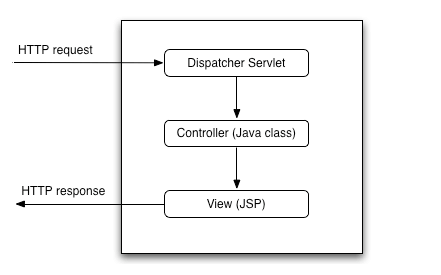
Servlet in Spring Boot

In Spring MVC all incoming requests go through a single servlet is called **Dispatcher Servlet** (**front controller**).

A single servlet receives all the requests and transfers them to all other components of the application to decide which will handle it.



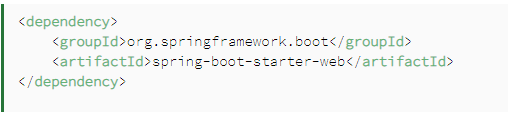
In a traditional Spring web application, this servlet is defined in the web.xml file.

Let's see a DispatcherServlet example configuration in the web.xml file:



Spring Boot provides the spring-boot-starter-web library for developing web applications using Spring MVC. One of the main features of Spring Boot is **autoconfiguration**. The Spring Boot autoconfiguration registers and configures the DispatcherServlet automatically. => we don’t need to do it manually.

Add the spring-boot-starter-web Maven dependency to our pom.xml file



**Compare:**

Basically, in a traditional web application, we use servlet as a control on the server side depending on the url that the user activates.

In spring boot, all servlets are auto-config and on the client side, the user calls the end point and it invokes the corresponding servlet to handle the request.

**How to register a servlet:**

Normally in web application without servlet,

We define a servlet by extending the HttpServlet class:



and we would configure our CustomServlet in the web.xml file:



In a Spring Boot application, the servlet is registered either as a Spring **@Bean** or by scanning the **@WebServlet** annotated classes.

With Spring **@Bean**, we can use the ServletRegistrationBean class to register the servlet.

