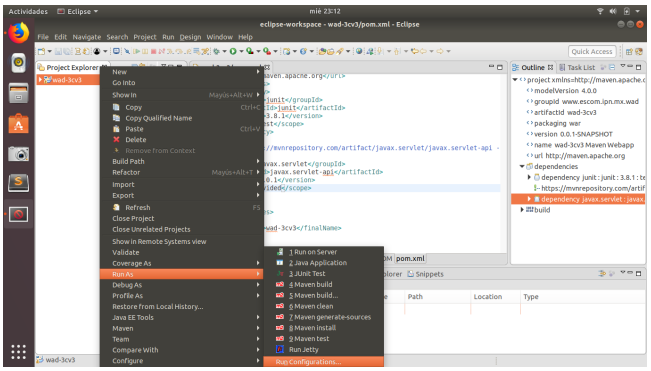

Figura 9: Entrando a las configuraciones de ejecución

El puerto que se le asignará a Jetty será el puerto 8081 como se muestra en la figura 10

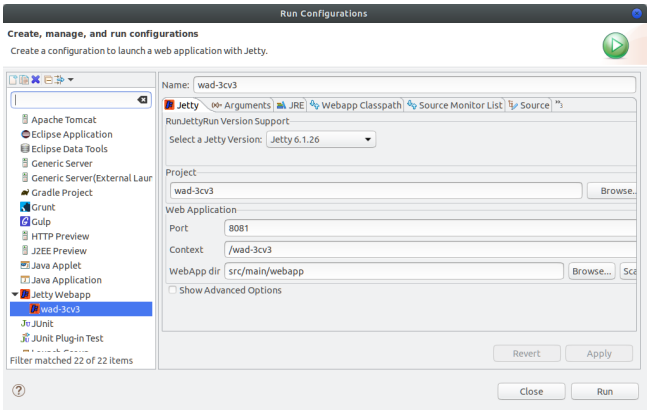

Figura 10: Configurando el puerto para Jetty

Para este momento ya debe correr nuestra aplicación en el puerto 8081.

2.2.3. Recursos

Para optimizar el tiempo, vamos a seleccionar la carpeta webapp como carpeta fuente para agregarla como recurso de java como se muestra en la figura 11

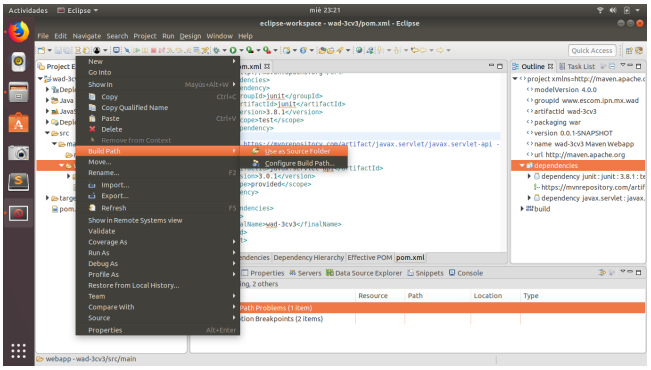

Figura 11: Seleccionando webapp como carpeta fuente

Vamos a crear un folder en la carpeta main de para ingresar ahí los archivos .java 12

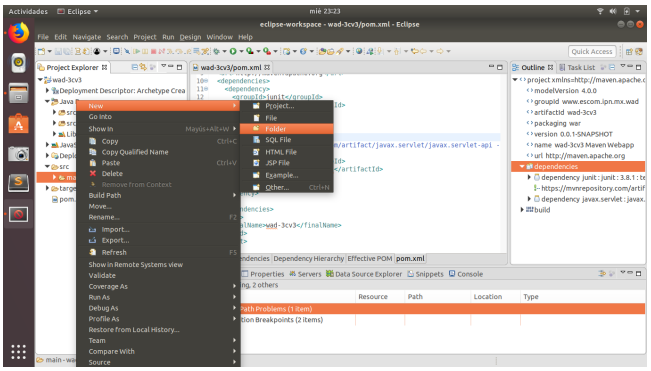

Figura 12: Nuevo folder Java

En el folder creado previamente vamos a crear un paquete en el que se almacenarán los servlets que creemos más adelante 13.

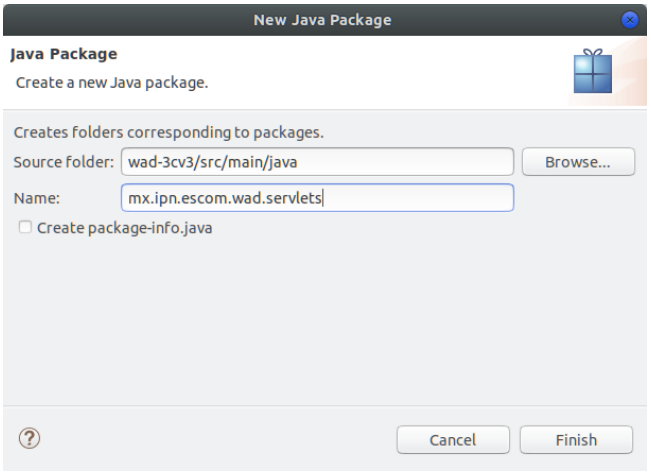

Figura 13: Package servlets

2.2.4. Java Build Path

Para corregir el último error que nos marca nuestro IDE, tenemos que ingresar a las propiedades de nuestro proyecto y en *Java Build Path* >Libraries vamos a eliminar la librería que tiene y sustituirla por la nuestra (JDK) 14.

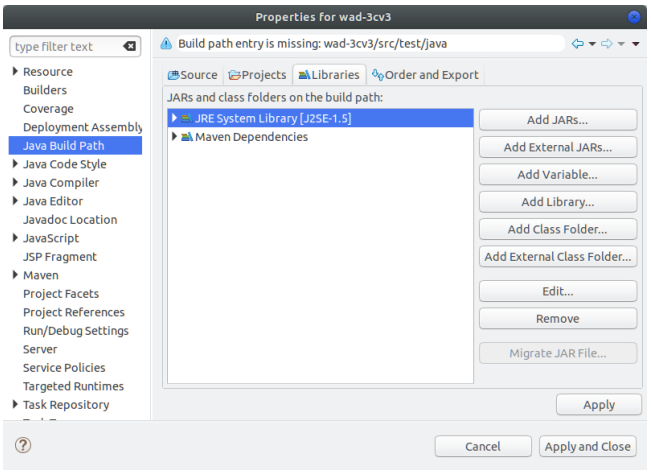

Figura 14: Librería por defecto

Cuando presionamos el botón Add Library, nos muestra la figura 15 en la cual escogeremos la opción de *JRE System Library* y presionaremos el botón "Next"

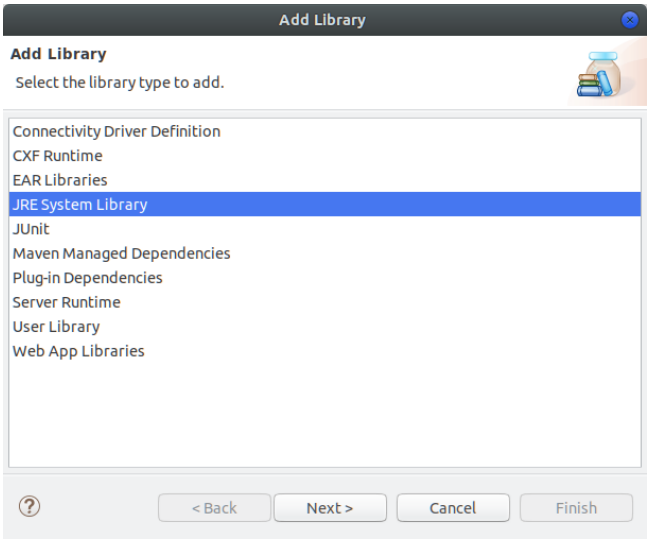

Figura 15: Añadiendo una librería

Finalmente marcaremos la opción mostrada en la figura 16

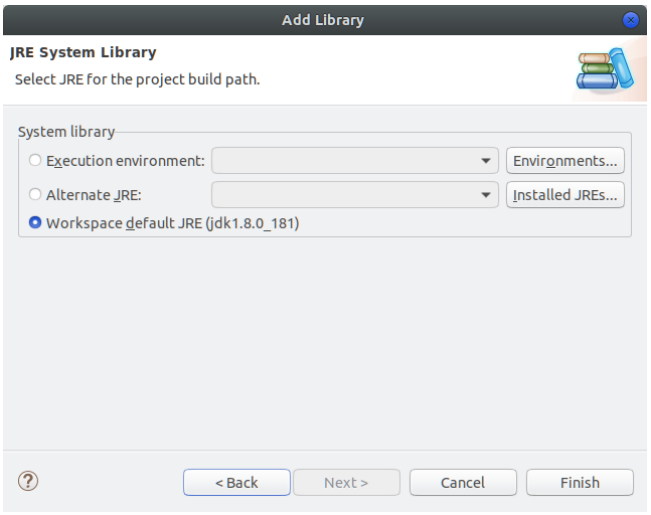

Figura 16: Seleccionando el nuevo JRE

2.3. Servlet 1: Hola Mundo

El primer servlet a realizar consiste en un muestre en pantalla el mensaje "Hola Mundo". Para crear dicho servlet, lo tenemos que ubicar en el paquete que creamos en el forlder Java 17.

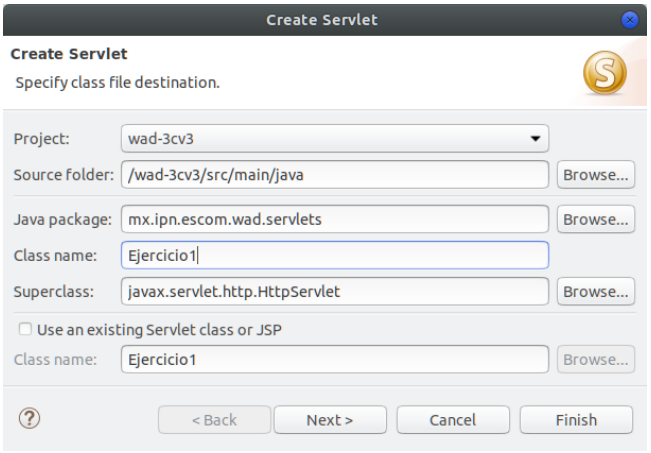

Figura 17: Creando un servlet

En la figura 18 se escogen lo métodos con los que vamos a trabajar, en este caso basta con los métodos Get y Post.

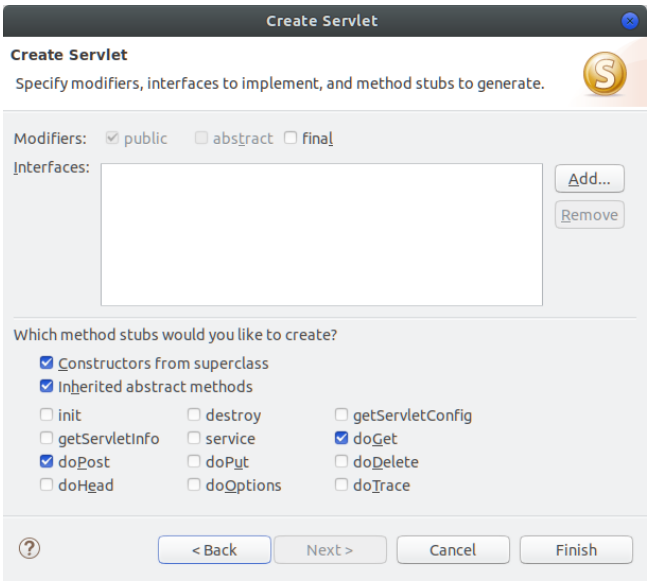

Figura 18: Métodos para el servlet

Cuando creamos nuestro Servlet podemos ver que nuestro archivo web.xml ha añadido la definición y mapeo del servlet que acabamos de crear 19.

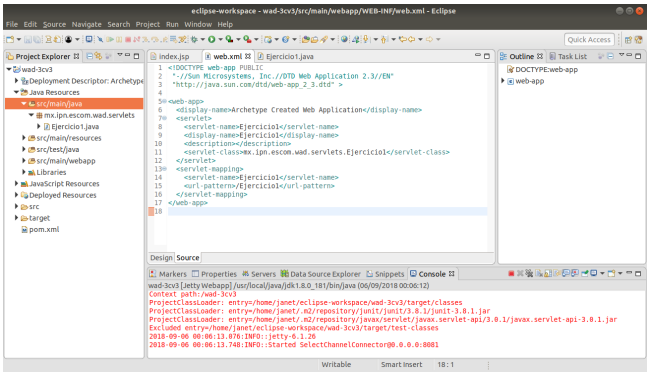

Figura 19: Definiendo el servlet y mapeandolo

El servlet como tal se encuentra en nuestro archivo .java, y es aquí en donde vamos a programar la funcionalidad que le queremos dar 20 .

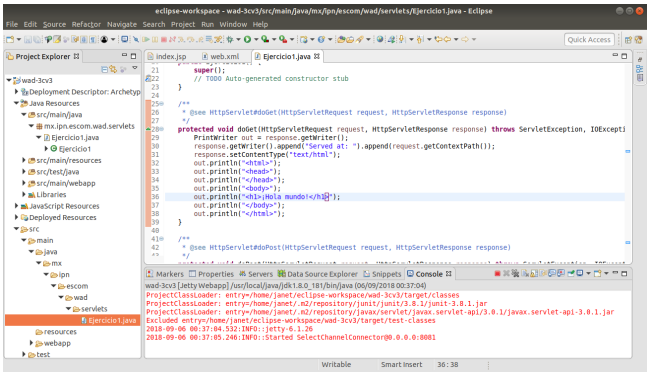

Figura 20: Creación del Servlet

Cuando ejecutemos el proyecto, podremos ver en el navegador que nuestro servlet 1 hace lo que tiene que hacer.

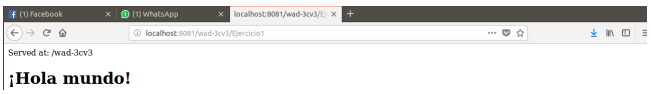

Figura 21: Prueba del servlet 1 funcionando correctamente

2.4. Servlet 2: Headers del Protocolo HTTP

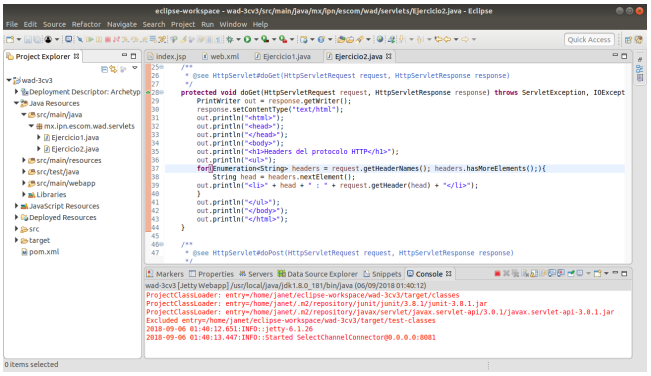

Figura 22: Creación del Servlet 2

Figura 23: Prueba del servlet 2 en el navegador

2.5. Servlet 3: Contador

En este servlet podemos ver que pasa cuando se declara una variable contador inicializada en 0 e imprimimos su valor.

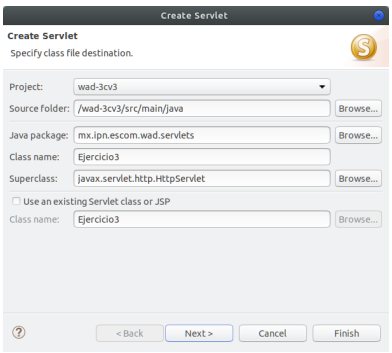

Figura 24: Creación del Servlet 3

El resultado que se obtiene es que la variable contador nunca va a incrementar su valor debido al alcance del request y sin importar cuantas veces recargue la página éste siempre será 0. El primer alcance es el de request y todo lo que se instancie dentro del método Service, doGet, doPost, etc, en cuanto salga, en ese momento el servlet le devuelve el request y el response al contenedor, mismo que se asegura de limpiarlo, de hacer un flush correcto, y de que todo se ejecute de manera satisfactoria y entonces, todo se borra. Para cuando se realicen peticiones sucesivas se creará una nueva instancia.

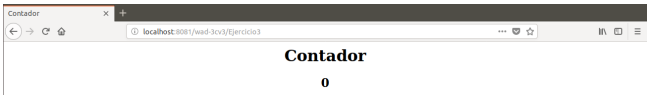

Figura 25: Prueba del servlet 3 en el navegador

2.6. Servlet 4: Contador con la variable como propiedad del servlet

En el servlet anterior pudimos observar que la variable contador implementada dentro del método nunca incrementaba su valor, entonces, en este servlet que pasará si la ponemos como propiedad del mismo y no implementada dentro de un método.

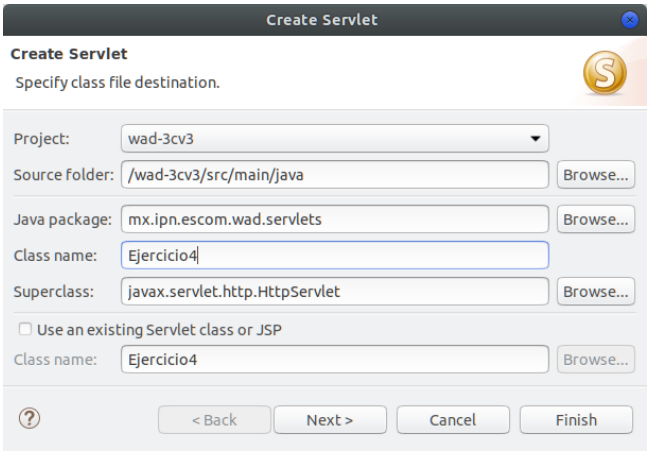

Figura 26: Creando el servlet 4

Existe un scope con mayor alcance que request, llamado session, y existe un scope más grande aún llamado scope de application, los servlets son la única instancia que existe en éste scope y lo que estamos haciendo en éste servlet es guardar una propiedad en un objeto único para la aplicación. Hay que recordar que los servlets son multithread por naturaleza, entonces si se pone un atributo dentro del mismo lo estamos dotando de un estado. Como resultado, podremos ver que la variable contador se va a incrementar cada vez que hagamos una petición.

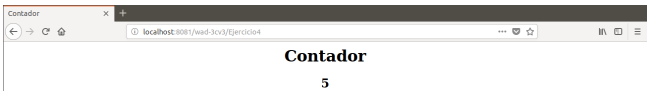

Figura 27: Prueba del servlet 4 en el navegador

3. Conclusiones.

Al finalizar esta práctica podemos estar conscientes de que básicamente los Servlets consisten en objetos que representan la petición de un cliente, HttpServletRequest, la respuesta del servidor,

HttpServletResponse y toda una aplicación web, ServletContext. Cada uno de estos objetos proporciona un conjunto completo de métodos para acceder y manipular información relacionada. Vimos como la implementación de una variable puede ser importante dependiendo el método en el que sea declarada y además somos conscientes de que no es conveniente ponerle atributos a un servlet ya que este puede ser modificado.

4. Bibliografías y Referencias.

1. Jason Hunter, William Crawford. (2001). Java Servlet Programming. Estados Unidos: O'Reilly.
2. Jayson Falkner, Kevin Jones. (2004). Servlets and JavaServer Pages. U.S: Pearson Education.