CS544 EA

Applications

SH Web Apps: Spring in a Web Container

Web Container

- The web-container will be the main application
  - Starting the Spring container when it starts

Including Container Startup

- Web Containers can register listeners
  - Allowing you to listen to container events
  - Spring provides a ContextLoaderListener that we can register in the web container

Web.xml

- The **`<context-param>`** tag can store data visible to the whole web app (all servlets etc)
- The **`<listener>`** tag registers a listener

```xml
<web-app … version="3.0">
  ...
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/springconfig.xml</param-value>
  </context-param>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  ...
</web-app>
```

Param to specify where to find Spring config file

Will start Spring when the app starts

Without web.xml

```java
package application03;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration;

import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.ContextLoaderListener;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;

public class MyWebAppInitializer implements WebApplicationInitializer {

	@Override
	public void onStartup(ServletContext container) throws ServletException {
		// Create the Spring 'root' application context
		AnnotationConfigWebApplicationContext rootContext =
				new AnnotationConfigWebApplicationContext();
		rootContext.register(Config.class);

		// Manage the lifecycle of the root application context
		container.addListener(new ContextLoaderListener(rootContext));

		ServletRegistration.Dynamic hello = container.addServlet("Hello", new Hello());
		hello.addMapping("/hello");
	}
}
```

Servlet 3.0 and later also allow you to configure the container with Java

The web container will automatically detect and run any class that implements WebApplicationInitializer

Servlet Registration can also be done with @WebServlet or in web.xml

Getting Spring Context in Servlet

```
public class ViewCustomer extends HttpServlet {
  private static final long serialVersionUID = 1L;

  public void doGet(HttpServletRequest req, HttpServletResponse resp)
                  throws ServletException, IOException {

    int custId = Integer.parseInt(req.getParameter("custId"));

    // get customerService bean from spring
    ServletContext context = getServletContext();
    WebApplicationContext applicationContext =
        WebApplicationContextUtils.getWebApplicationContext(context);
    CustomerService custServ = applicationContext.getBean(
        "customerService", CustomerService.class);

    // make customer available in request, for view rendering
    Customer cust = custServ.getCust(custId);
    req.setAttribute("cust", cust);

    // forward to view customer page
    req.getRequestDispatcher("customer.jsp").forward(req, resp);
  }
}
```

Inside a Servlet or Filter get the Spring Context from Web Context

After which you can get Spring Beans from it