渐行渐远的 Servlet

Servlet 简介

什么是 Servlet ?

Servlet 是一种基于 Java 技术的 Web 组件，用于生成动态内容，由容器管理。类似于其他 Java 技术组件，Servlet 是平台无关的 Java 类组成，并且由 Java Web 服务器加载执行。通常情况，由 Servlet 容器提供运行时环境。Servlet 容器，有时候也称作为 Servlet 引擎，作为Web服务器或应用服务器的一部分。通过请求和响应对话，提供Web 客户端与 Servlets 交互的能力。容器管理Servlets实例以及它们的生命周期。

从功能上，Servlet 介于 CGI（Common Gateway Interface）与服务扩展（如：Netscape Server API 或 Apache 模块）之间。

在体系上，Servlet 技术（或者规范）属于 Java EE 技术（规范）的一部分。不过 Servlet 并非一开始就隶属于 J2EE 或者 Java EE。接下来的小节将会介绍 Servlet 各个版本。

Servlet 版本

规范版本	发布时间	Java 平台	主要更新
Servlet 4.0	2017 年 9 月	Java EE 8	支持 HTTP/2
Servlet 3.1	2013 年 5 月	Java EE 7	非阻塞 I/O、HTTP 协议更新机制（WebSocket）
Servlet 3.0	2009 年 12 月	Java EE 6	可插拔、简化部署、异步 Servlet、安全、文件上传
Servlet 2.5	2005 年 9 月	Java EE 5	Annotation 支持
Servlet 2.4	2003 年 11月	J2EE 1.4	web.xml 支持 XML Scheme
Servlet 2.3	2001 年 8月	J2EE 1.3	新增 Filter、事件/监听器、Wrapper
Servlet 2.2	1999 年 8月	J2EE 1.2	作为 J2EE 的一部分， 以 `.war` 文件作为独立 web 应用

Servlet 核心 API

核心组件 API	说明	起始版本	Spring Framework 代表实现
`javax.servlet.Servlet`	动态内容组件	1.0	`DispatcherServlet`
`javax.servlet.Filter`	`Servlet` 过滤器	2.3	`CharacterEncodingFilter`
`javax.servlet.ServletContext`	Servlet 应用上下文		
`javax.servlet.AsyncContext`	异步上下文	3.0	无
`javax.servlet.ServletContextListener`	`ServletContext` 生命周期监听器	2.3	`ContextLoaderListener`
`javax.servlet.ServletRequestListener`	`ServletRequest` 生命周期监听器	2.3	`RequestContextListener`
`javax.servlet.http.HttpSessionListener`	`HttpSession` 生命周期监听器	2.3	`HttpSessionMutexListener`
`javax.servlet.AsyncListener`	异步上下文监听器	3.0	`StandardServletAsyncWebRequest`
`javax.servlet.ServletContainerInitializer`	Servlet 容器初始化器	3.0	`SpringServletContainerInitializer`

Servlet 组件注册

`Servlet` 注册

注册方式	传统方式	注解方式	编程方式
`Servlet` 注册	`web.xml` 部署 `<servlet>` + `<servlet-mapping>`	`@WebServlet`	`ServletContext#addServlet`
`Filter` 注册	`web.xml` 部署 `<filter>` + `<filter-mapping>`	`@WebFilter`	`ServletContext#addFilter`
`*Listener` 注册	`web.xml` 部署 `<listener>`	`@WebListener`	`ServletContext#addListener`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
         metadata-complete="true" version="2.5">

    <context-param>
        <description>
            Spring 配置文件路径参数，
            该参数值将被 org.springframework.web.context.ContextLoaderListener 使用
        </description>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            classpath*:/META-INF/spring/spring-context.xml
        </param-value>
    </context-param>

    <listener>
        <description>
            org.springframework.web.context.ContextLoaderListener 为可选申明Listener
        </description>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

</web-app>
```

Spring Servlet Web

理解 `Servlet` 生命周期

- 初始化：`init(ServletConfig)`
- 服务：`service(ServletRequest,ServletResponse)`
- 销毁：`destroy()`

`DispatcherServlet` 初始化过程

理解 `Filter` 生命周期

- 初始化：`init(FilterConfig)`
- 服务：`doFilter(ServletRequest,ServletResponse,FilterChain)`
- 销毁：`destroy()`

理解 `ServletContext` 生命周期

- 初始化：`contextInitialized(ServletContextEvent)`
- 销毁：`contextDestroyed(ServletContextEvent )`

Servlet 异步支持

`DeferredResult` 支持

`Callable` 支持

`CompletionStage` 支持

Spring Web MVC 异步 Servlet 实现原理

Java Specification Requests (JSR) : https://github.com/mercyblitz/jsr

Spring Boot Servlet Web

Spring Boot 嵌入式 Servlet 容器限制

Servlet 特性	兼容性	解决方案
`web.xml`	不支持	`RegistrationBean` 或 `@Bean` 注册
`ServletContainerInitializer`	不支持	`ServletContextInitializer`
`@WebServlet` 等	有限支持	依赖 `@ServletComponentScan`

参考资料一

> 87.2 Convert an Existing Application to Spring Boot
>
> you may need to add some configuration to your `Application` context, by replacing those elements from the `web.xml`, as follows:
>
> - A `@Bean` of type `Servlet` or `ServletRegistrationBean` installs that bean in the container as if it were a `<servlet/>` and `<servlet-mapping/>` in `web.xml`.
> - A `@Bean` of type `Filter` or `FilterRegistrationBean` behaves similarly (as a `<filter/>` and `<filter-mapping/>`).
> - An `ApplicationContext` in an XML file can be added through an `@ImportResource` in your `Application`. Alternatively, simple cases where annotation configuration is heavily used already can be recreated in a few lines as `@Bean` definitions.

参考资料二

> 27.4.2 Servlet Context Initialization

> Embedded servlet containers do not directly execute the Servlet 3.0+ `javax.servlet.ServletContainerInitializer` interface or Spring's `org.springframework.web.WebApplicationInitializer` interface. This is an intentional design decision intended to reduce the risk that third party libraries designed to run inside a war may break Spring Boot applications.

参考材料三

> Scanning for Servlets, Filters, and listeners
>
> When using an embedded container, automatic registration of classes annotated with `@WebServlet`, `@WebFilter`, and `@WebListener` can be enabled by using `@ServletComponentScan`.

Spring Boot Servlet 注册

通过 `RegistrationBean` 注册

- `ServletContextInitializer`
  - `RegistrationBean`
    - `ServletListenerRegistrationBean`
      - `@WebListener`
    - `FilterRegistrationBean`
      - `@WebFilter`
    - `ServletRegistrationBean`
      - `@WebServlet`

`@ServletComponentScan` 扫描 package -> `@Web*` -> `RegistrationBean` Bean 定义 -> `RegistrationBean` Bean

通过 `@Bean` 注册

通过 `@ServletComponentScan` 注册

Spring Boot 应用传统 Servlet 容器部署

基本原理

扩展 `SpringBootServletInitializer`

使用 Tomcat 7 插件（Servlet 3.0）

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.1</version>
            <executions>
                <execution>
                    <id>tomcat-run</id>
                    <goals>
                        <goal>exec-war-only</goal>
                    </goals>
                    <phase>package</phase>
                    <configuration>
                        <!-- ServletContext path -->
                        <path>/</path>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

使用 Tomcat 8 插件（Servlet 3.1）

```
<build>
    <plugins>
        <!-- Tomcat 8 Maven 插件用于构建可执行 war -->
        <!-- https://mvnrepository.com/artifact/org.apache.tomcat.maven/tomcat8-maven-plugin -->
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat8-maven-plugin</artifactId>
            <version>3.0-r1655215</version>
            <executions>
                <execution>
                    <id>tomcat-run</id>
                    <goals>
                        <!-- 最终打包成可执行的jar包 -->
                        <goal>exec-war-only</goal>
                    </goals>
                    <phase>package</phase>
```

```
                    <configuration>
                        <!-- ServletContext 路径 -->
                        <path>/</path>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>

<pluginRepositories>
    <pluginRepository>
        <!-- tomcat8-maven-plugin 所在仓库 -->
        <id>Alfresco</id>
        <name>Alfresco Repository</name>
        <url>https://artifacts.alfresco.com/nexus/content/repositories/public/</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
</pluginRepositories>
```

https://artifacts.alfresco.com/nexus/content/repositories/public/org/apache/tomcat/maven/tomcat-maven-plugin/3.0-r1655215/tomcat-maven-plugin-3.0-r1655215.pom

回顾 Spring Web 自动装配

版本依赖

- Spring Framework 3.1 +
- Servlet 3.0 +

Servlet SPI

Servlet SPI `ServletContainerInitializer`，参考 Servlet 3.0 规范

配合 `@HandlesTypes`

Spring 适配

`SpringServletContainerInitializer`

Spring SPI

基础接口：`WebApplicationInitializer`

编程驱动：`AbstractDispatcherServletInitializer`

注解驱动：`AbstractAnnotationConfigDispatcherServletInitializer`