UD1 – Examen (Modelo A)

POR FAVOR, LEE LAS INSTRUCCIONES ATENTAMENTE. DEBERÁS ENTREGAR ESTE MISMO DOCUMENTO EN FORMATO PDF (**CUALQUIER OTRO FORMATO DISTINTO DE PDF NO SERÁ CORREGIDO**). SE ESPERA QUE PUEDAS RESOLVER ESTAS ACTIVIDADES EMPLEANDO EXCLUSIVAMENTE TUS CONOCIMIENTOS Y LOS MATERIALES QUE HAYAS DESCARGADO PREVIAMENTE. EN CASO DE CUALQUIER EVIDENCIA DE COPIA, EL EXAMEN SE EVALUARÁ COMO SUSPENSO (0).

Pregunta 1. Crea una web simple:

- Se recomienda que utilices HTML. Se valorará positivamente el uso de hoja de estilos.
- Debe de incluir una sola página, la cual debe contener una imagen. Si no dispones de imágenes en tu equipo puedes tomar una captura de pantalla.
- Ha de ser desplegada en tu servidor Apache.

Lo primero es crear los archivos necesarios. Una página Web básica llamada index.html y su archivo de estilos correspondiente llamado styles.css

A partir de ahora debemos de ir realizando los pasos necesarios para desplegarlo en Apache.

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Alejandro Sainz</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css" rel="stylesheet">
  <link rel="stylesheet" href="styles.css">
</head>
<body>


  <!-- Carrusel -->
  <div class="window mb-4 p-3">
    <div class="window-title">Carrusel</div>
    <div id="colorCarousel" class="carousel slide mt-3" data-bs-ride="carousel">
      <div class="carousel-inner text-center">
        <div class="carousel-item active">
          <div class="d-flex justify-content-center">
            <div class="square bg-pastel-blue"></div>
            <div class="square bg-pastel-pink mx-3"></div>
            <div class="square bg-pastel-green"></div>
          </div>
        </div>
        <div class="carousel-item">
          <div class="d-flex justify-content-center">
            <div class="square bg-pastel-yellow"></div>
            <div class="square bg-pastel-purple mx-3"></div>
            <div class="square bg-pastel-cyan"></div>
          </div>
        </div>
      </div>
      <button class="carousel-control-prev" type="button" data-bs-target="#colorCarousel" data-bs-slide="prev">
```

Aquí podemos ver algo del código html.

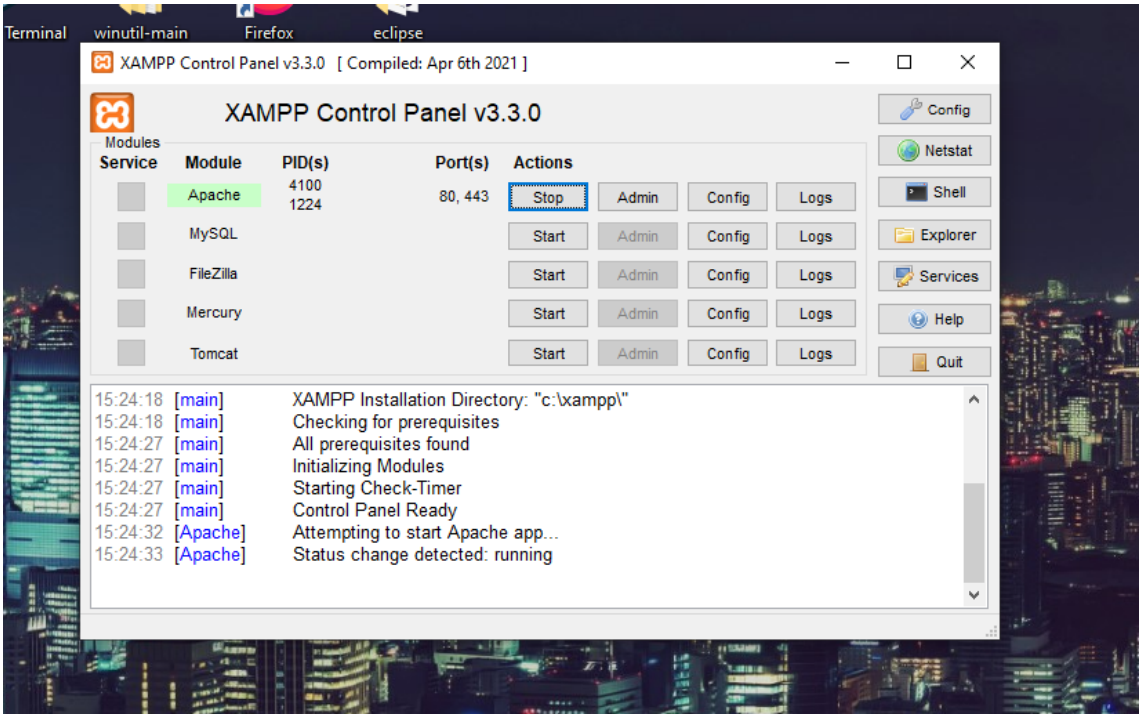
```
Archivo  Edición  Formato  Ver  Ayuda
body {
  background-color: #1e1e2e;
  color: #e0e6ed;
  font-family: Consolas, "Courier New", monospace;
  margin: 0;
  padding: 0;
}

/* Ventanas con borde y título estilo Hyprland */
.window {
  background-color: #2a2a3b;
  border: 1px solid #44475a;
  border-radius: 8px;
  box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.4);
  position: relative;
  overflow: hidden;
}

.window-title {
  background-color: #383a59;
  color: #d0d8e0;
  font-size: 0.9rem;
  padding: 6px 12px;
  border-bottom: 1px solid #44475a;
  text-align: left;
}

.square {
  width: 150px;
  height: 150px;
  border-radius: 6px;
}

/* Paleta pastel inspirada en Hyprland */
```

El css que le he asociado.

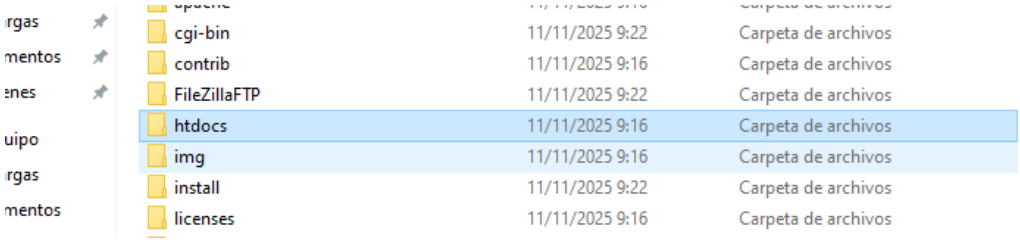
Para completar este ejercicio, deberás proporcionar lo siguiente:

1. Una captura de pantalla mostrando tu sitio web, incluyendo la imagen y la url de la barra de direcciones. (1 punto)

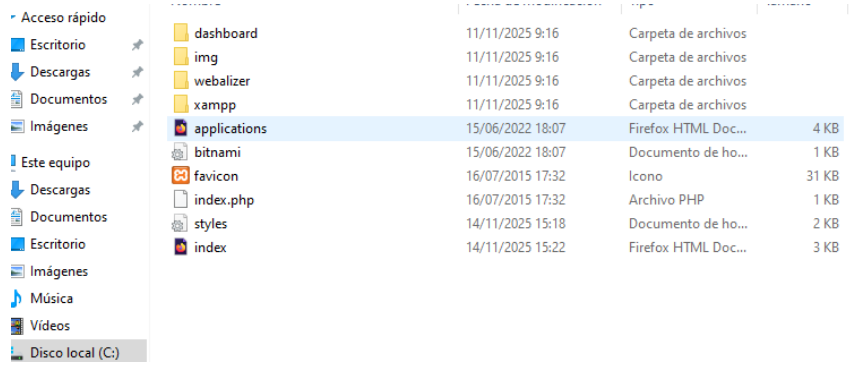
Lo primero vamos a arrancar Apache.



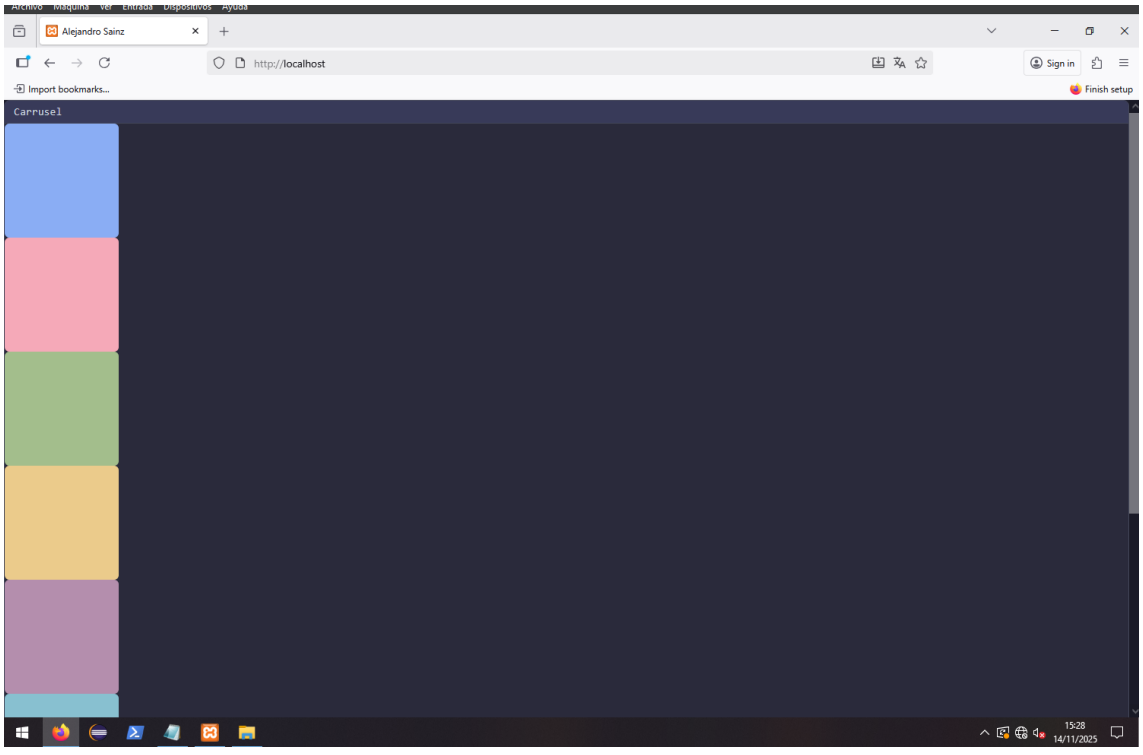
Abro Xampp y lanzo el servicio de Apache.



Dentro de Apache tenemos que ir a HTDOCS



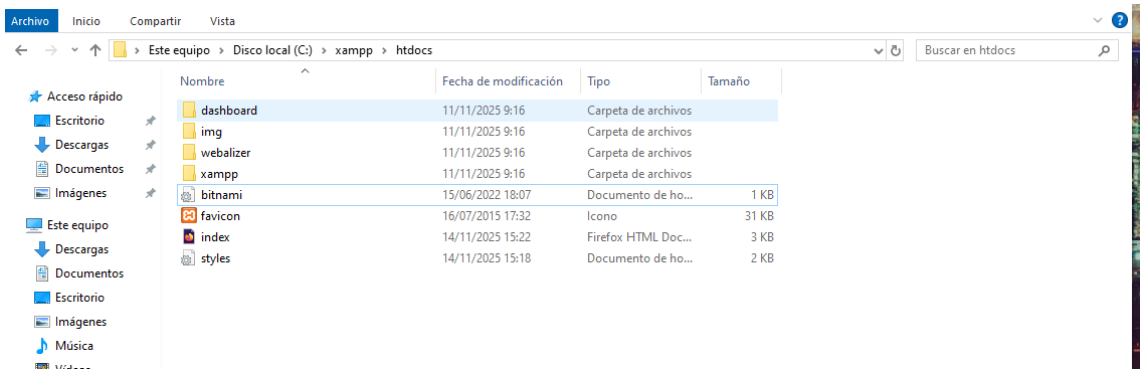
Copiamos nuestros archivos dentro de la carpeta



Si abrimos nuestro navegador y escribimos http://localhost nos aparecerá la página. Yo en este caso he elgido divs del mismo tamaño con diferentes colores. Para no volvernos locos.

2. Una captura mostrando si has modificado algo de los ficheros de configuración de Apache y la carpeta donde tu html/css/imagen han sido desplegados, así como una breve descripción de los pasos que has seguido para el despliegue. (2 puntos)

En los puntos anteriores ya he comentado lo que tenia que hacer, pero hay que puntualizar una cosa. Cuando copiamos nuestros archivos dentro de esa carpeta debemos de asegurarnos que hemos borrado los propios de Apache. Apache tiene por defecto un par de archivos, uno .html y otro .php que son su página de inicio cuando escribimos http://localhost, se ven en una captura anterior. Si no borramos esos archivos, o los cambiamos de folder o de nombre, cuando busquemos la página inicial, Apache siempre mostrará la suya en vez de la nuestra. Una vez eliminados o movidos ya no tenemos ningún problema, salvo lo que tarde el navegador.



Aquí tenemos de Nuevo la captura una vez eliminados los archivos propios de Apache. Puede que solo necesitemos eliminar el archive HTML. Pero nunca se sabe, yo por si acaso siempre elimino los dos.

Pregunta 2. Los litros por cada 100 kilómetros (l/100 km) son una unidad de medida del consumo de combustible utilizada principalmente en Europa y en otros países que emplean el sistema métrico. En esta escala, un valor más bajo indica un menor consumo de combustible. En cambio, en Estados Unidos se utiliza con más frecuencia la unidad millas por galón (mpg), donde un valor más alto representa una mayor eficiencia. Para convertir de litros por 100 km a millas por galón, podemos emplear la siguiente operación matemática:

$$mpg = \frac{235{,}214}{l/100\ km}$$

Crea un servlet o cualquier otra aplicación que puedas ejecutar en tu servidor Tomcat. Tu aplicación web debe aceptar un consumo en litros a los 100 (como número decimal) y tras pulsar un botón, deberá mostrar su conversión a mpg.

Para completar este ejercicio, deberás proporcionar lo siguiente:

1. Dos capturas de tu aplicación web mostrando el entorno Tomcat (barra de direcciones), incluyendo tanto la pantalla principal como el resultado de la ejecución. Convierte un consumo de 8l/100 km a mpg. Explica los pasos que has seguido para conseguirlo. (2 puntos)
2. El impuesto sobre el consumo de combustible (gas guzzler tax) se aplica a la venta de automóviles en Estados Unidos que no cumplen con ciertos estándares de ahorro de combustible (tabla 1). Haz que tu aplicación calcule el importe del impuesto según el consumo facilitado por el usuario. Muestra dos ejemplos del funcionamiento diferentes (4 litros a los 100 km y 20 litros a los 100 km). (2 puntos)

mpg desde	mpg hasta	Impuesto ($)
22,5	-	0
17,5	22,4	2.000
12,5	17,4	4.000
-	12,4	7.700

Tabla 1. Impuesto sobre el consumo de combustible

En ambos casos, se valorará positivamente el uso de hoja de estilos para lograr una presentación profesional de tu desarrollo.

Para esta actividad vamos a necesitar crear un Dynamic Web Project. Para ello tendremos que lanzar eclipse y crear ese proyecto. Una vez creado el mismo vamos a tener que crear la parte de la lógica, el servlet, y también la parte dinámica, el archivo jsp.

Para este ejercicio no me voy a complicar mucho. Lo que hago es usar una base que ya tengo de otro ejercicio, simplemente porque ya tengo creado en el un método post en el jsp con un input.

```
8  </head>
9  <body>
10     <form action="MultiplicationServlet" method="post">
11     <input type="number" name="num1" placeholder="Litros/100Km" required>
12     <button type="submit">Calcular</button>
13 </body>
14 </html>
```

Evidentemente, aunque tenga esto, todavía me queda la parte más difícil que es el back. Que también se llama Multiplicación. Al fin y al cabo, las dos operaciones son una multiplicación. Ahora debo de coger mi servlet y modificarle para que realice las operaciones necesarias.

```
18
19     /**
20      * @see HttpServlet#HttpServlet()
21      */
22     public MultiplicationServlet() {
23         super();
24         // TODO Auto-generated constructor stub
25     }
26
27     /**
28      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse r
29      */
30     protected void doPost(HttpServletRequest request, HttpServletResponse respor
31             throws ServletException, IOException {
32
33         double n1 = Double.parseDouble(request.getParameter("num1"));
34         double n2 = Double.parseDouble(request.getParameter("num2"));
35         double resultado = n1 * n2;
```

Este es el servlet que tenía de proyecto para las primeras pruebas y tareas, ahora lo que toca es modificarle.

Ahora si no he entendido mal, para calcular las millas por galón debo de dividir una cantidad entre el consumo de litros a los 100. Para esto debo de preparar el código.

```
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse respon
            throws ServletException, IOException {

        double n1 = Double.parseDouble(request.getParameter("num1"));
        double resultado = 235.214 / n1;
```

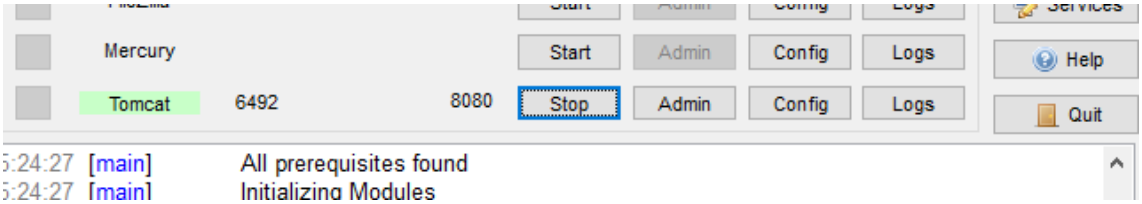
En principio este debería de ser el único cambio que tengo que realizar por ahora. Ahora viene la siguiente parte, que es devolver el resultado, ya sea en la misma página o en otro jsp distinto.

Para este caso, yo voy a mostrarlo en otro archivo. Es la forma en la que lo he hecho otras veces.

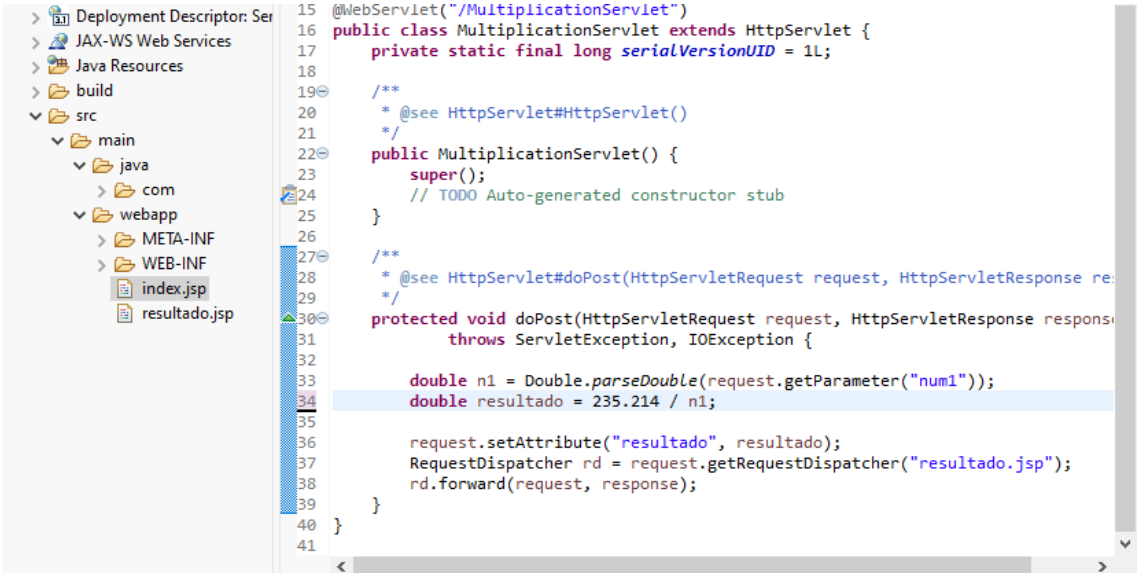
```
    request.setAttribute("resultado", resultado);
    RequestDispatcher rd = request.getRequestDispatcher("resultado.jsp");
    rd.forward(request, response);
}
```

Para conseguirlo debo de hacer lo que se ve en la imagen superior. Para la función del post instanciamos dos objetos de la clase HTTPServletRequest. Uno es request, que son los datos que recibimos de la web o interface. El otro es el response, que es el que vemos aquí. Con el response lo que hacemos es, o bien devolver código html junto a variables de java, un poco al estilo de lo que se puede hacer en JavaScript o PHP pero sin ser lo mismo. Aquí con getRequestDispatcher y forward lo que hacemos es decirle a nuestro servlet que la respuesta se debe de servir y mandar a otro archivo .jsp, A otra página vamos.

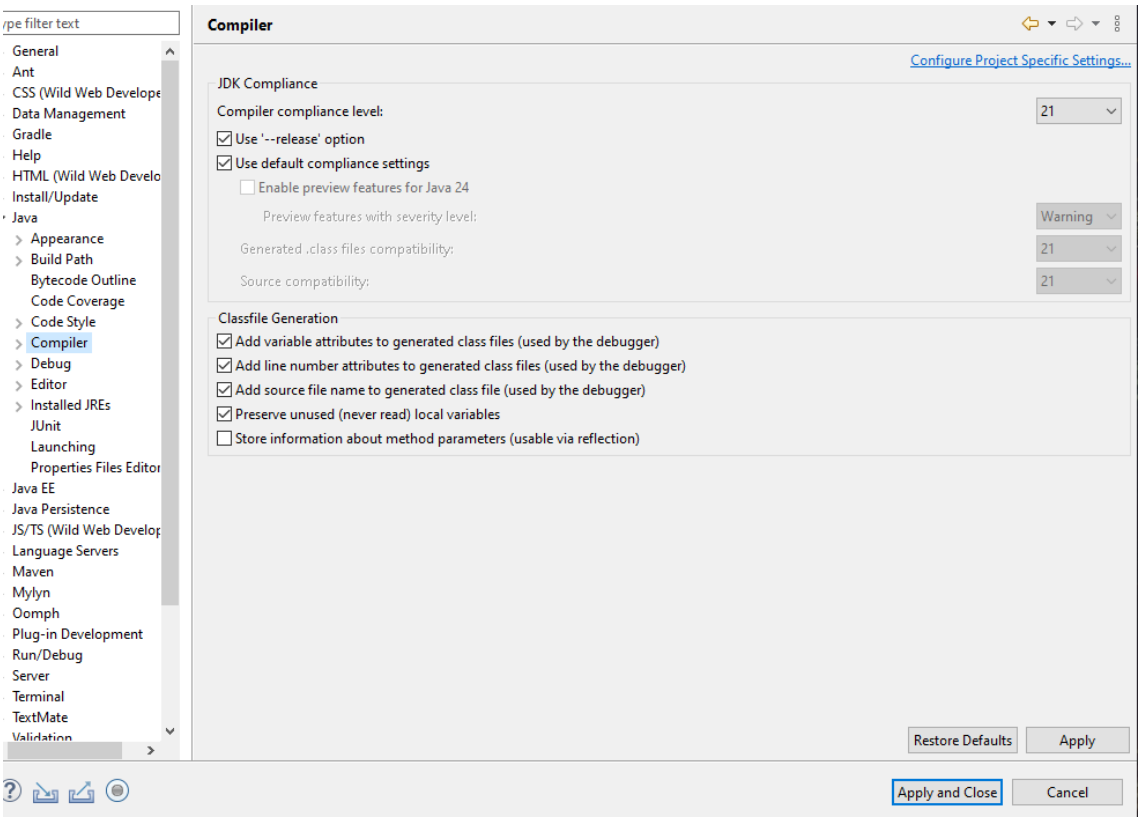
Ahora, para terminar esta parte, lo que vamos a hacer es desplegar lo que tenemos de aplicación por ahora en tomcat.



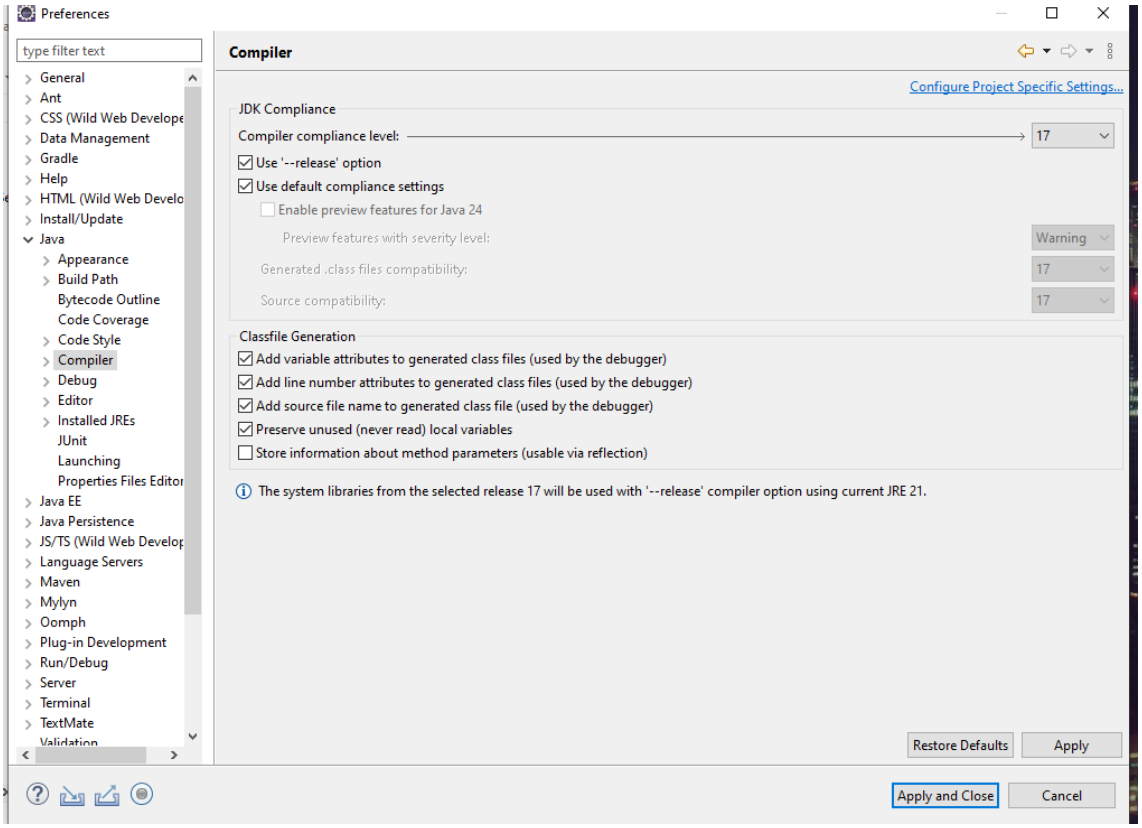
Abrimos el panel de control y arrancamos el servicio Tomcat, que es el que va a alojar nuestra aplicación.



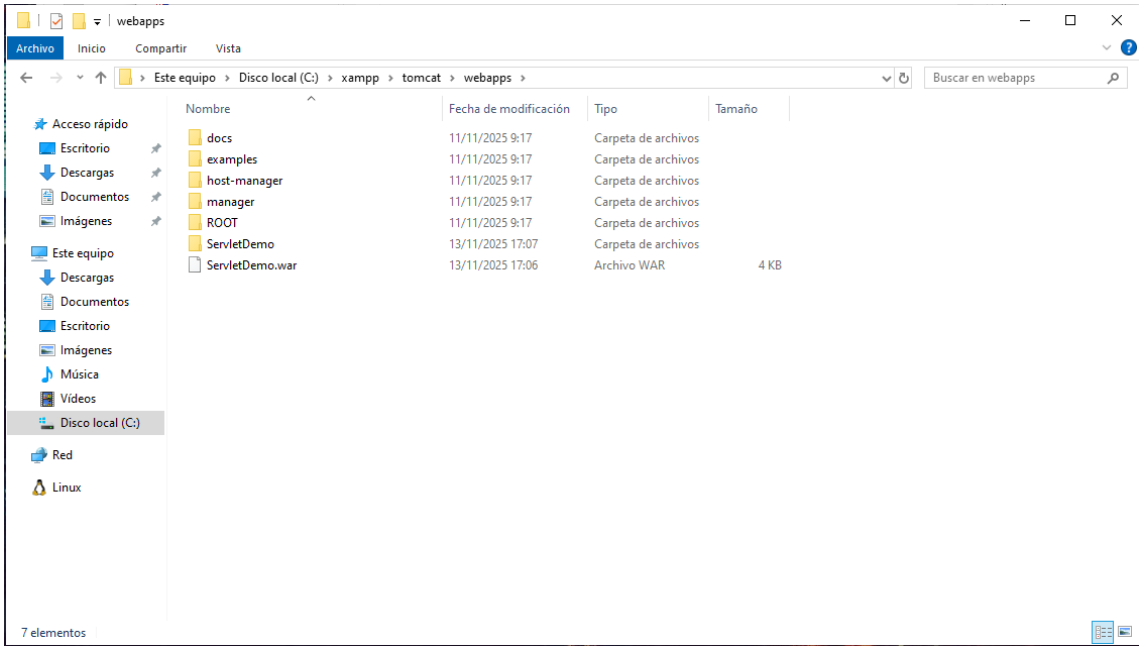
Nos aseguramos de que tenemos todo listo.



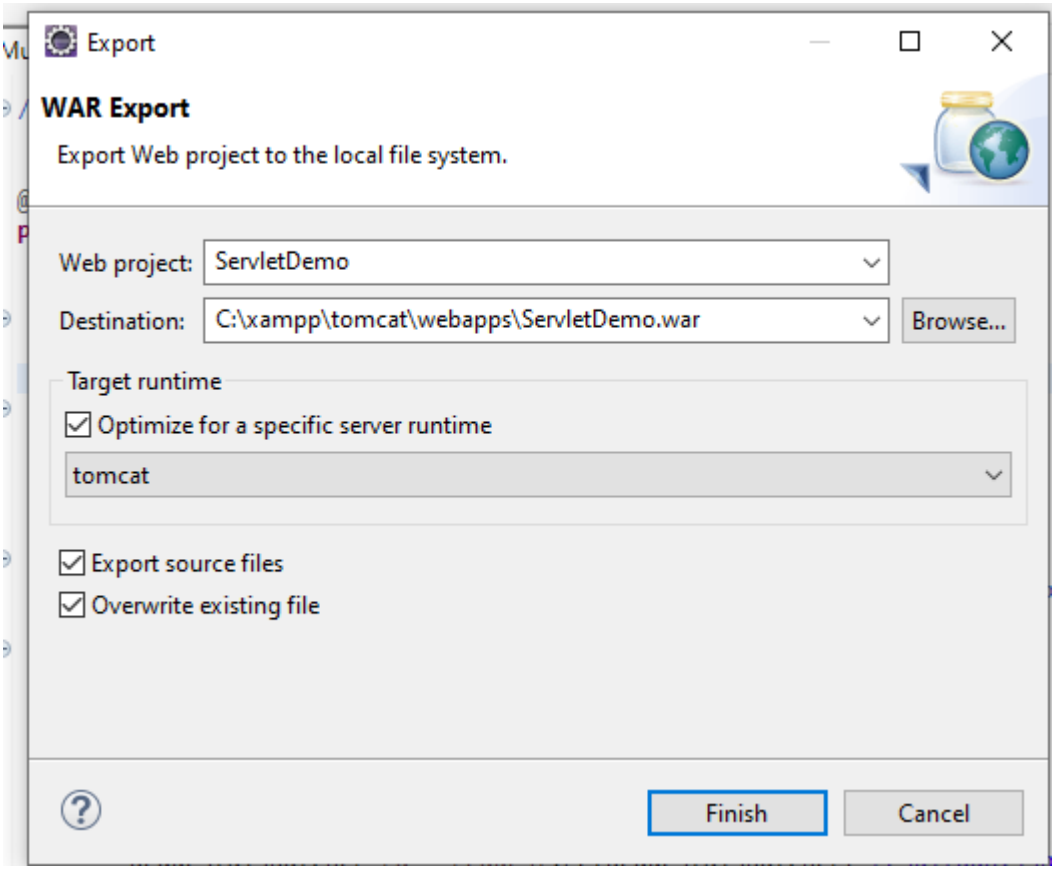
Antes de seguir debemos asegurarnos de que versión del JDK vamos a usar. Lo normal es usar la 17 para la versión de Tomcat que usamos. Así que en ese box a la derecha que vemos debo de cambiar 21 por 17. Ya me ha pasado otras veces que si no lo hago no va.



Lo dejamos ya cambiado a la versión 17, aplicamos y salimos.

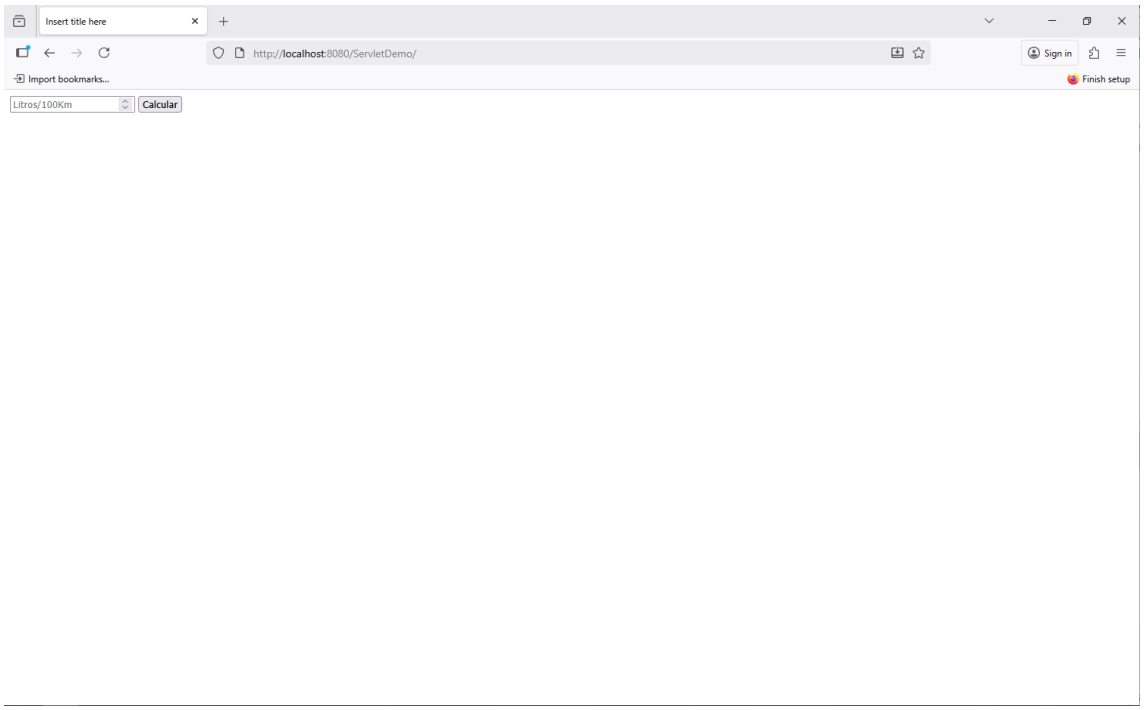


En esta captura vemos donde se alojan los ficheros .war en los que se compilan y comprimen este tipo de proyectos. Como vemos aquí, esto es del proceso anterior. Debo de eliminar los archivos anteriores para eliminar problemas.

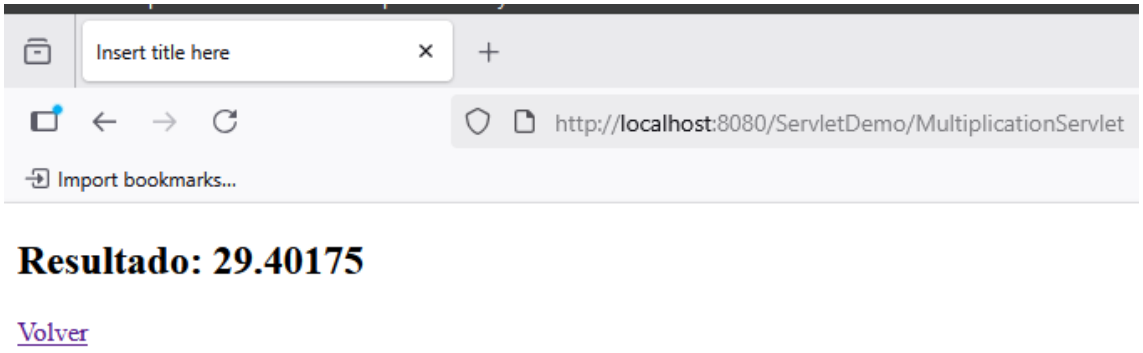


Con todas las comprobaciones hechas, botón derecho sobre nuestro proyecto, exportar como .WAR, le damos un nombre, importante que sea el mismo que el proyecto y que concuerde con lo puesto en los servlets, y le indicamos como destino la carpeta webapps dentro del directorio tomcat de Xampp.

Ahí tenemos el archivo. Ahora debemos de ir al navegador.



Si abrimos el navegador con la ruta localhost:8080/ServletDemo que es el nombre de nuestro proyecto vemos que aparece el input indicado, el botón para mandar los datos y el código html introducido. Vamos a hacer la prueba con 8.



Si insertamos 8 en el input y pulsamos calcular, nos aparece el resultado en el segundo archivo .jsp, que es el que recibe la respuesta. Ahora vamos con la segunda parte del ejercicio.

Para la segunda parte debo de añadir código al archivo respuesta .jsp.

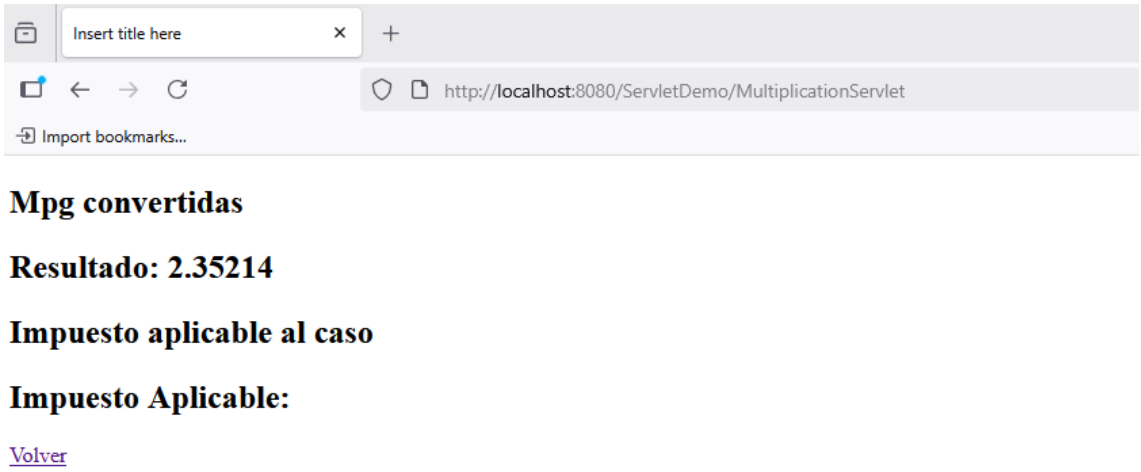
```
</head>
<body>
    <h2>Mpg convertidas</h2>
    <h2>Resultado: ${resultado}</h2>

    <h2>Impuesto aplicable al caso</h2>
    <h2>Impuesto Aplicable: $(impuesto)</h2>
    <a href="index.jsp">Volver</a>

</body>
</html>
```

Como vemos he añadido tres etiquetas h2. Dos de ellas identificativas, para que al añadir esta nueva respuesta sepamos cual corresponde a cual. Después me aseguro de incluir otra más en el que el archivo va a recibir el resultado de la variable impuesto, para que la muestre por pantalla. Ahora debo de volver a modificar el servlet para que incluya esta operación.

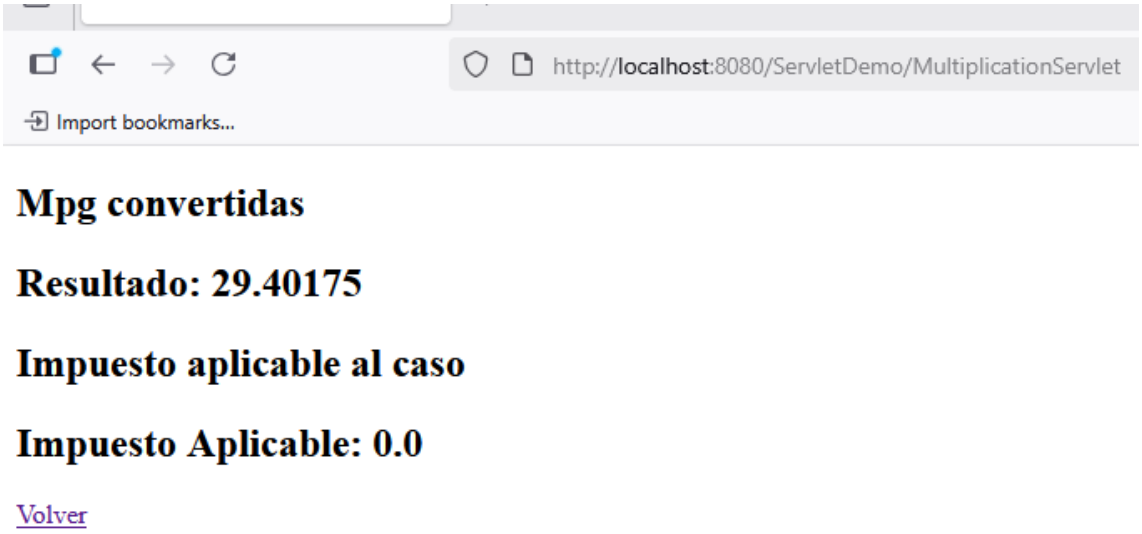
```
    protected void doPost(HttpServletRequest request, HttpServletResponse respons
            throws ServletException, IOException {

        double n1 = Double.parseDouble(request.getParameter("num1"));
        double resultado = 235.214 / n1;

        double impuesto = 0;

        if(n1 > 22.5) {
            impuesto = 0;
        }else if(n1 > 17.5 && n1 < 22.4) {
            impuesto = 2000;
        }else if(n1 > 12.5 && n1 < 17.4) {
            impuesto = 4000;
        }else if(n1 < 12.4) {
            impuesto = 7700;
        }

        request.setAttribute("resultado", resultado);
        RequestDispatcher rd = request.getRequestDispatcher("resultado.jsp");
        rd.forward(request, response);

        request.setAttribute("impuesto", impuesto);
        RequestDispatcher rd = request.getRequestDispatcher("resultado.jsp");
        rd.forward(request, response);
    }
}
```

Como vemos en la foto de la página anterior, el código ha crecido un poco. En este caso lo que hago es evaluar el resultado de la operación anterior. Miro su valor, dependiendo de cual sea la variable impuesto tendrá un valor u otro.

Ahora, no voy a volver a mostrar todo el proceso anterior, pero por si acaso compruebo la versión de JDK, elimino los archivos antiguos del Tomcat, exporto el war a webaps, e busco en localhost:8080/ServletDemo.

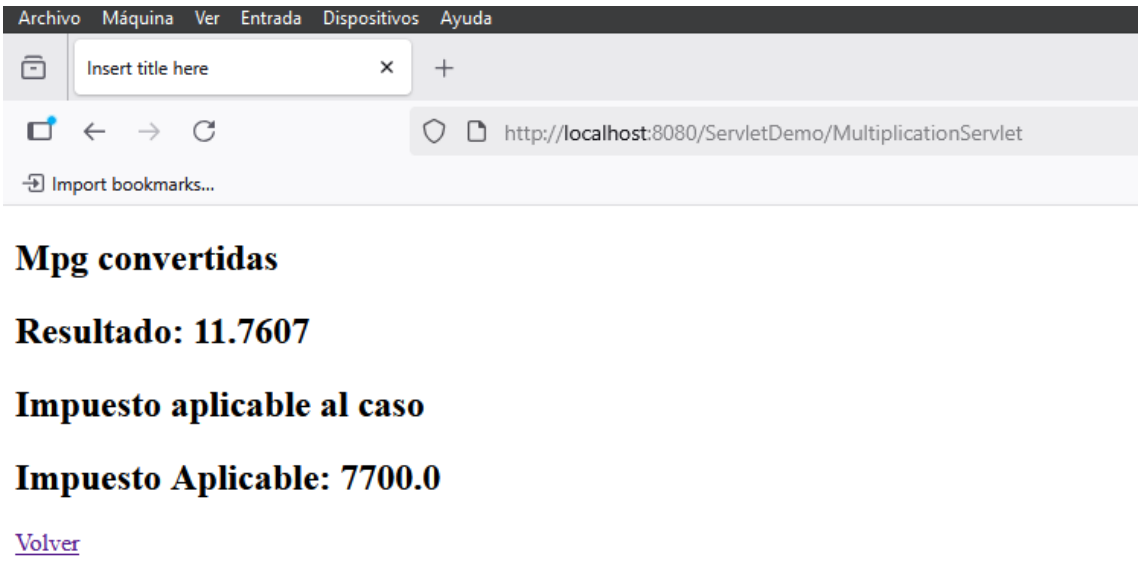


En principio la teoría funciona, pero algo no va bien del todo, ya que no nos muestra un resultado como tiene que ser. Aquí debería de mostrar un impuesto bastante grande. Por lo tanto hay que revisar el código.



Ahora sí, las malditas prisas. Si nos fijamos en la captura en la que muestro el código, vemos que hago un rd.forward dos veces, al archivo resultado.jsp, por lo que la aplicación se vuelve loca. Una vez lo cambio aparece un resultado que ya se puede comprobar. Vamos con los casos requeridos.

Este es el ejemplo de 4L. Como en el caso anterior el impuesto es 0.



Ya por último este es el ejemplo de 20L, en el que vemos que el impuesto es bastante distinto.

Pregunta 3. En este ejercicio se pide que conectes ambas actividades creadas en las preguntas 1 y 2. Tu aplicación web debe ser accesible desde tu sitio web.

Para completar este ejercicio, deberás proporcionar lo siguiente:

1. Cuantas capturas sean necesarias para mostrar la nueva funcionalidad, así como una breve descripción de los pasos que has seguido para lograrlo. (1 punto)