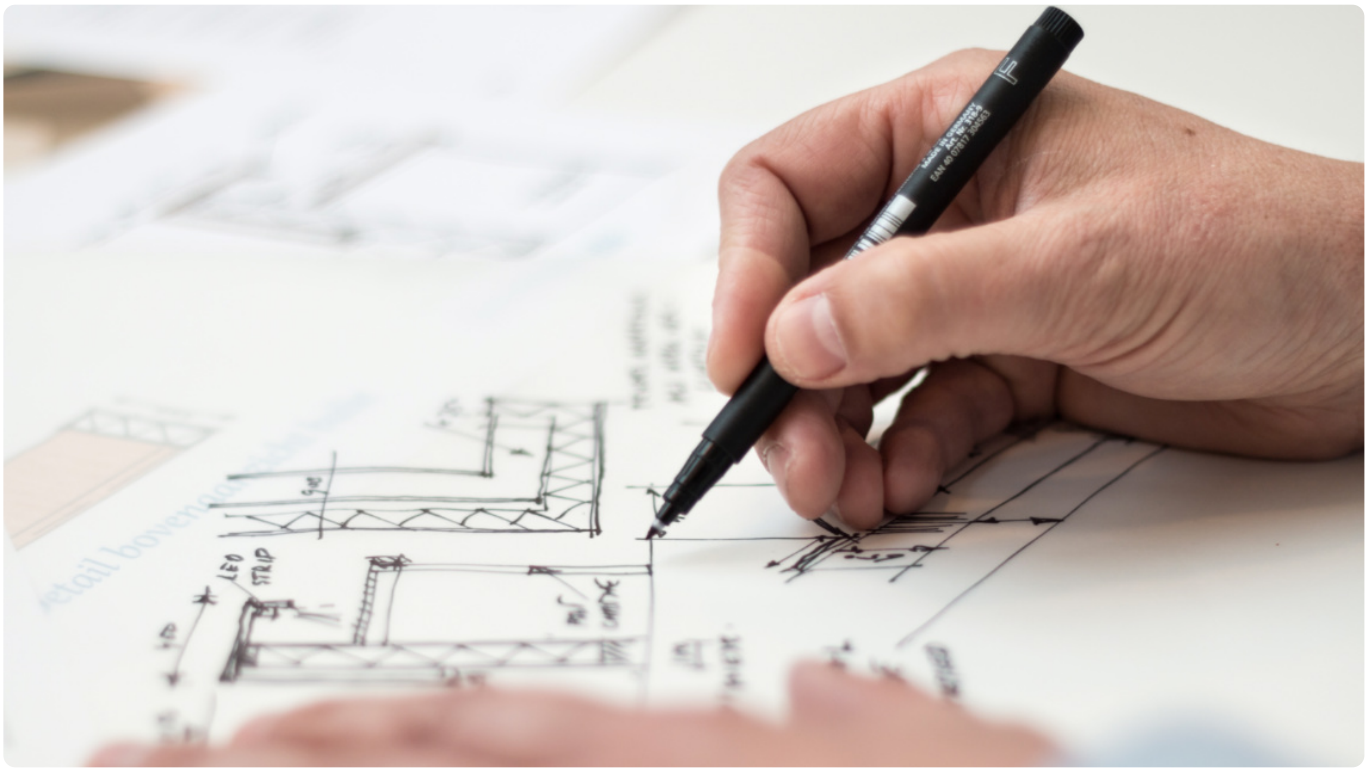
28 | 新特性：Tomcat和Jetty如何处理Spring Boot应用?

2019-07-13 李号双

深入拆解Tomcat & Jetty

进入课程 >



讲述：李号双
时长 06:12　大小 5.69M

为了方便开发和部署，Spring Boot 在内部启动了一个嵌入式的 Web 容器。我们知道 Tomcat 和 Jetty 是组件化的设计，要启动 Tomcat 或者 Jetty 其实就是启动这些组件。在 Tomcat 独立部署的模式下，我们通过 startup 脚本来启动 Tomcat，Tomcat 中的 Bootstrap 和 Catalina 会负责初始化类加载器，并解析`server.xml`和启动这些组件。

在内嵌式的模式下，Bootstrap 和 Catalina 的工作就由 Spring Boot 来做了，Spring Boot 调用了 Tomcat 和 Jetty 的 API 来启动这些组件。那 Spring Boot 具体是怎么做的呢？而作为程序员，我们如何向 SpringBoot 中的 Tomcat 注册 Servlet 或者 Filter 呢？我们又如何定制内嵌式的 Tomcat？今天我们就来聊聊这些话题。

Spring Boot 中 Web 容器相关的接口

既然要支持多种 Web 容器，Spring Boot 对内嵌式 Web 容器进行了抽象，定义了**WebServer**接口：

复制代码

```
public interface WebServer {
    void start() throws WebServerException;
    void stop() throws WebServerException;
    int getPort();
}
```

各种 Web 容器比如 Tomcat 和 Jetty 需要去实现这个接口。

Spring Boot 还定义了一个工厂**ServletWebServerFactory**来创建 Web 容器，返回的对象就是上面提到的 WebServer。

复制代码

```
public interface ServletWebServerFactory {
    WebServer getWebServer(ServletContextInitializer... initializers);
}
```

可以看到 getWebServer 有个参数，类型是**ServletContextInitializer**。它表示 ServletContext 的初始化器，用于 ServletContext 中的一些配置：

复制代码

```
public interface ServletContextInitializer {
    void onStartup(ServletContext servletContext) throws ServletException;
}
```

这里请注意，上面提到的 getWebServer 方法会调用 ServletContextInitializer 的 onStartup 方法，也就是说如果你想在 Servlet 容器启动时做一些事情，比如注册你自己的 Servlet，可以实现一个 ServletContextInitializer，在 Web 容器启动时，Spring Boot 会把所有实现了 ServletContextInitializer 接口的类收集起来，统一调它们的 onStartup 方法。

为了支持对内嵌式 Web 容器的定制化，Spring Boot 还定义了**WebServerFactoryCustomizerBeanPostProcessor**接口，它是一个 BeanPostProcessor，它在 postProcessBeforeInitialization 过程中去寻找 Spring 容器中 WebServerFactoryCustomizer

类型的 Bean，并依次调用 WebServerFactoryCustomizer

接口的 customize 方法做一些定制化。

复制代码

```
public interface WebServerFactoryCustomizer<T extends WebServerFactory> {
    void customize(T factory);
}
```

内嵌式 Web 容器的创建和启动

铺垫了这些接口，我们再来看看 Spring Boot 是如何实例化和启动一个 Web 容器的。我们知道，Spring 的核心是一个 ApplicationContext，它的抽象实现类 AbstractApplicationContext

实现了著名的**refresh**方法，它用来新建或者刷新一个 ApplicationContext，在 refresh 方法中会调用 onRefresh 方法，AbstractApplicationContext 的子类可以重写这个方法 onRefresh 方法，来实现特定 Context 的刷新逻辑，因此 ServletWebServerApplicationContext 就是通过重写 onRefresh 方法来创建内嵌式的 Web 容器，具体创建过程是这样的：

复制代码

```
@Override
protected void onRefresh() {
      super.onRefresh();
      try {
         // 重写 onRefresh 方法，调用 createWebServer 创建和启动 Tomcat
         createWebServer();
      }
      catch (Throwable ex) {
      }
}

```

```
//createWebServer 的具体实现
private void createWebServer() {
    // 这里 WebServer 是 Spring Boot 抽象出来的接口，具体实现类就是不同的 Web 容器
    WebServer webServer = this.webServer;
    ServletContext servletContext = this.getServletContext();

    // 如果 Web 容器还没创建
    if (webServer == null && servletContext == null) {
        // 通过 Web 容器工厂来创建
        ServletWebServerFactory factory = this.getWebServerFactory();
        // 注意传入了一个 "SelfInitializer"
        this.webServer = factory.getWebServer(new ServletContextInitializer[]{this.getSe

    } else if (servletContext != null) {
        try {
            this.getSelfInitializer().onStartup(servletContext);
        } catch (ServletException var4) {
          ...
        }
    }

    this.initPropertySources();
}
```

再来看看 getWebSever 具体做了什么，以 Tomcat 为例，主要调用 Tomcat 的 API 去创建各种组件：

复制代码

```
public WebServer getWebServer(ServletContextInitializer... initializers) {
    //1. 实例化一个 Tomcat，可以理解为 Server 组件。
    Tomcat tomcat = new Tomcat();

    //2. 创建一个临时目录
    File baseDir = this.baseDirectory != null ? this.baseDirectory : this.createTempDir
    tomcat.setBaseDir(baseDir.getAbsolutePath());

    //3. 初始化各种组件
    Connector connector = new Connector(this.protocol);
    tomcat.getService().addConnector(connector);
    this.customizeConnector(connector);
    tomcat.setConnector(connector);
    tomcat.getHost().setAutoDeploy(false);
    this.configureEngine(tomcat.getEngine());

    //4. 创建定制版的 "Context" 组件。
    this.prepareContext(tomcat.getHost(), initializers);
    return this.getTomcatWebServer(tomcat);
```

```
20 }
```

你可能好奇 prepareContext 方法是做什么的呢？这里的 Context 是指**Tomcat 中的 Context 组件**，为了方便控制 Context 组件的行为，Spring Boot 定义了自己的 TomcatEmbeddedContext，它扩展了 Tomcat 的 StandardContext：

复制代码

```
1 class TomcatEmbeddedContext extends StandardContext {}
```

注册 Servlet 的三种方式

1. Servlet 注解

在 Spring Boot 启动类上加上 @ServletComponentScan 注解后，使用 @WebServlet、@WebFilter、@WebListener 标记的 Servlet、Filter、Listener 就可以自动注册到 Servlet 容器中，无需其他代码，我们通过下面的代码示例来理解一下。

复制代码

```
1 @SpringBootApplication
2 @ServletComponentScan
3 public class xxxApplication
4 {}
```

复制代码

```
1 @WebServlet("/hello")
2 public class HelloServlet extends HttpServlet {}
```

在 Web 应用的入口类上加上 @ServletComponentScan， 并且在 Servlet 类上加上 @WebServlet，这样 SpringBoot 会负责将 Servlet 注册到内嵌的 Tomcat 中。

2. ServletRegistrationBean

同时 Spring Boot 也提供了 ServletRegistrationBean、FilterRegistrationBean 和 ServletListenerRegistrationBean 这三个类分别用来注册 Servlet、Filter、Listener。假如要注册一个 Servlet，可以这样做：

复制代码

```
@Bean
public ServletRegistrationBean servletRegistrationBean() {
    return new ServletRegistrationBean(new HelloServlet(),"/hello");
}
```

这段代码实现的方法返回一个 ServletRegistrationBean，并将它当作 Bean 注册到 Spring 中，因此你需要把这段代码放到 Spring Boot 自动扫描的目录中，或者放到 @Configuration 标识的类中。

3. 动态注册

你还可以创建一个类去实现前面提到的 ServletContextInitializer 接口，并把它注册为一个 Bean，Spring Boot 会负责调用这个接口的 onStartup 方法。

复制代码

```
@Component
public class MyServletRegister implements ServletContextInitializer {

    @Override
    public void onStartup(ServletContext servletContext) {

        //Servlet 3.0 规范新的 API
        ServletRegistration myServlet = servletContext
                .addServlet("HelloServlet", HelloServlet.class);

        myServlet.addMapping("/hello");

        myServlet.setInitParameter("name", "Hello Servlet");
    }

}
```

这里请注意两点：

ServletRegistrationBean 其实也是通过 ServletContextInitializer 来实现的，它实现了 ServletContextInitializer 接口。

注意到 onStartup 方法的参数是我们熟悉的 ServletContext，可以通过调用它的 addServlet 方法来动态注册新的 Servlet，这是 Servlet 3.0 以后才有的功能。

Web 容器的定制

我们再来考虑一个问题，那就是如何在 Spring Boot 中定制 Web 容器。在 Spring Boot 2.0 中，我们可以通过两种方式来定制 Web 容器。

第一种方式是通过通用的 Web 容器工厂 ConfigurableServletWebServerFactory，来定制一些 Web 容器通用的参数：

复制代码

```
@Component
public class MyGeneralCustomizer implements
  WebServerFactoryCustomizer<ConfigurableServletWebServerFactory> {

    public void customize(ConfigurableServletWebServerFactory factory) {
        factory.setPort(8081);
        factory.setContextPath("/hello");
     }
}
```

第二种方式是通过特定 Web 容器的工厂比如 TomcatServletWebServerFactory 来进一步定制。下面的例子里，我们给 Tomcat 增加一个 Valve，这个 Valve 的功能是向请求头里添加 traceid，用于分布式追踪。TraceValve 的定义如下：

复制代码

```
class TraceValve extends ValveBase {
    @Override
    public void invoke(Request request, Response response) throws IOException, ServletE

        request.getCoyoteRequest().getMimeHeaders().
        addValue("traceid").setString("1234xxxxabcd");

        Valve next = getNext();
        if (null == next) {
            return;
        }
```

```
12
13          next.invoke(request, response);
14      }
15
16  }
```

跟第一种方式类似，再添加一个定制器，代码如下：

复制代码

```
1  @Component
2  public class MyTomcatCustomizer implements
3          WebServerFactoryCustomizer<TomcatServletWebServerFactory> {
4
5      @Override
6      public void customize(TomcatServletWebServerFactory factory) {
7          factory.setPort(8081);
8          factory.setContextPath("/hello");
9          factory.addEngineValves(new TraceValve() );
10
11     }
12 }
```

本期精华

今天我们学习了 Spring Boot 如何利用 Web 容器的 API 来启动 Web 容器、如何向 Web 容器注册 Servlet，以及如何定制化 Web 容器，除了给 Web 容器配置参数，还可以增加或者修改 Web 容器本身的组件。

课后思考

我在文章中提到，通过 ServletContextInitializer 接口可以向 Web 容器注册 Servlet，那 ServletContextInitializer 跟 Tomcat 中的 ServletContainerInitializer 有什么区别和联系呢？

不知道今天的内容你消化得如何？如果还有疑问，请大胆的在留言区提问，也欢迎你把你的课后思考和心得记录下来，与我和其他同学一起讨论。如果你觉得今天有所收获，欢迎你把它分享给你的朋友。

上一篇　27 | 新特性：Tomcat如何支持异步Servlet？

精选留言 (5)

写留言

despacito
2019-07-13

老师，springboot 中 getWebServer方法的实现类不仅有tomcat，还有其他web容器，比如jetty，那为什么我们在运行启动类的时候默认都是用的tomcat容器，如果我运行启动类的时候想用jetty作为应用容器，应该怎么做？

展开

1

刘冬
2019-07-13

和"飞翔"同问： 有@RestController，为什么还要自己去注册Servlet给Tomcat?
我感觉老师很善于将负责的问题、长的逻辑链讲的简洁清晰，还请老师帮忙详细说明一

下。
谢谢！
展开

1

飞翔
2019-07-13

老师 sprongboot 不注册servlet 给tomcat 直接用@controller 就能实现servlet功能是咋回事呀

大漠落木
2019-07-13

@FunctionalInterface
org.springframework.boot.web.servlet.ServletContextInitializer
This interface is primarily designed to allow ServletContextInitializers to bemanaged by Spring and not the Servlet container.
javax.servlet.ServletContainerInitializer...
展开

nightmare
2019-07-13

感觉还是要跟着操作一下才能懂了
展开