

The DispatcherServlet is an actual Servlet (it inherits from the HttpServlet base class), and as such is declared in the web.xml of your web application. You need to map requests that you want the DispatcherServlet to handle, by using a URL mapping in the same web.xml file. This is standard Java EE Servlet configuration; the following example shows such a DispatcherServlet declaration and mapping:

<web-app>

<servlet>

<servlet-name>example</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>example</servlet-name>

<url-pattern>/example/\*</url-pattern>

</servlet-mapping>

</web-app>

**Do we need web.xml in web application always? Ans:No**

In a Servlet 3.0+ environment, you also have the option of configuring the Servlet container programmatically. Below is the code based equivalent of the above web.xml example:

**public** **class** MyWebApplicationInitializer **implements** WebApplicationInitializer {

*@Override*

**public** **void** onStartup(ServletContext container) {

ServletRegistration.Dynamic registration = container.addServlet("dispatcher", **new** DispatcherServlet());

registration.setLoadOnStartup(1);

registration.addMapping("/example/\*");

}

}

Upon initialization of a DispatcherServlet, Spring MVC looks for a file named [servlet-name]-servlet.xml in the WEB-INF directory of your web application and creates the beans defined there, overriding the definitions of any beans defined with the same name in the global scope.