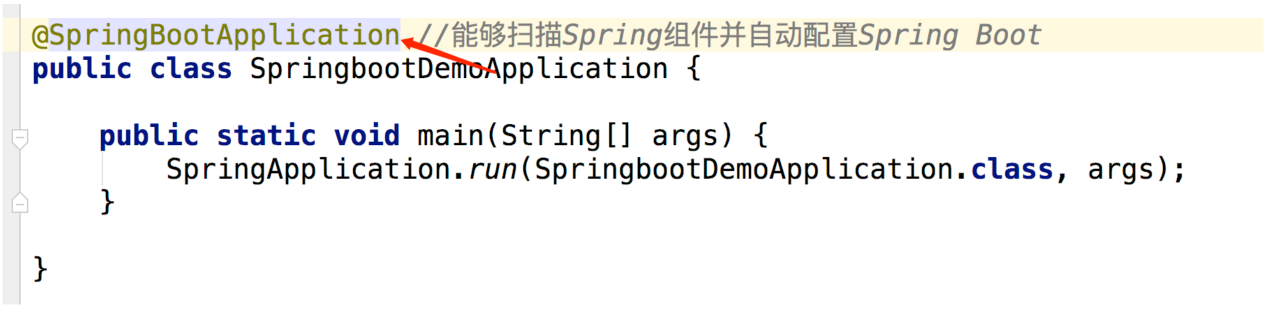
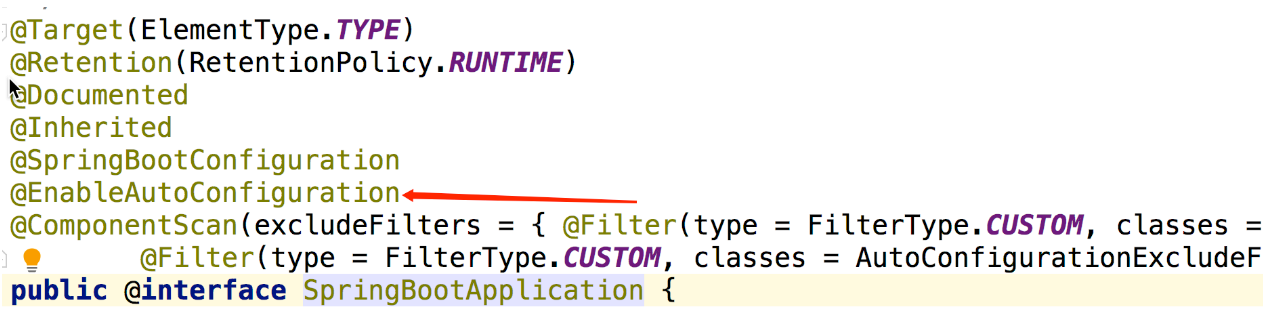
SpringBoot拓展

嵌入式Tomcat源码剖析

1、进入SpringBoot启动类，点进@SpringBootApplication源码，如下图

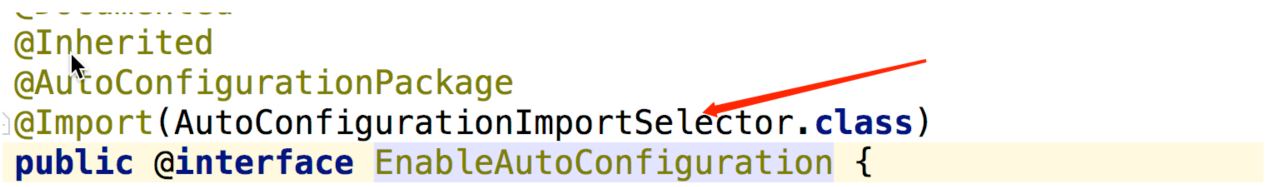


```
@SpringBootApplication //能够扫描Spring组件并自动配置Spring Boot
public class SpringbootDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootDemoApplication.class, args);
    }

}
```



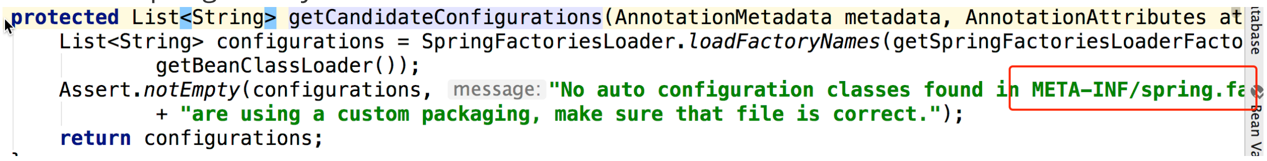
```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes =
        @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeF
public @interface SpringBootApplication {
```

2、继续点进@EnableAutoConfiguration,进入该注解，如下图



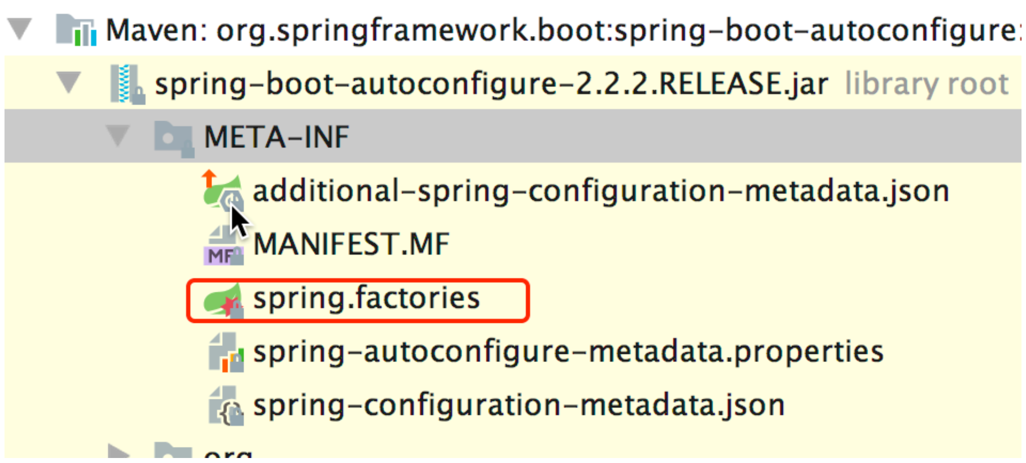
```
@Inherited
@AutoConfigurationPackage
@Import(AutoConfigurationImportSelector.class)
public @interface EnableAutoConfiguration {
```

3、上图中使用@Import注解对AutoConfigurationImportSelector 类进行了引入，该类做了什么事情呢？进入源码，首先调用selectImport()方法，在该方法中调用了getAutoConfigurationEntry（）方法，在之中又调用了getCandidateConfigurations()方法，getCandidateConfigurations()方法就去META-INF/spring.factory配置文件中加载相关配置类



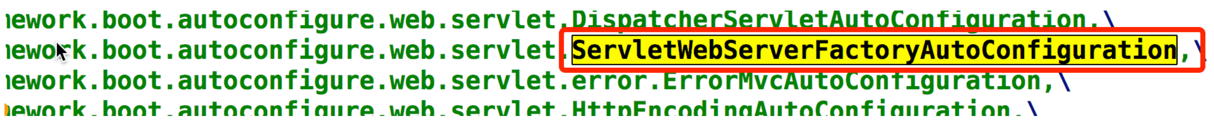
```
protected List<String> getCandidateConfigurations(AnnotationMetadata metadata, AnnotationAttributes at
    List<String> configurations = SpringFactoriesLoader.loadFactoryNames(getSpringFactoriesLoaderFacto
            getBeanClassLoader());
    Assert.notEmpty(configurations, message: "No auto configuration classes found in META-INF/spring.fa
            + "are using a custom packaging, make sure that file is correct.");
    return configurations;
```

这个spring.factories配置文件是加载的spring-boot-autoconfigure的配置文件



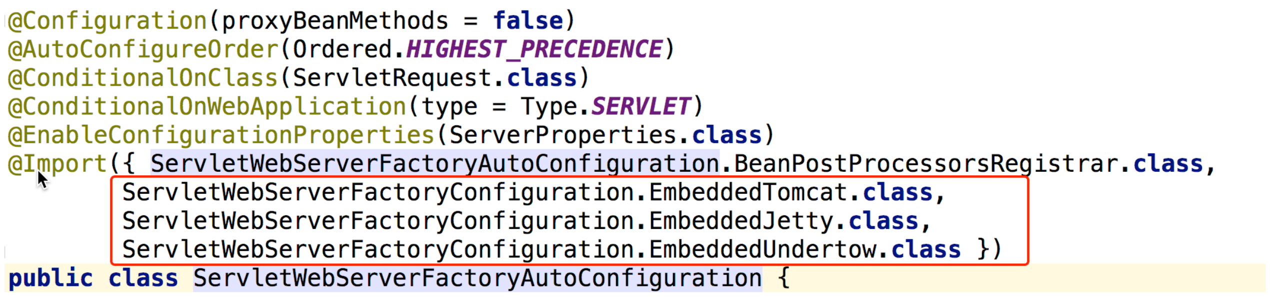
```
Maven: org.springframework.boot:spring-boot-autoconfigure:
  spring-boot-autoconfigure-2.2.2.RELEASE.jar  library root
    META-INF
      additional-spring-configuration-metadata.json
      MANIFEST.MF
      spring.factories
      spring-autoconfigure-metadata.properties
      spring-configuration-metadata.json
    org
```

继续打开spring.factories配置文件，找到tomcat所在的类，tomcat加载在ServletWebServerFactoryAutoConfiguration配置类中



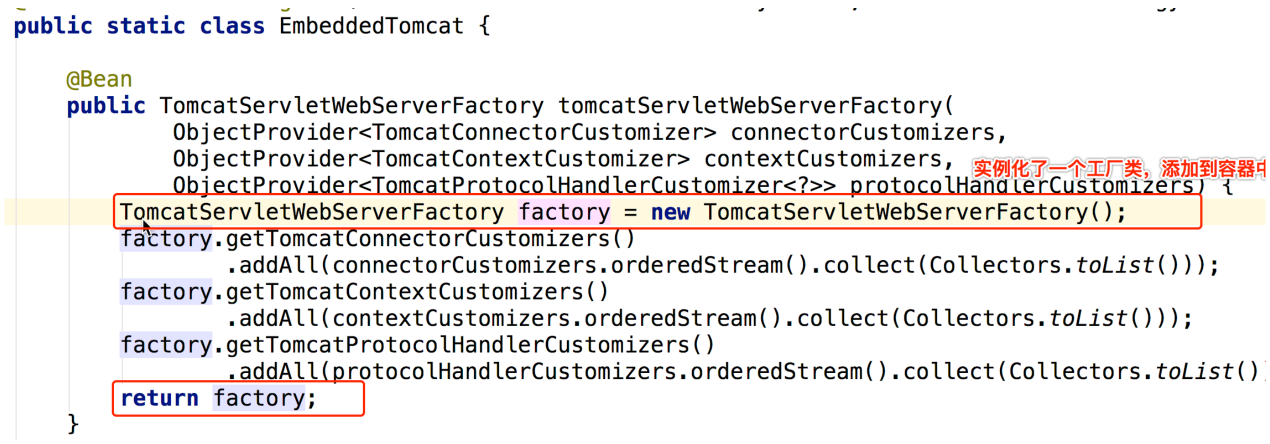
```
ework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration,\
ework.boot.autoconfigure.web.servlet.ServletWebServerFactoryAutoConfiguration,\
ework.boot.autoconfigure.web.servlet.error.ErrorMvcAutoConfiguration,\
ework.boot.autoconfigure.web.servlet.HttpEncodingAutoConfiguration,\
```

进入该类，里面也通过@Import注解将EmbeddedTomcat、EmbeddedJetty、EmbeddedUndertow等嵌入式容器类加载进来了，springboot默认是启动嵌入式tomcat容器，如果要改变启动jetty或者undertow容器，需在pom文件中去设置。如下图



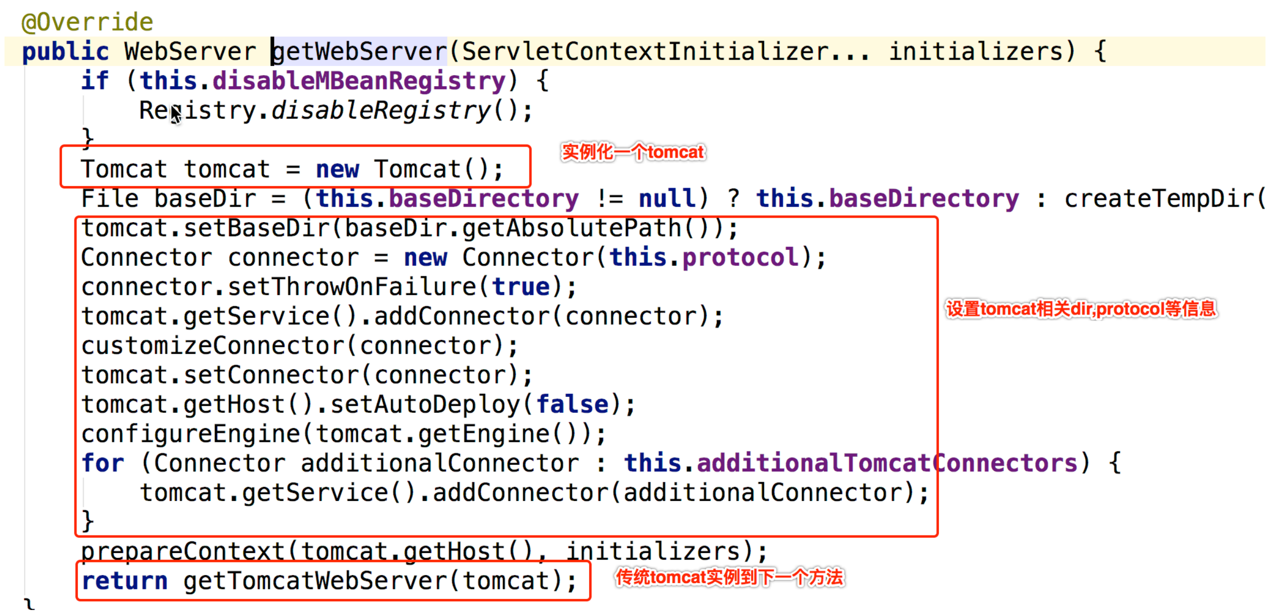
```
@Configuration(proxyBeanMethods = false)
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE)
@ConditionalOnClass(ServletRequest.class)
@ConditionalOnWebApplication(type = Type.SERVLET)
@EnableConfigurationProperties(ServerProperties.class)
@Import({ ServletWebServerFactoryAutoConfiguration.BeanPostProcessorsRegistrar.class,
        ServletWebServerFactoryConfiguration.EmbeddedTomcat.class,
        ServletWebServerFactoryConfiguration.EmbeddedJetty.class,
        ServletWebServerFactoryConfiguration.EmbeddedUndertow.class })
public class ServletWebServerFactoryAutoConfiguration {
```

继续进入EmbeddedTomcat类中，见下图：



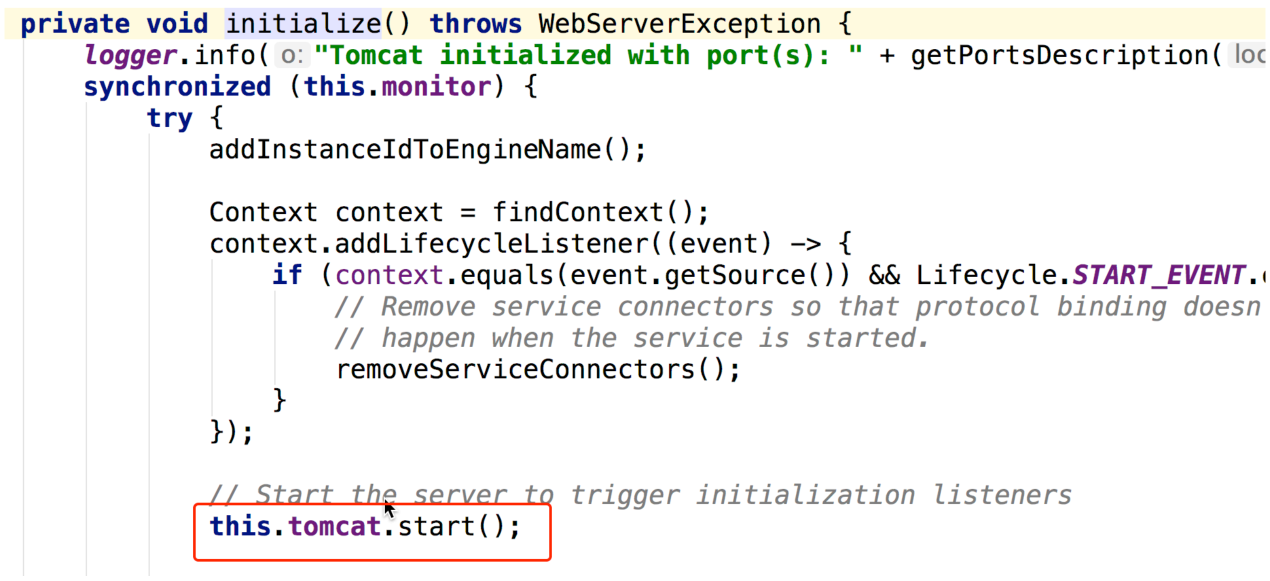
```
public static class EmbeddedTomcat {

    @Bean
    public TomcatServletWebServerFactory tomcatServletWebServerFactory(
            ObjectProvider<TomcatConnectorCustomizer> connectorCustomizers,
            ObjectProvider<TomcatContextCustomizer> contextCustomizers,
            ObjectProvider<TomcatProtocolHandlerCustomizer<?>> protocolHandlerCustomizers) {
        TomcatServletWebServerFactory factory = new TomcatServletWebServerFactory();
        factory.getTomcatConnectorCustomizers()
                .addAll(connectorCustomizers.orderedStream().collect(Collectors.toList()));
        factory.getTomcatContextCustomizers()
                .addAll(contextCustomizers.orderedStream().collect(Collectors.toList()));
        factory.getTomcatProtocolHandlerCustomizers()
                .addAll(protocolHandlerCustomizers.orderedStream().collect(Collectors.toList()
        return factory;
    }
```

进入TomcatServletWebServerFactory类，里面的getWebServer（）是关键方法，如图：



```
@Override
public WebServer getWebServer(ServletContextInitializer... initializers) {
    if (this.disableMBeanRegistry) {
        Registry.disableRegistry();
    }
    Tomcat tomcat = new Tomcat();
    File baseDir = (this.baseDirectory != null) ? this.baseDirectory : createTempDir(
    tomcat.setBaseDir(baseDir.getAbsolutePath());
    Connector connector = new Connector(this.protocol);
    connector.setThrowOnFailure(true);
    tomcat.getService().addConnector(connector);
    customizeConnector(connector);
    tomcat.setConnector(connector);
    tomcat.getHost().setAutoDeploy(false);
    configureEngine(tomcat.getEngine());
    for (Connector additionalConnector : this.additionalTomcatConnectors) {
        tomcat.getService().addConnector(additionalConnector);
    }
    prepareContext(tomcat.getHost(), initializers);
    return getTomcatWebServer(tomcat);
}
```

继续进入getTomcatWebServer()等方法，一直往下跟到tomcat初始化方法，调用tomcat.start()方法，tomcat就正式开启运行，见图

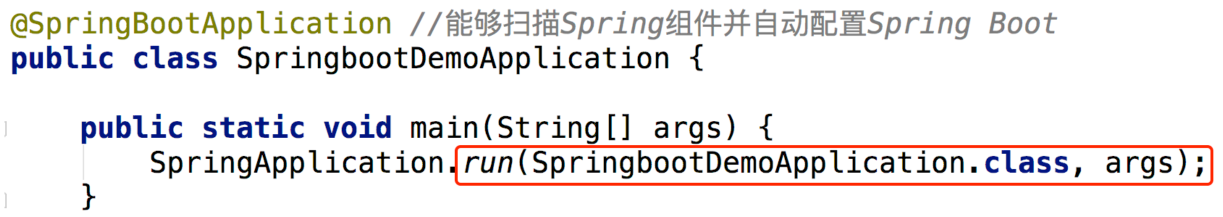


```
private void initialize() throws WebServerException {
    logger.info( o: "Tomcat initialized with port(s): " + getPortsDescription( loc
    synchronized (this.monitor) {
        try {
            addInstanceIdToEngineName();

            Context context = findContext();
            context.addLifecycleListener((event) -> {
                if (context.equals(event.getSource()) && Lifecycle.START_EVENT.
                    // Remove service connectors so that protocol binding doesn
                    // happen when the service is started.
                    removeServiceConnectors();
                }
            });

            // Start the server to trigger initialization listeners
            this.tomcat.start();
```

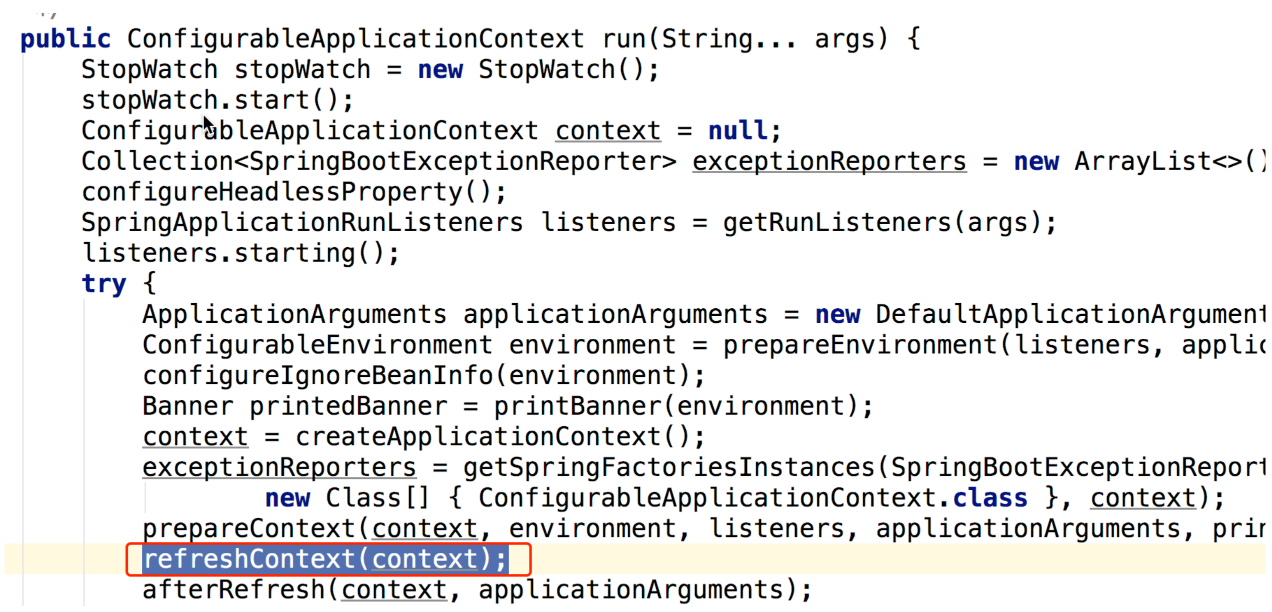
走到这里tomcat在springboot中的配置以及最终启动的流程就走完了，相信大家肯定有一个疑问，上上图中的getWebServer()方法是在哪里调用的呢？上面的代码流程并没有发现getWebServer()被调用的地方。因为getWebServer()方法的调用根本就不在上面的代码流程中，它是在另外一个流程中被调用的

源码解析之调用getWebServer()

首先进入SpringBoot启动类的run方法：

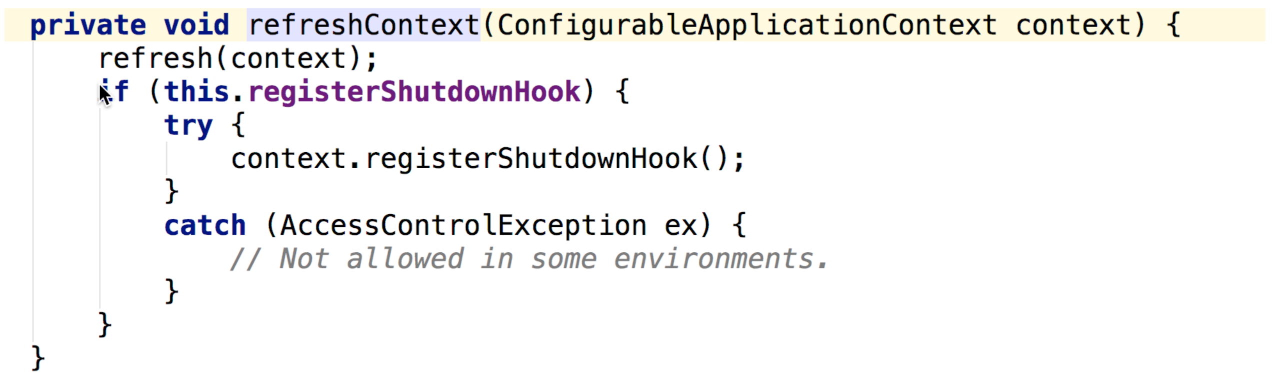


```
@SpringBootApplication //能够扫描Spring组件并自动配置Spring Boot
public class SpringbootDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootDemoApplication.class, args);
    }
```



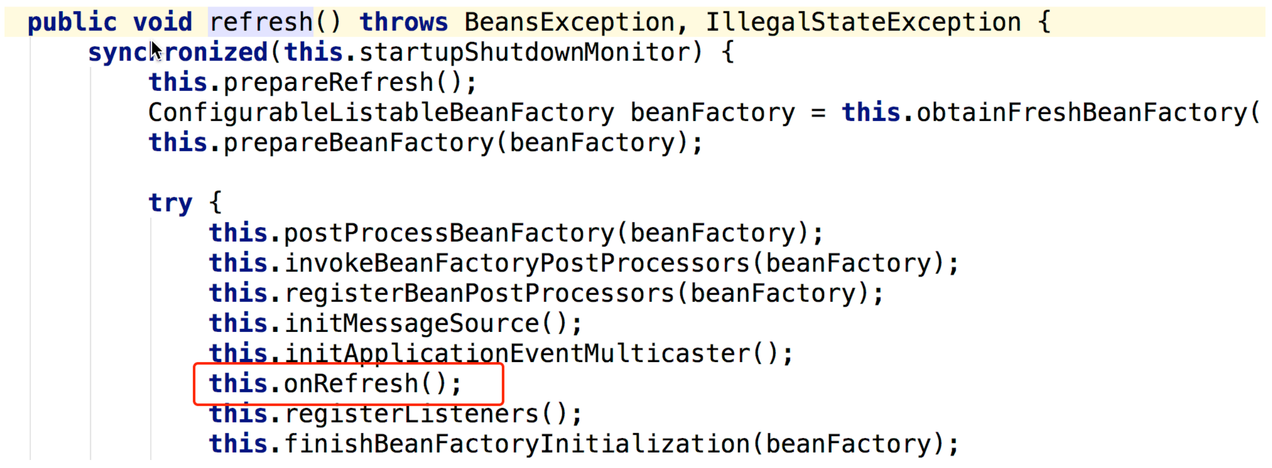
```java
public ConfigurableApplicationContext run(String... args) {
    StopWatch stopWatch = new StopWatch();
    stopWatch.start();
    ConfigurableApplicationContext context = null;
    Collection<SpringBootExceptionReporter> exceptionReporters = new ArrayList<>()
    configureHeadlessProperty();
    SpringApplicationRunListeners listeners = getRunListeners(args);
    listeners.starting();
    try {
        ApplicationArguments applicationArguments = new DefaultApplicationArgument
        ConfigurableEnvironment environment = prepareEnvironment(listeners, applic
        configureIgnoreBeanInfo(environment);
        Banner printedBanner = printBanner(environment);
        context = createApplicationContext();
        exceptionReporters = getSpringFactoriesInstances(SpringBootExceptionReport
                new Class[] { ConfigurableApplicationContext.class }, context);
        prepareContext(context, environment, listeners, applicationArguments, prir
        refreshContext(context);
        afterRefresh(context, applicationArguments);
```

进入refreshContext()方法，如图：

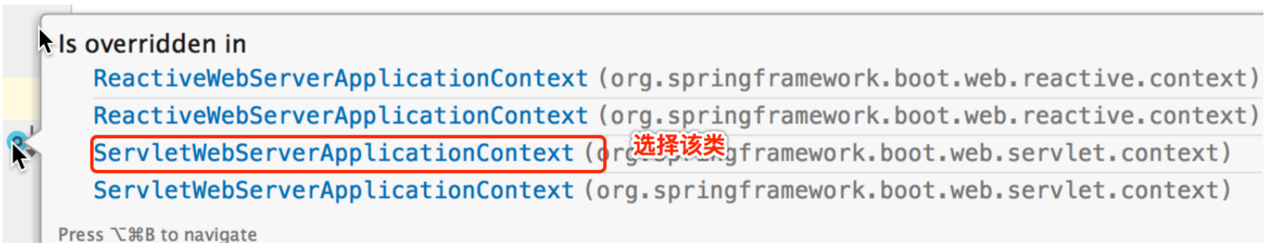


```java
private void refreshContext(ConfigurableApplicationContext context) {
    refresh(context);
    if (this.registerShutdownHook) {
        try {
            context.registerShutdownHook();
        }
        catch (AccessControlException ex) {
            // Not allowed in some environments.
        }
    }
}
```

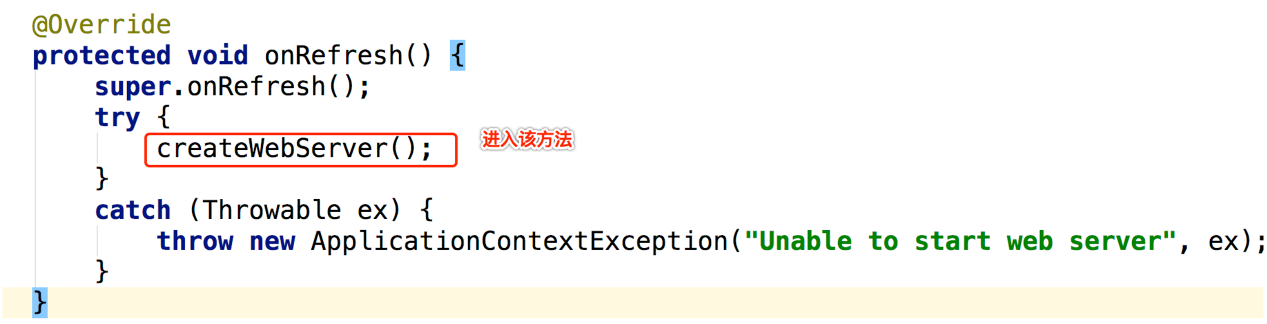
一直点击refresh()方法，如图：



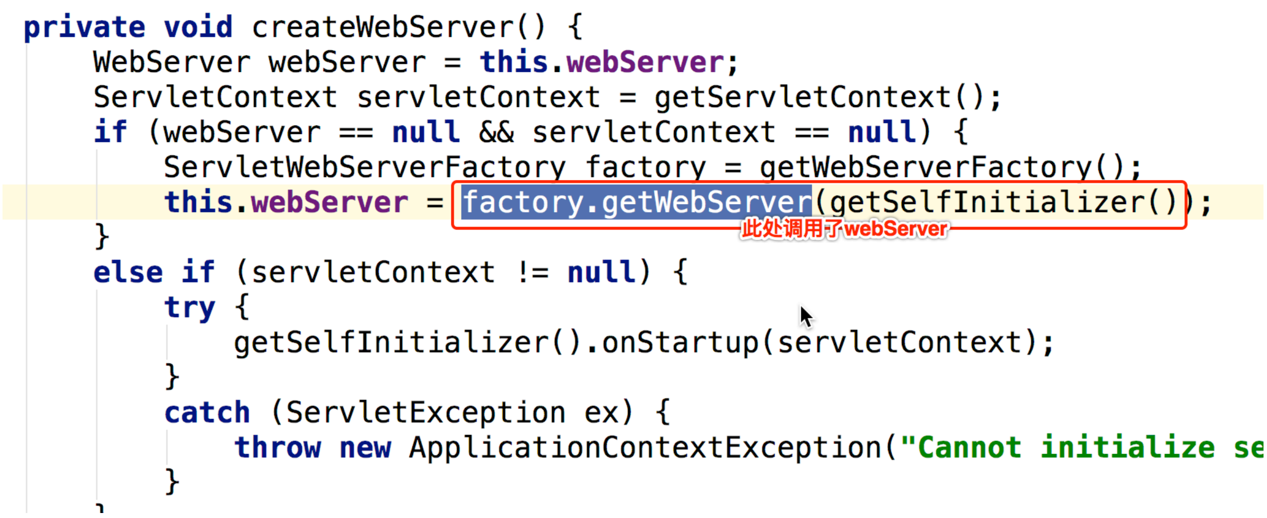
```java
public void refresh() throws BeansException, IllegalStateException {
    synchronized(this.startupShutdownMonitor) {
        this.prepareRefresh();
        ConfigurableListableBeanFactory beanFactory = this.obtainFreshBeanFactory(
        this.prepareBeanFactory(beanFactory);

        try {
            this.postProcessBeanFactory(beanFactory);
            this.invokeBeanFactoryPostProcessors(beanFactory);
            this.registerBeanPostProcessors(beanFactory);
            this.initMessageSource();
            this.initApplicationEventMulticaster();
            this.onRefresh();
            this.registerListeners();
            this.finishBeanFactoryInitialization(beanFactory);
```



```
Is overridden in
    ReactiveWebServerApplicationContext (org.springframework.boot.web.reactive.context)
    ReactiveWebServerApplicationContext (org.springframework.boot.web.reactive.context)
    ServletWebServerApplicationContext (org.springframework.boot.web.servlet.context)
    ServletWebServerApplicationContext (org.springframework.boot.web.servlet.context)
Press ⌥⌘B to navigate
```

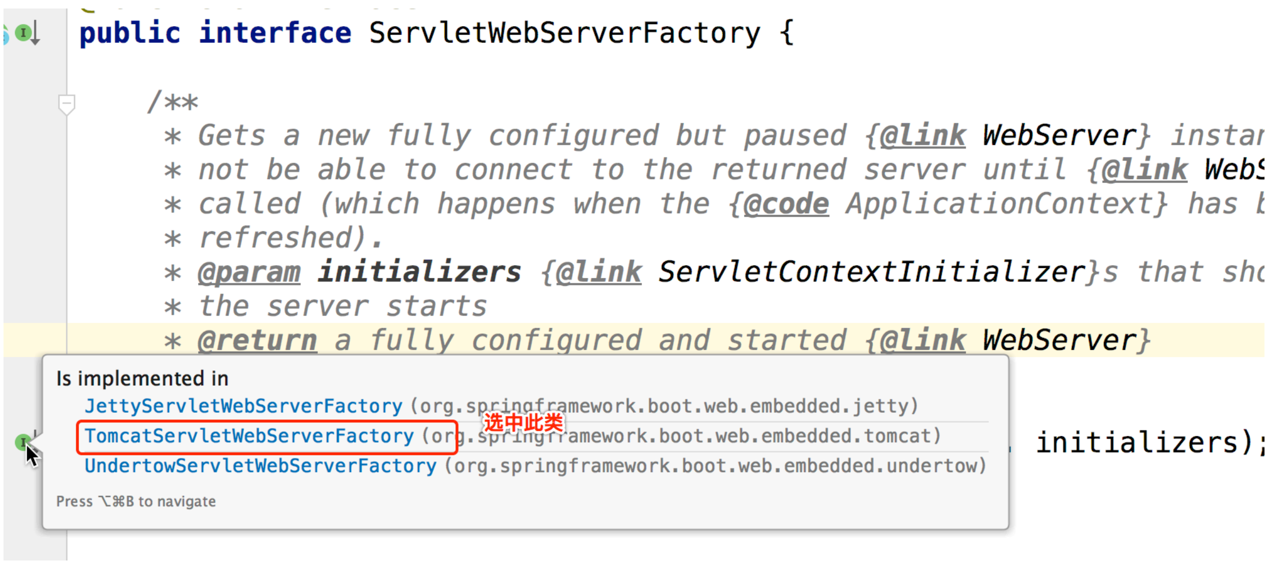


```java
@Override
protected void onRefresh() {
    super.onRefresh();
    try {
        createWebServer();
    }
    catch (Throwable ex) {
        throw new ApplicationContextException("Unable to start web server", ex);
    }
}
```



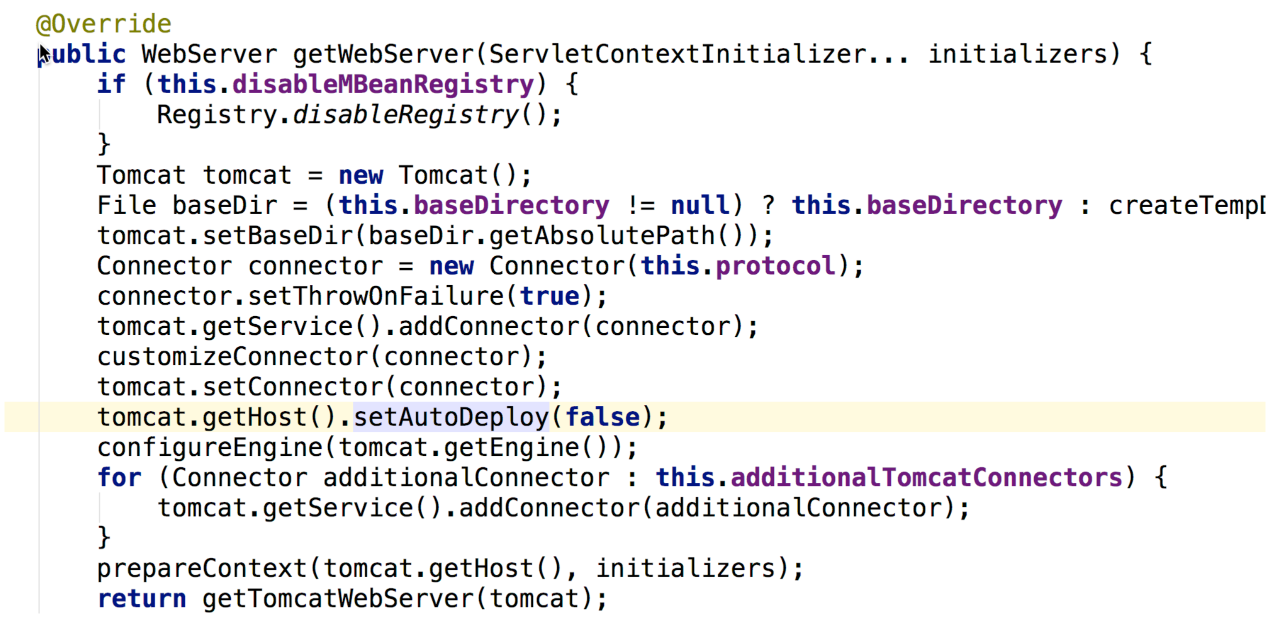
```java
private void createWebServer() {
    WebServer webServer = this.webServer;
    ServletContext servletContext = getServletContext();
    if (webServer == null && servletContext == null) {
        ServletWebServerFactory factory = getWebServerFactory();
        this.webServer = factory.getWebServer(getSelfInitializer());
    }
    else if (servletContext != null) {
        try {
            getSelfInitializer().onStartup(servletContext);
        }
        catch (ServletException ex) {
            throw new ApplicationContextException("Cannot initialize se
        }
```

继续进入getWebServer（）方法，如图：



```java
public interface ServletWebServerFactory {

    /**
     * Gets a new fully configured but paused {@link WebServer} instar
     * not be able to connect to the returned server until {@link WebS
     * called (which happens when the {@code ApplicationContext} has b
     * refreshed).
     * @param initializers {@link ServletContextInitializer}s that sho
     * the server starts
     * @return a fully configured and started {@link WebServer}
                                                    initializers);
```

```
Is implemented in
    JettyServletWebServerFactory (org.springframework.boot.web.embedded.jetty)
    TomcatServletWebServerFactory (org.springframework.boot.web.embedded.tomcat)
    UndertowServletWebServerFactory (org.springframework.boot.web.embedded.undertow)
Press ⌥⌘B to navigate
```



```java
@Override
public WebServer getWebServer(ServletContextInitializer... initializers) {
    if (this.disableMBeanRegistry) {
        Registry.disableRegistry();
    }
    Tomcat tomcat = new Tomcat();
    File baseDir = (this.baseDirectory != null) ? this.baseDirectory : createTempD
    tomcat.setBaseDir(baseDir.getAbsolutePath());
    Connector connector = new Connector(this.protocol);
    connector.setThrowOnFailure(true);
    tomcat.getService().addConnector(connector);
    customizeConnector(connector);
    tomcat.setConnector(connector);
    tomcat.getHost().setAutoDeploy(false);
    configureEngine(tomcat.getEngine());
    for (Connector additionalConnector : this.additionalTomcatConnectors) {
        tomcat.getService().addConnector(additionalConnector);
    }
    prepareContext(tomcat.getHost(), initializers);
    return getTomcatWebServer(tomcat);
```

最终就调用了TomcatServletWebServerFactory类的getWebServer()方法。