¿Qué es Spring boot?

Spring Boot es un framework que busca reducir la longitud del código fuente de las aplicaciones web así como simplificar su desarrollo.

Cualquier aplicación web se divide en las siguientes tareas:

1. Crear un proyecto Maven y añadir las dependencias y plugins al archivo pom.xml.
2. Desarrollar la aplicación.
3. Desplegar la aplicación en un servidor.

Spring boot nos permite centrarnos exclusivamente en la tarea 2, es decir, en el desarrollo de nuestra aplicación. Las tareas 1 y 3 son realizadas por Spring boot.

Spring boot nos permite crear la aplicación por medio de un asistente intuitivo. Solo tenemos que escoger el paquete donde queremos ubicar nuestras clases, elegir un nombre para el proyecto y seleccionar las dependencias dentro de un listado que se nos ofrece.

Los Starters de Spring boot

Para las dependencias Maven del proyecto, ya no hay que elegirlas una por una.
Un Spring Starter agrupa de una vez 10 o 20 dependencias.

Despliegue de la aplicación

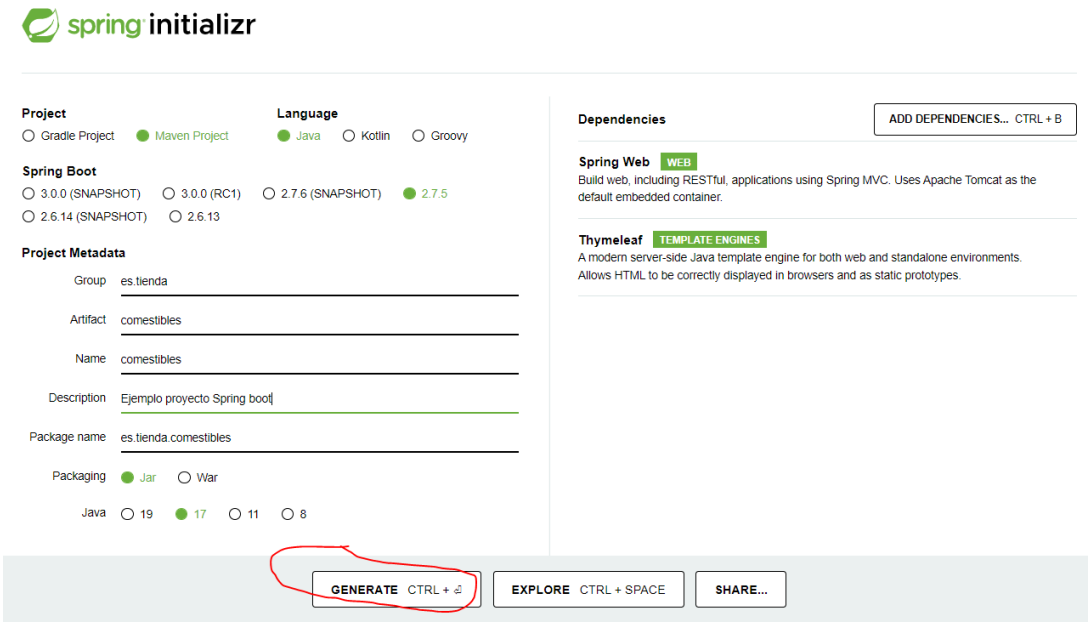
Spring boot lleva integrado Tomcat. Basta con crear la aplicación y al ejecutarla levanta un servidor Tomcat en el que la desplegará.

¿Cómo crear un proyecto Spring boot?

Spring boot lleva integrado Tomcat. Basta con crear la aplicación y al ejecutarla levanta un servidor Tomcat en el que la desplegará.

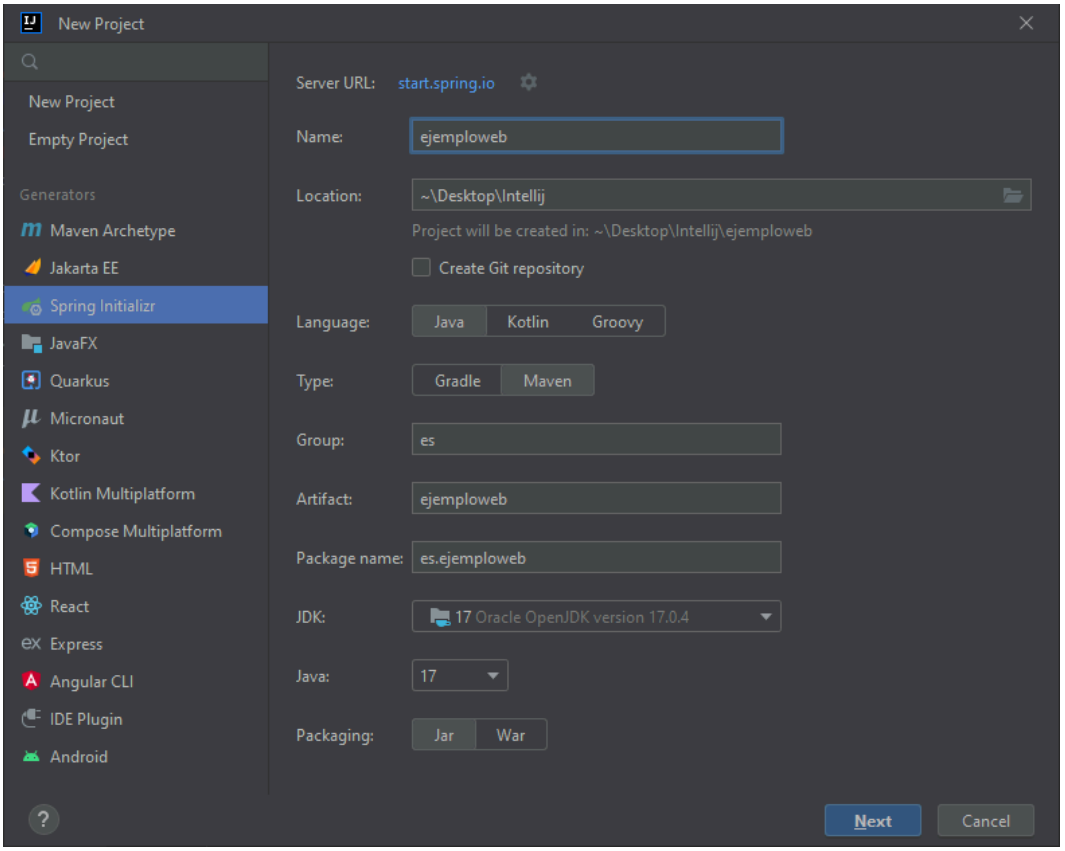
Es posible crear un proyecto Spring boot en la web Spring Initializr https://start.spring.io/. Tendremos que indicar las propiedades del proyecto maven (group id, artifact, etc.), la versión de java, tipo de empaquetado y por último seleccionamos las dependencias en un listado.

Una vez configurado el proyecto, pulsamos el botón GENERATE y obtendremos un archivo .zip que podremos importar desde la mayoría de los IDEs.

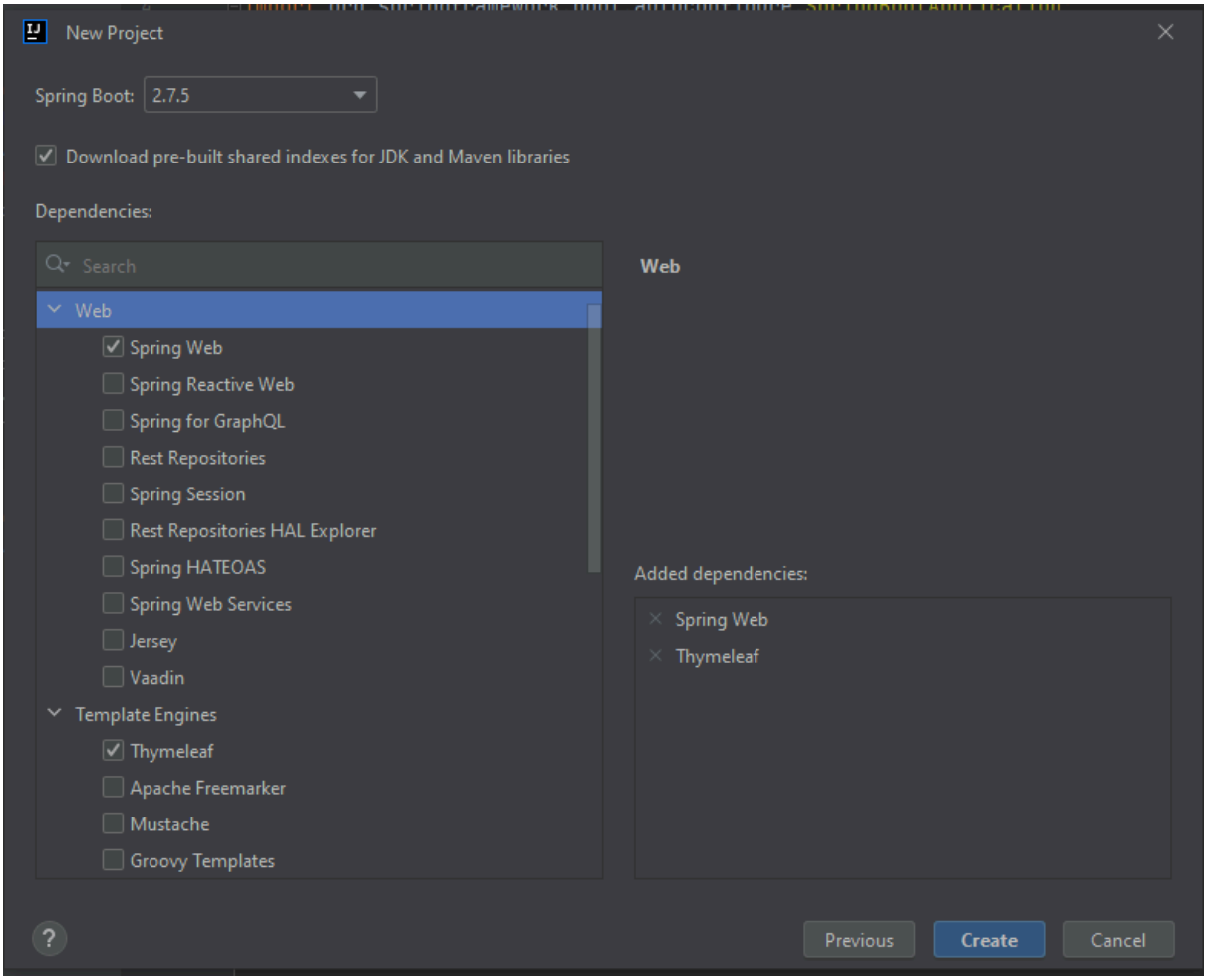


Con Intellij IDEA Ultimate

Si disponemos de la versión Ultimate de Intellij IDEA, podemos crear un nuevo proyecto de tipo Spring Initializr.
En el primer paso, escogemos el nombre del proyecto, localización, lenguaje, groupid, artifactid, etc.



En el segundo paso comenzaremos a escoger las dependencias. Recuerda que lo que escogemos son uno o varios Starters de Spring boot, cada uno de los cuales incluye varias dependencias.



Por el momento vamos a hacer algo sencillo, así que escogeremos un proyecto Spring Web que incluye la tecnología Spring MVC y la librería Thymeleaf para las vistas.

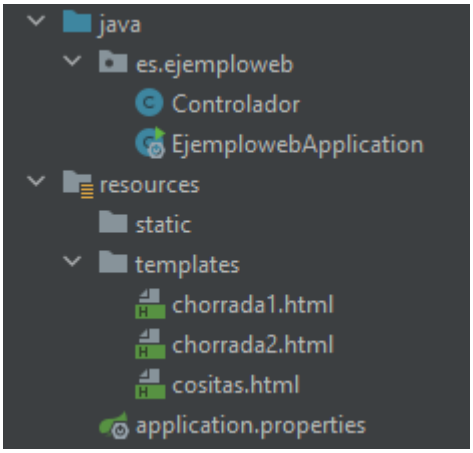
Terminaremos pulsando el botón Create.

Spring boot ya nos ha montado el proyecto.

Programando nuestro primer proyecto Spring MVC con Spring boot

En este punto ya tenemos creado el proyecto, ahora podemos ir manos a la obra para realizar la lógica del flujo web de la aplicación.

Vamos a comenzar por crear un controlador y tres vistas tal y cómo se ve en la imagen.



Aprovecharemos nuestro controlador para recordar las distintas formas de gestionar el modelo de datos.

```
package es.ejemploweb;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;

@Controller
public class Controlador {

    // Controlador que retorna un ModelAndView.
    @RequestMapping("/")
    public ModelAndView peticion1(HttpServletRequest request){
        ModelAndView mv = new ModelAndView();
        HttpSession sesion = request.getSession();
        mv.addObject("sms1", "Hola desde Spring boot");
        mv.addObject("sms2", "Id de sesión: "+sesion.getId());
        mv.setViewName("cositas");
        return mv;
    }

    // Controlador que retorna un String.
    @RequestMapping("/chorrada1")
    public String peticion2(ModelMap model) {
        model.addAttribute("sms1", "Aquí desde la primera chorrada");
        model.addAttribute("sms2", "Finalizada nuestra primera chorrada");
        return "chorrada1";
    }
```

```
    // Utilizando un método anotado con @ModelAttribute
    @RequestMapping("/chorrada2")
    public String peticion3(@ModelAttribute("mensajes") String[] mensajes){
        return "chorrada2";
    }
    @ModelAttribute("mensajes")
    public String[] getMensajes() {
        String[] textos = new String[3];
        textos[0] = "Hola Mundo";
        textos[1] = "Hola Amigos";
        textos[2] = "Hola Caracola";
        return textos;
    }
}
```

cositas.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1 th:text="${sms1}"></h1>
    <h1 th:text="${sms2}"></h1>
    <p><a href="chorrada1">Primera chorrada</a></p>
    <p><a href="chorrada2">Segunda chorrada</a></p>
</body>
</html>
```

chorrada1.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>Vista chorrada1</h1>
<h1 th:text=${sms1}></h1>
<h1 th:text=${sms2}></h1>
</body>
</html>
```

chorrada2.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>Vista chorrada2</h1>
<h1 th:text=${mensajes[0]}></h1>
<h1 th:text=${mensajes[1]}></h1>
<h1 th:text=${mensajes[2]}></h1>
</body>
</html>
```