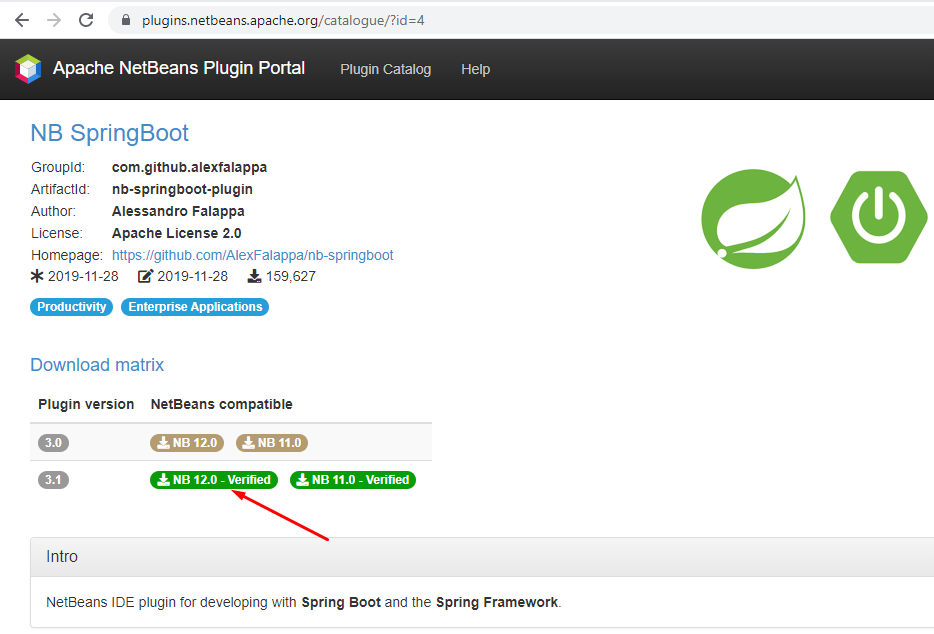
**Desarrollo Avanzado de Aplicaciones I**

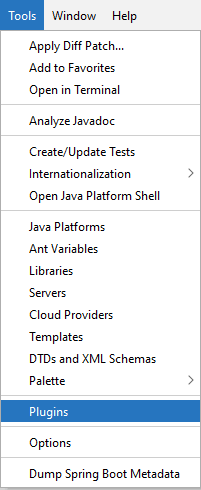
**Ejercicio 1**. Desarrollar una aplicación que muestre el mensaje “Hola Mundo” en un navegador utilizando Spring Boot.

1. Descargar el plugin de SpringBoot para NetBeans de la siguiente ruta: <https://plugins.netbeans.apache.org/catalogue/?id=4>

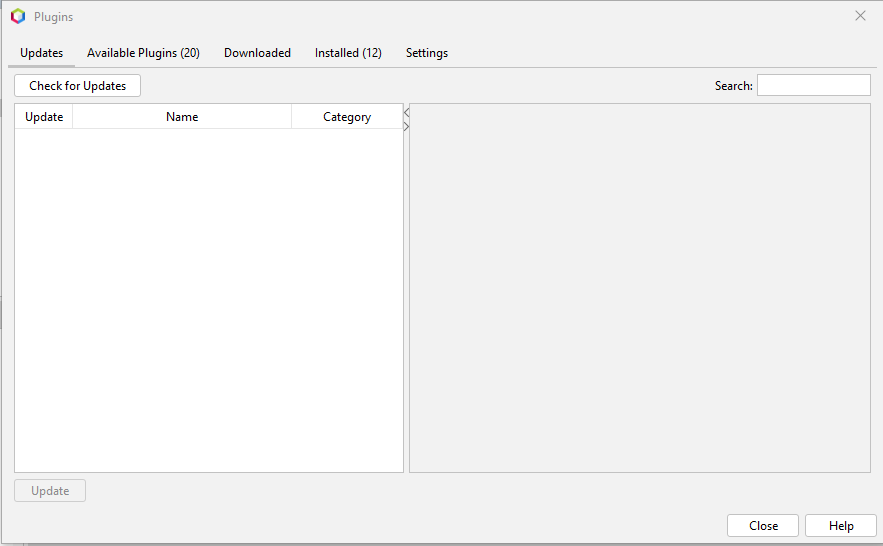


Descargar la versión 12.0.

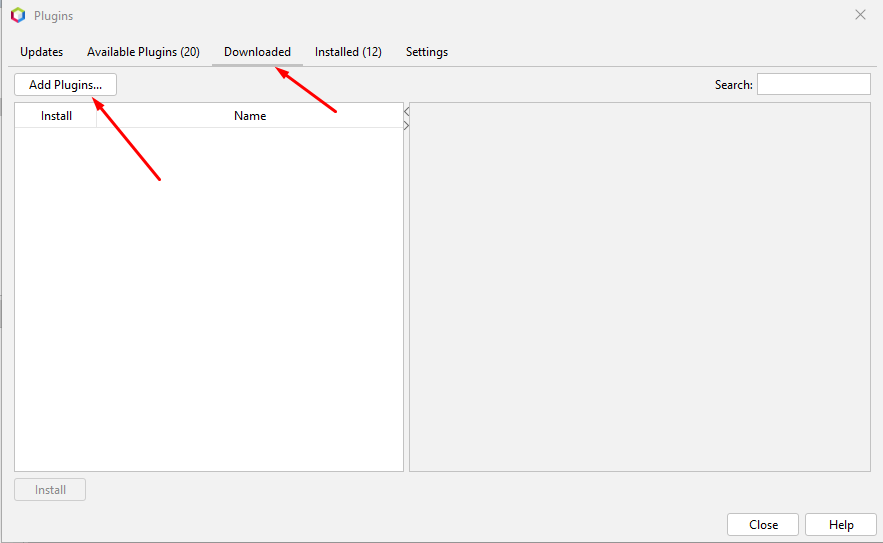
1. En NetBeans, hacer click en el menú “Tools” opción “Plugins”.



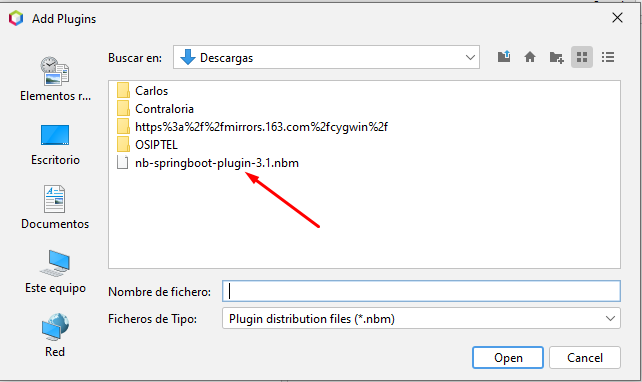
Le debe mostrar la siguiente pantalla:



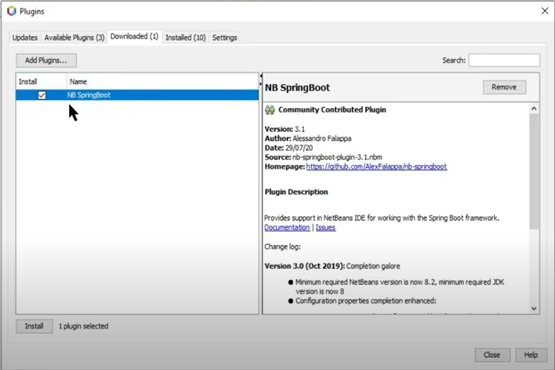
1. Hacer click en la opción “Downloaded” y hacer click en el botón “Add Plugins…”



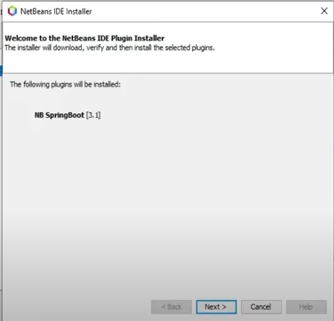
Seleccionar el archivo descargado:



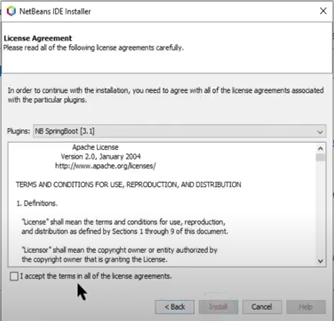
1. Verificar que esté seleccionada la casilla del Plugin y darle click al botón “Install”.



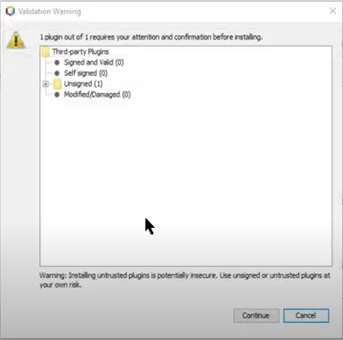
1. Hacer click en “Next”:



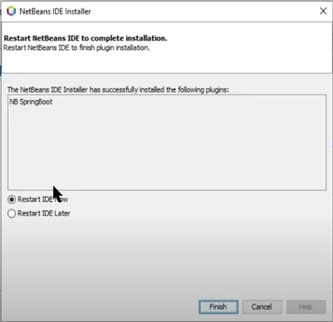
1. Seleccionar la casilla para aceptar los términos y condiciones y hacer click en “Install”:



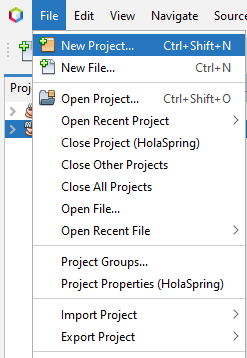
1. Si sale esta ventana de “Warning”, hacer click en el botón “Continue”:



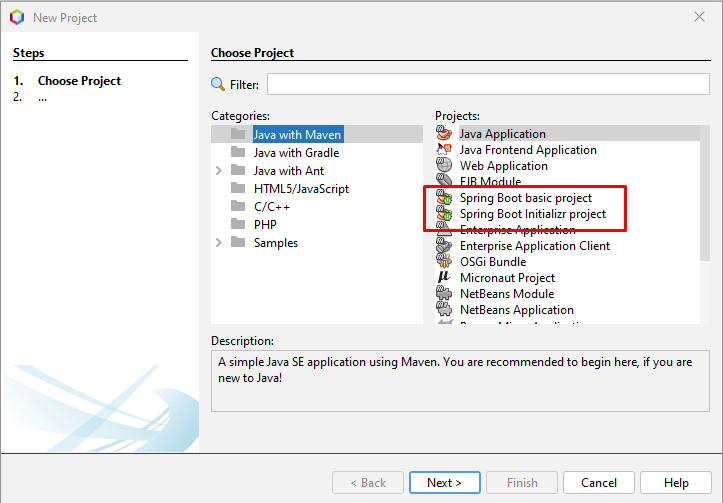
1. Verificar que este seleccionada la opción de “Reiniciar IDE ahora” y hacer click en el botón “Finish”:



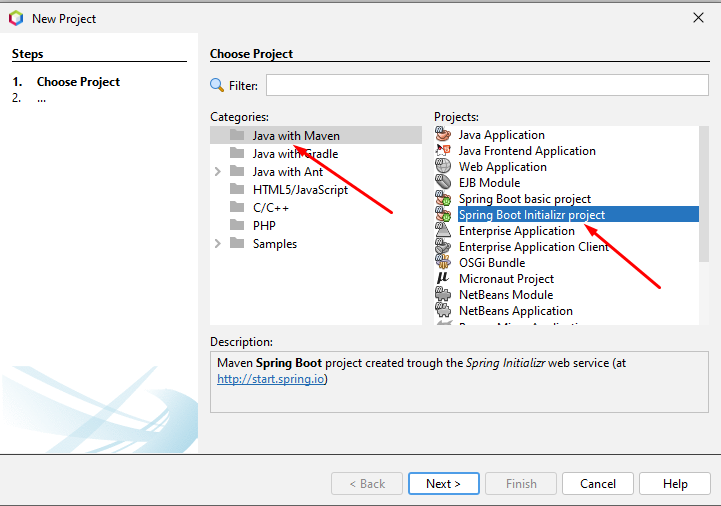
1. Una vez que reinició el NetBeans. Creamos un nuevo proyecto:



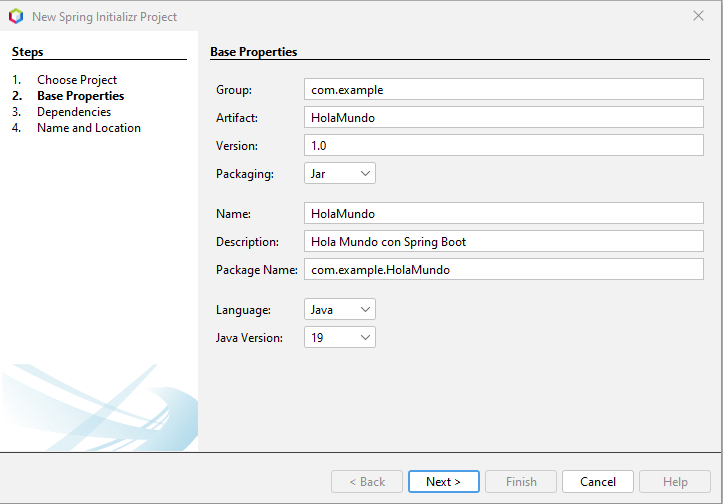
1. Como pueden ver se han agregado dos nuevos tipos de proyectos relacionados a Spring:



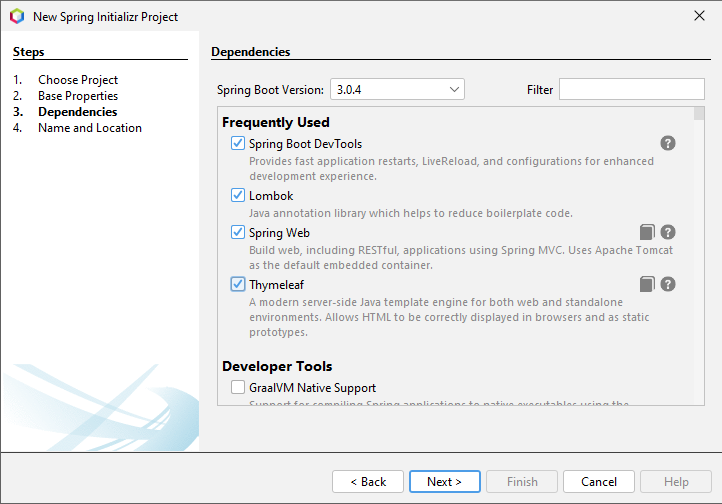
1. Seleccionar la categoría “Java with Maden” y en tipo de proyecto “Spring Boot Initializr project”. Luego hacer click en “Next”:



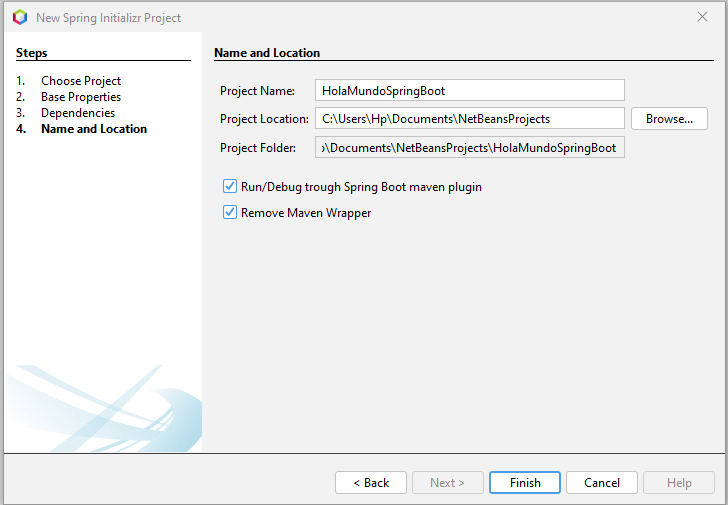
1. En la ventana de “Propiedades Básicas”, completar los valores de acuerdo a lo indicado en la imagen y luego hacer click en “Next”:



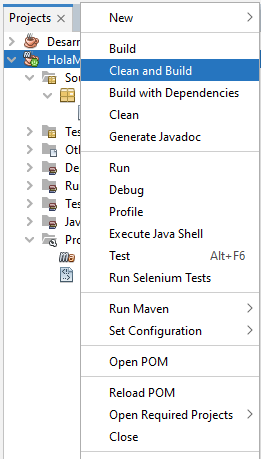
1. En la ventana de “Dependencias”, seleccionar las dependencias “Spring Boot DevTools”, “Lombok”, “SpringWeb” y “Thymeleaf” y luego hacer click en “Next”. Buscar las dependencias en la lista, no necesariamente les va a aparecer como en la imagen:



1. En la ventana “Nombre y Ubicación”, ponerle como nombre “HolaMundoSpringBoot” y luego hacer click en “Finish”: (Verificar que en el nombre del proyecto y en la ruta de la carpeta no haya espacios en blanco, ni caracteres raros, ni nombres muy largos de carpetas, para que no se produzcan errores posteriores. De ser así crear una nueva carpeta lo más cercana a la unidad “c:\” o seleccionar otra carpeta).



1. Una vez que haya creado el proyecto. Para comprobar que no tiene problemas y que no falta descargar ninguna dependencia, hacemos click con el botón derecho del mouse en el nombre del proyecto y en el menú emergente hacemos click en la opción “Clean and Build”:

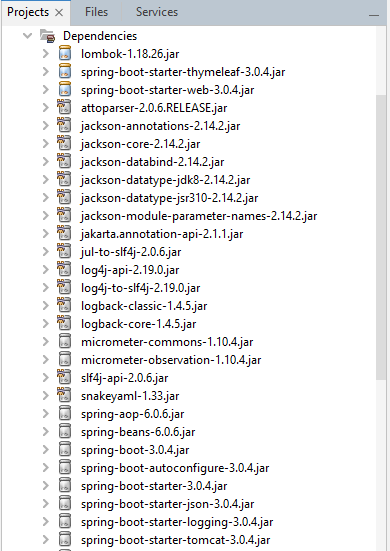


Nos debe de salir en la ventana “Output” el mensaje “BUILD SUCCES”:



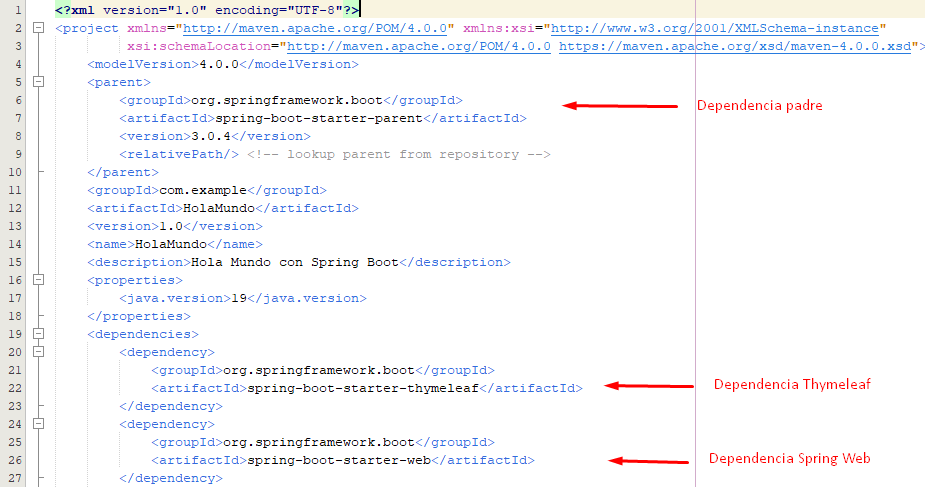
Si no les muestra el mensaje de compilación exitosa, puede ser porque tengan operando algún antivirus o firewall activo, el cual tendrían que deshabilitar para que realice las descargas e instalaciones necesarias.

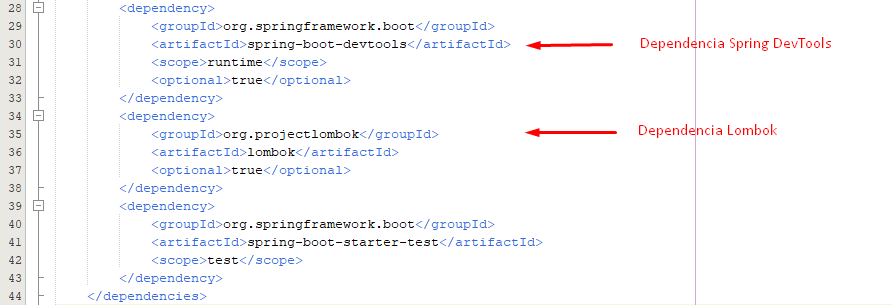
Verificar en la sección “Dependencias” si algún archivo tiene algún ícono de error:



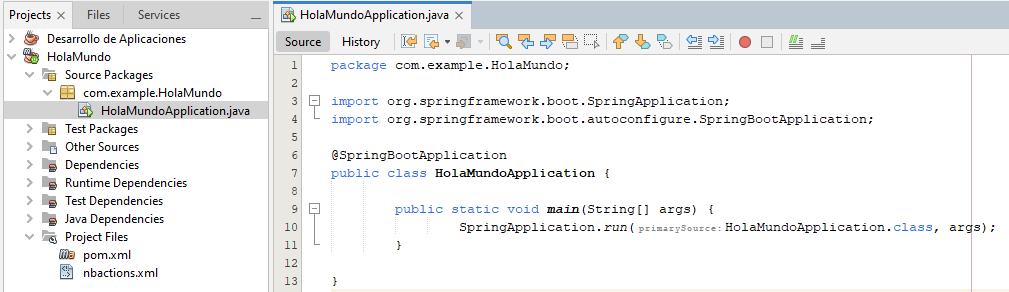
De ser así, en el explorador de archivos ir a la carpeta “C:\Users\XXX\.m2”, donde XXX es el nombre del usuario de la máquina donde están trabajando y eliminar la carpeta “repository” y volver a realizar la acción de “Clean and Build” del proyecto (Paso 15).

1. En el archivo “pom.xml” que se abre por defecto, hacer click con el botón derecho del mouse y seleccionar la opción “Format” para que nos ordene de mejor manera el archivo y guardamos los cambios. Luego podemos apreciar ahí las diferentes dependencias que hemos agregado:

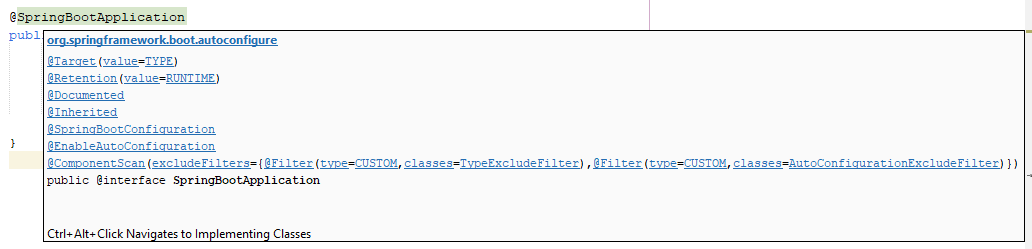




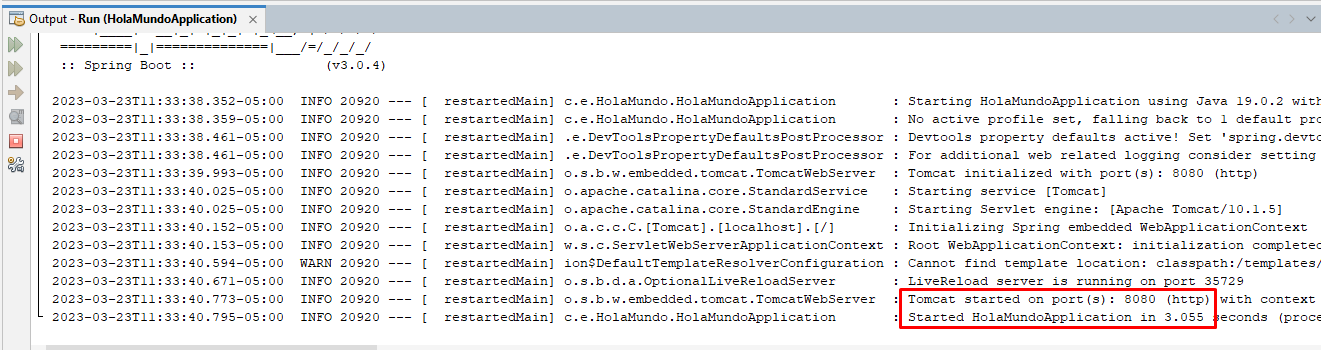
1. Abrimos el archivo “HolaMundoApplication.java” que fue generado de forma automática y que se encuentra en el paquete “com.example.HolaMundo”:



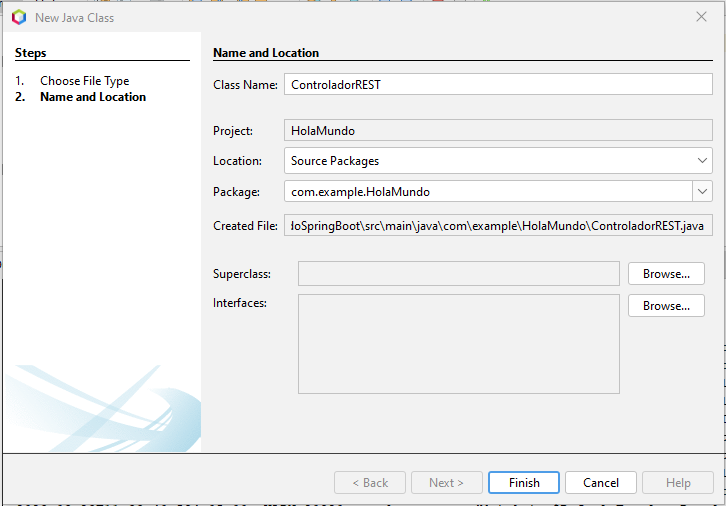
1. La clase “HolaMundoApplication” tiene la anotación “@SpringBootApplication” que indica que es una aplicación de Spring Boot. Si le das “click + ctrl” podrás ver el código origen de esa anotación:



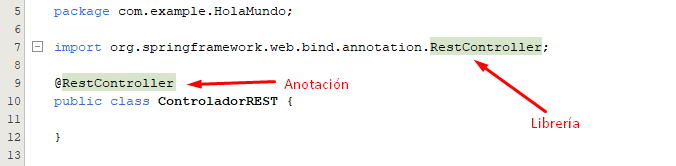
1. Con la sentencia “SpringApplication.run(HolaMundoApplication.class, args);” estamos indicando que debe ejecutar la clase “HolaMundoApplication” y le podemos hasta pasar argumentos con el parámetro “args”. Ejecutamos el archivo con “Shift + F6” y puede realizar algunas descargas adicionales y por último inicializa el server “Tomcat” y la aplicación:



1. Ahora vamos a modificar la aplicación para que usando un controlador de tipo REST pueda enviar información y ser vista a través de un navegador. Creamos una clase java de nombre “ControladorREST” dentro del mismo paquete “com.example.HolaMundo”:



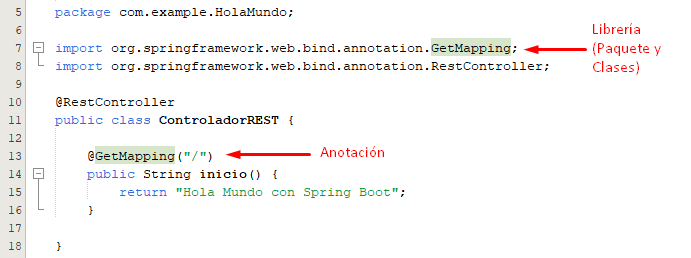
1. A la clase “ControladorREST” le agregamos la anotación “@RestController” e importamos la librería que se necesita. Si le das “click + ctrl” podrás ver el código origen de esa anotación:



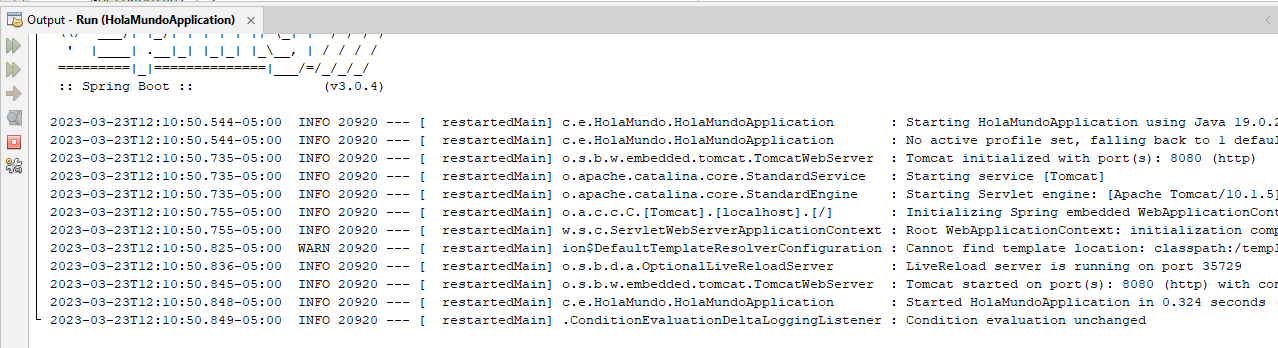
1. A la clase ” ControladorREST” le agregamos el siguiente método:



1. Al método inicio() le agregamos la anotación “@GetMapping(“/”)” e importamos la librería que se necesita. Esta anotación se utiliza para asignar solicitudes HTTP GET a métodos de controlador específicos. El “/” indica la ruta donde va a realizar la petición:

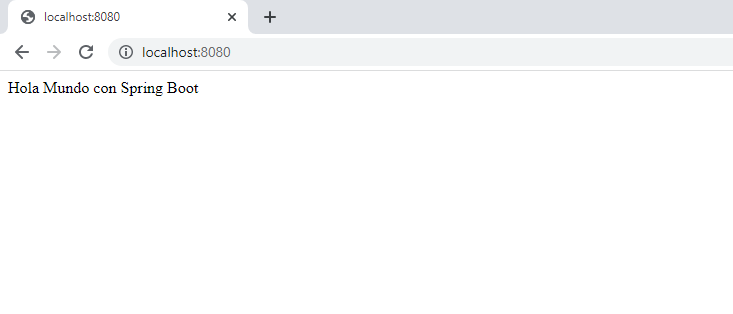


Grabamos los cambios realizados y automáticamente nuestra aplicación “HolaMundoApplication” se actualizará, lo cual se puede apreciar en la ventana “Output”:

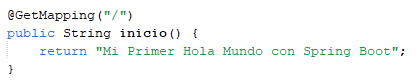


Esto se realiza de manera automática porque la aplicación tiene agregada la dependencia “Spring Boot DevTools” que hace que no necesitemos hacer despliegues.

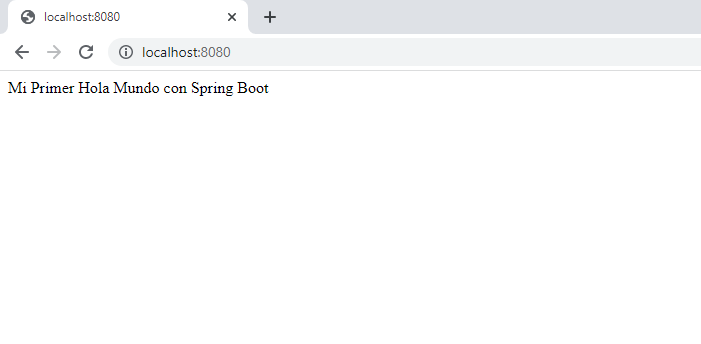
1. Abrimos un navegador y ejecutamos nuestra aplicación poniendo la dirección “localhost:8080”:



1. Podemos actualizar nuestro código, modificando el mensaje del método inicio() de la siguiente manera:

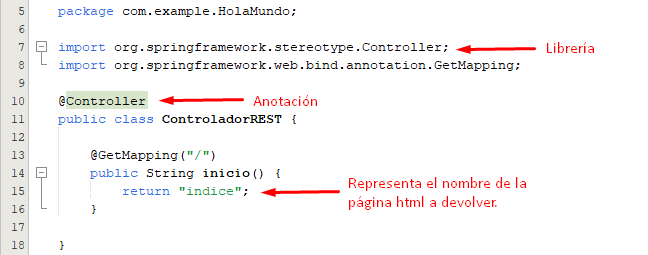


Grabamos y verificamos que se actualizó el server Tomcat en la ventana “Output” y actualizamos la página en el navegador:

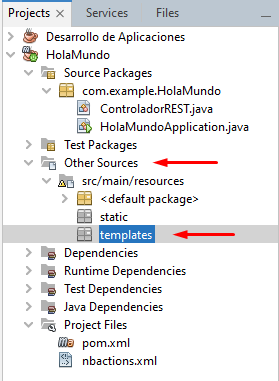


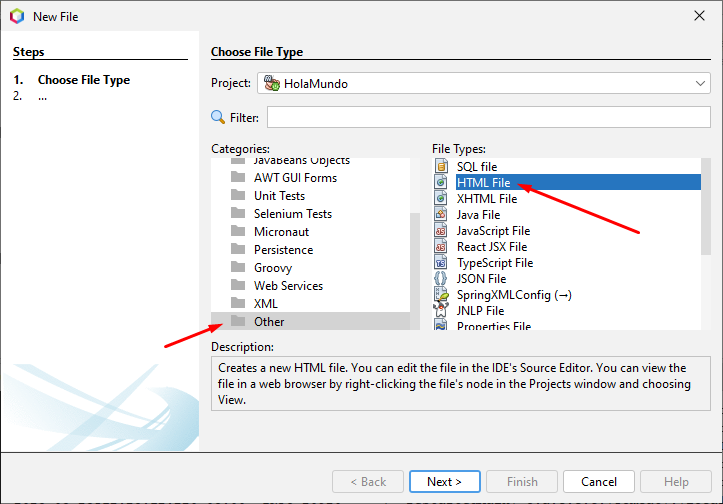
**Ejercicio 2**. Modificar la aplicación anterior para usar el controlador MVC y podamos ver una página html en el navegador, enviando información a la vista controlador y usando Thymeleaf en esa página html.

1. En el archivo “ControladorREST.java”, quitarle a la clase “ControladorREST” la anotación “@RestController” y eliminar el import de su librería. Luego agregarle la anotación “@Controller” y agregar el import de su librería. Adicionalmente, modificar el método inicio() para que ahora devuelva la palabra “indice”:

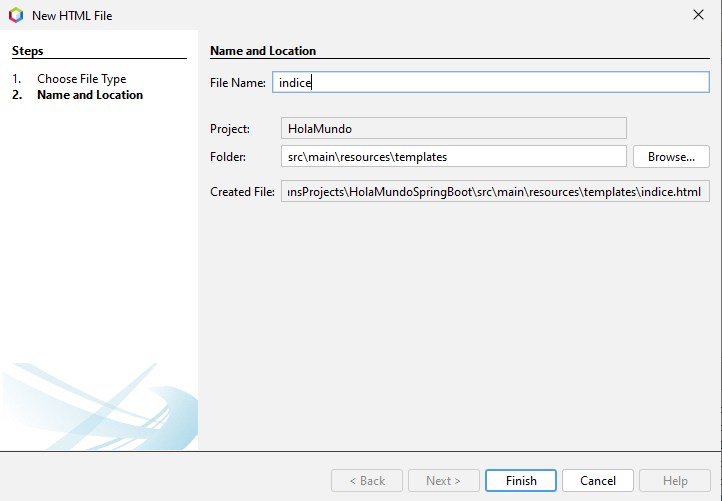


1. Dentro de la sección “Other Sources” se encuentra la carpeta “templates”. En esa carpeta creamos un archivo html:



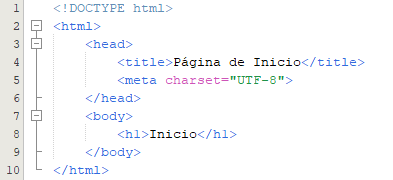


Le ponemos como nombre “indice” y le damos click en “Finish”:

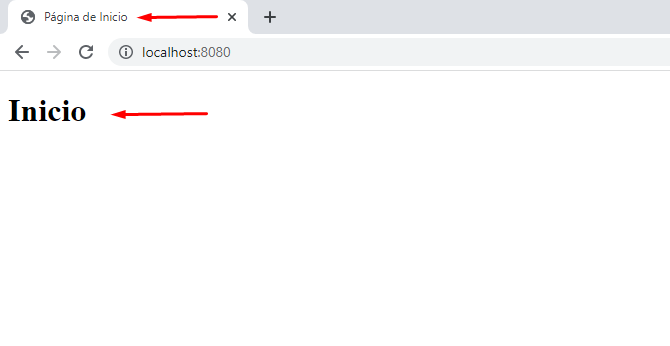




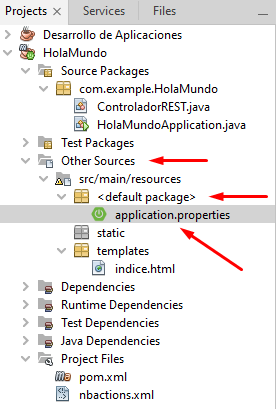
1. En el archivo “indice.html” realizar las siguientes acciones:
   1. Borrar los comentarios
   2. Cambiar el título por “Página de Inicio”: <title>Página de Inicio</title>
   3. Borrar el tag: <meta name="viewport" content="width=device-width, initial-scale=1.0">
   4. Borrar el tag: <div>TODO write content</div>
   5. En su lugar poner el tag: <h1>Inicio</h1>



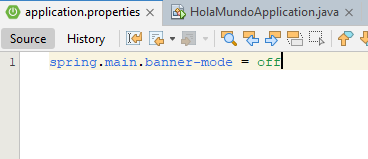
1. Abrimos un navegador y ejecutamos nuestra aplicación poniendo la dirección “localhost:8080”:



1. Dentro de la sección “Other Sources” se encuentra la carpeta “<default package>”. Dentro de ella se encuentra el archivo “application.properties”:

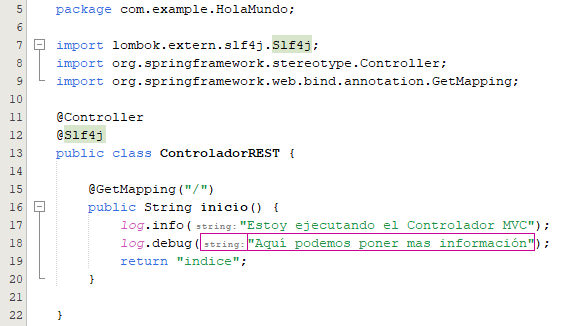


Este archivo se encuentra vacío y sirve para colocar ahí configuraciones como cambiar de puerto del server web (Tomcat), modificar la información que se muestra en la ventana “Output” (Info, Debug, Formato de Fechas, etc.). En este archivo colocamos la siguiente instrucción: spring.main.banner-mode = off, para que no salga el banner de spring en la ventana “Output”.

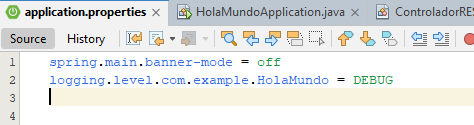


Adicionalmente, podemos usar la dependencia “Lombok” y este archivo de propiedades para gestionar la información que se muestra en la ventana “Output”.

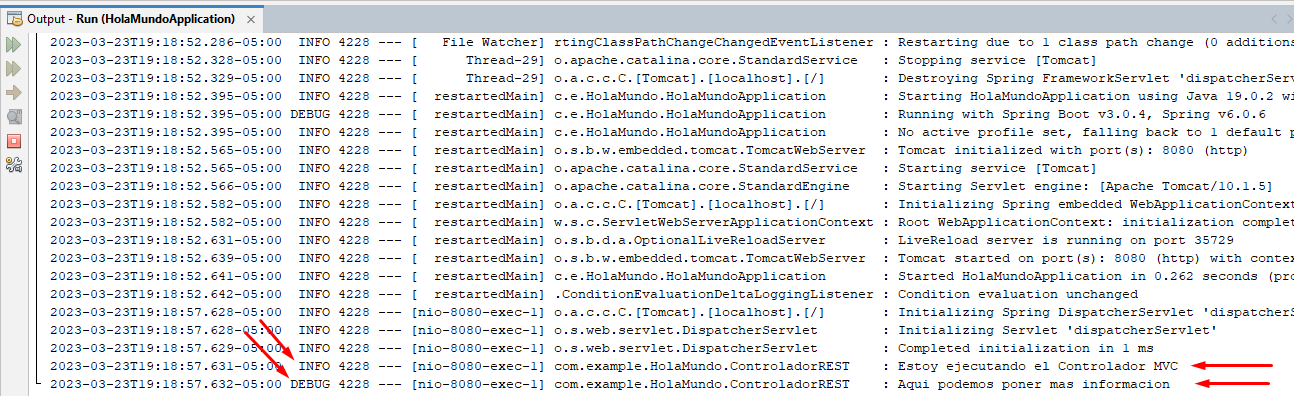
1. En el archivo “ControladorREST.java” a la clase “ControladorREST” le agregamos la anotación “@Slf4j” e importamos su librería. Adicionalmente ponemos las siguientes líneas de código en el método inicio():
   1. log.info("Estoy ejecutando el Controlador MVC");
   2. log.debug("Aquí podemos poner mas información");



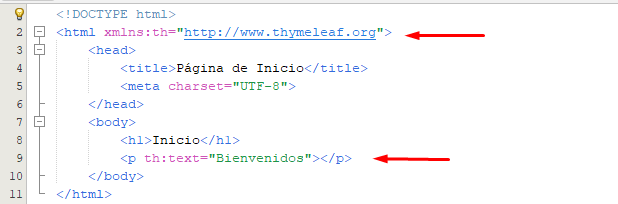
Luego agregamos la siguiente instrucción en el archivo “application.properties”: logging.level.com.example.HolaMundo = DEBUG



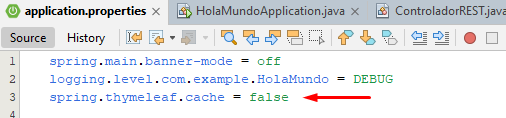
Ejecutamos la aplicación, refrescando en el navegador y visualizamos en la ventana “Output” las líneas que hemos escrito:



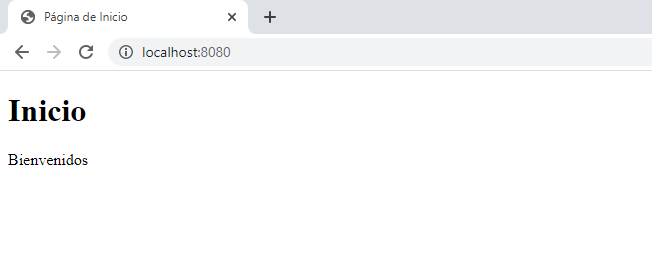
1. Agregar al archivo “indice.html” la referencia al espacio de nombres de “Thymeleaf” y agregarle un tag de párrafo usando Thymeleaf. Se debe deshabilitar el chequeo de errores html para el archivo:



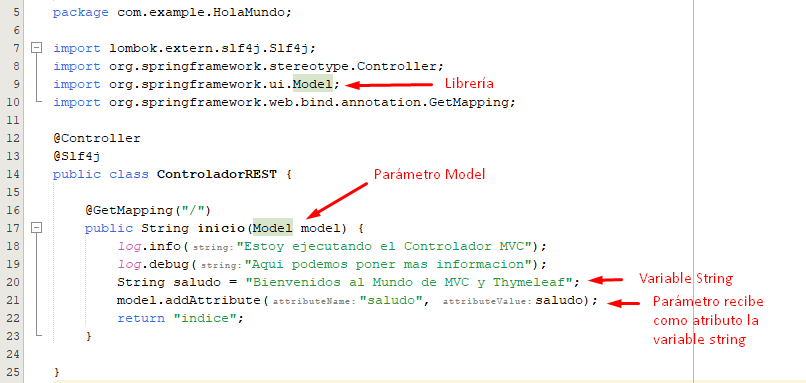
1. Agregamos la siguiente instrucción en el archivo “application.properties”: spring.thymeleaf.cache = false, para que no haya demoras en las actualizaciones.



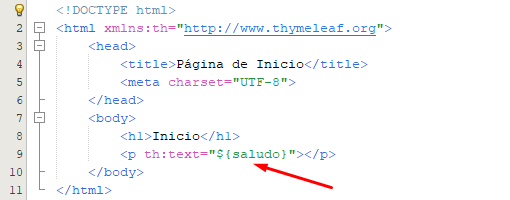
1. Ejecutamos la aplicación, refrescando la página en el navegador:



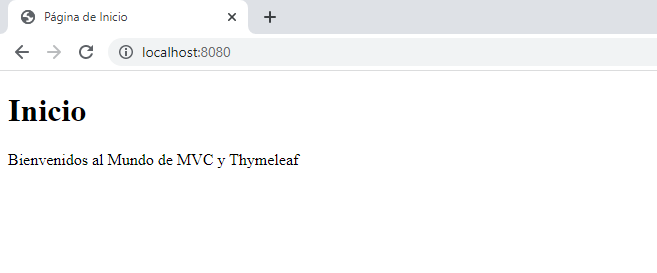
1. En el archivo “ControladorREST.java”, al método inicio() le agregamos un parámetro de tipo “Model” e importamos su librería. Adicionalmente, usamos ese parámetro para agregar un atributo usando una variable de tipo String:



1. En el archivo “indice.html” modificamos el tag del parrafo para usar el atributo de Model, es decir la variable String “saludo”:

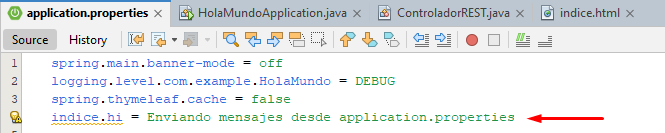


1. Ejecutamos la aplicación, refrescando la página en el navegador:

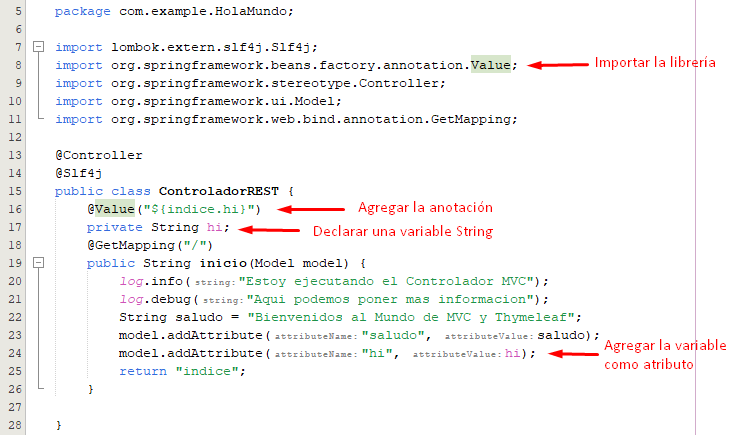


1. Agregar en el archivo “application.properties” la siguiente instrucción:

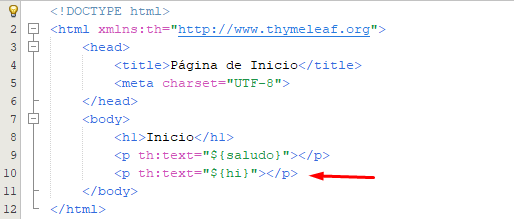
indice.hi = Enviando mensajes desde application.properties



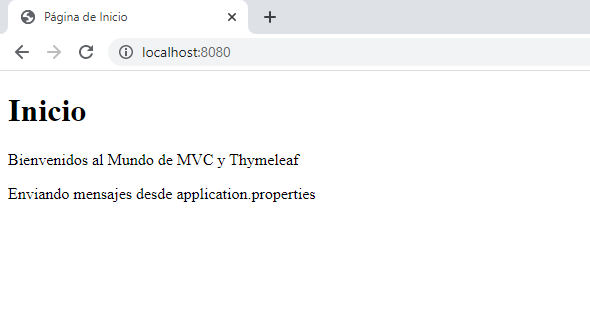
1. En el archivo “ControladorREST.java”, a la clase “ControladorREST” agregarle una variable privada de tipo String con nombre “hi”. A esta variable agregarle la anotación “@Value(“${indice.hi}”)” e importar su librería. En el método inicio() al parámetro “model” le agregamos un atributo usando la variable de tipo String “hi”:



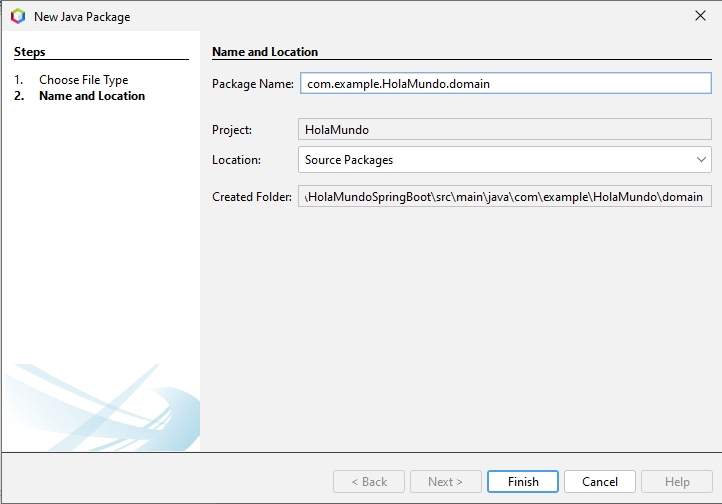
1. En el archivo “indice.html” le agregamos otro tag de parrafo para usar el otro atributo de Model, la variable String “hi”:



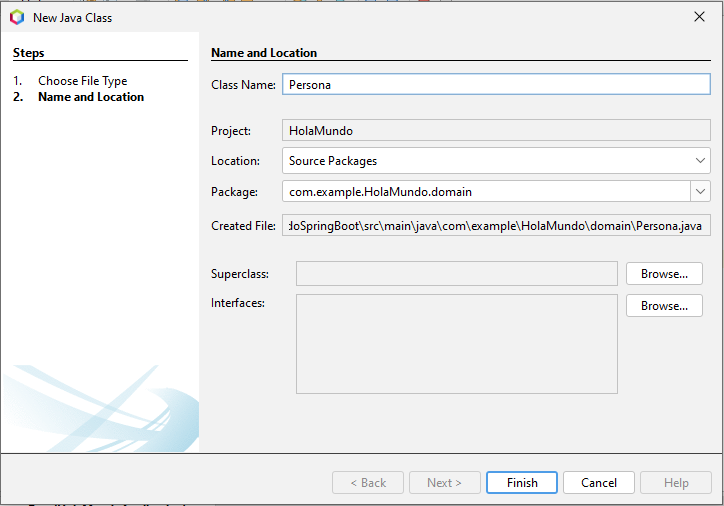
1. Ejecutamos la aplicación, refrescando la página en el navegador:



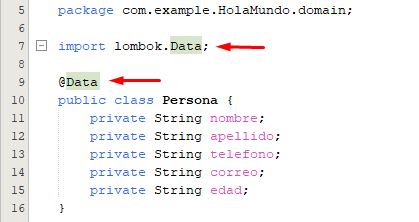
1. En “Source Packages” creamos un nuevo paquete con nombre “com.example.HolaMundo.domain”:



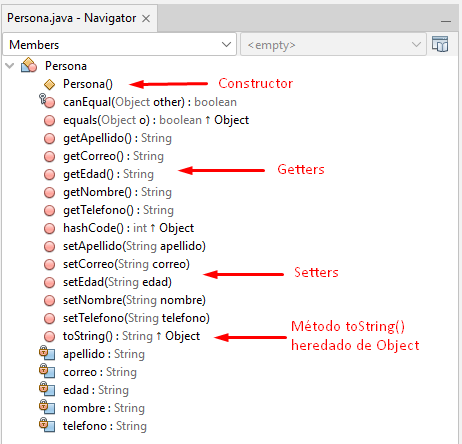
1. En el nuevo paquete creamos una nueva clase java con nombre “Persona”:



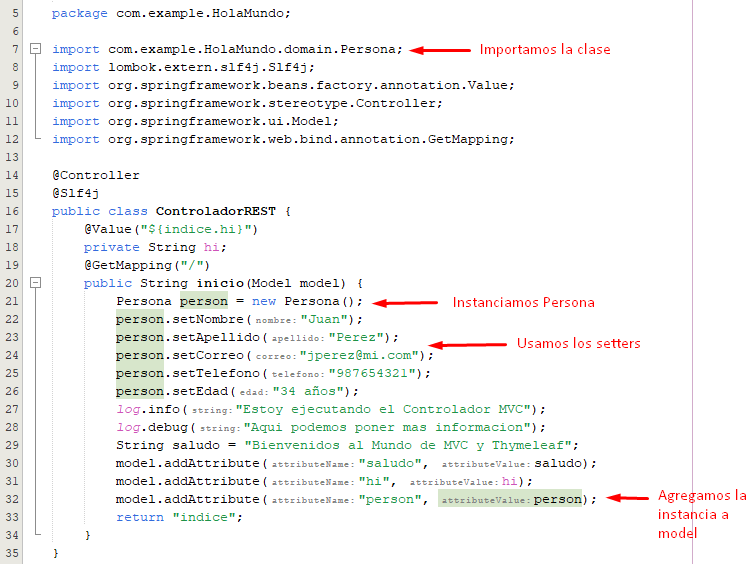
1. A la clase “Persona” agregarle los atributos privados “nombre”, “apellido”, “telefono”, “correo” y “edad”, todos ellos de tipo String. También agregarle la anotación “@Data” e importar su librería de Lombok.



1. En la ventana “Navigator” podemos apreciar todo lo que le ha agregado Lombok a la clase “Persona”:



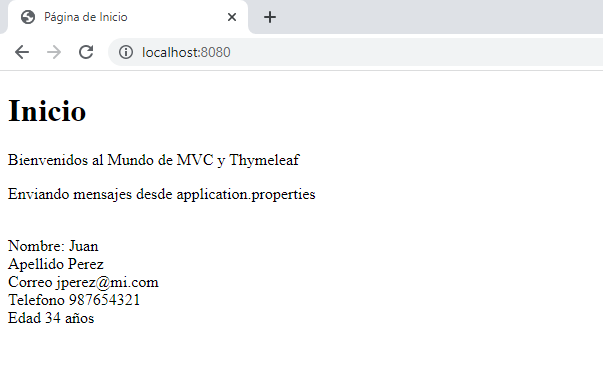
1. En el archivo “ControladorREST.java”, en el método inicio() creamos una instancia de la clase “Persona” y le ponemos como nombre “person” (no olvidar importar la clase porque está en otro paquete), luego le asignamos valores a sus atributos a través de sus métodos de acceso setters y por último al parámetro “model” le agregamos un atributo usando la instancia “person”:



1. En el archivo “indice.html” le agregamos los tags necesarios para mostrar la información de “person”:



1. Ejecutamos la aplicación, refrescando la página en el navegador:



**Ejercicio 3**. Desarrollar una aplicación que muestre la aplicación del concepto de Inyección de Dependencia.



