Spring boot 启动流程分析

现在以一个简单的 spring demo 程序， 讲解一下 spring boot 启动过程
Demo 代码：

```
package com.allen.spring.src.learning.applicationListener;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

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

}
```

上面通过 @SpringBootApplication 注解了一个启动类，然后 调用
SpringApplication.run 方法

1，SpringApplication 类的静态 run 方法

```
public static ConfigurableApplicationContext run(Class<?> primarySource, String... args) {
    return run(new Class[]{primarySource}, args);
}
public static ConfigurableApplicationContext run(Class<?>[] primarySources, String[] args) {
    return (new SpringApplication(primarySources)).run(args);
}
```

如上源代码，通过调用了另外一个静态方法 run，然后调用 SpringApplication 的 构造器 并且执行正在的 run 方法

2，SpringApplication 构造器 调用分析，见下面代码

```
public SpringApplication(Class... primarySources) {
```


阿伦读源码 allengent@163.com

```
        this((ResourceLoader)null, primarySources);
    }
    public SpringApplication(ResourceLoader resourceLoader, Class... primarySources) {
        this.sources = new LinkedHashSet();
        this.bannerMode = Mode.CONSOLE;
        this.logStartupInfo = true;
        this.addCommandLineProperties = true;
        this.headless = true;
        this.registerShutdownHook = true;
        this.additionalProfiles = new HashSet();
        this.resourceLoader = resourceLoader;
        Assert.notNull(primarySources, "PrimarySources must not be null");
        this.primarySources = new LinkedHashSet(Arrays.asList(primarySources));
        this.webApplicationType = this.deduceWebApplicationType();
        this.setInitializers(this.getSpringFactoriesInstances(ApplicationContextInitializer.class));
        this.setListeners(this.getSpringFactoriesInstances(ApplicationListener.class));
        this.mainApplicationClass = this.deduceMainApplicationClass();
    }
```

2.1 deduceWebApplicationType()

分析项目 web 类型 ，返回值 WebApplicationType.REACTIVE ， WebApplicationType.SERVLET

2.2 setInitializers

this.setInitializers(this.getSpringFactoriesInstances(ApplicationContextInitializer.class)); 这里会读取 spring.factories 里面的 相关工厂类，并且执行他们的构造函数。

见 本项目另外一个文档<< 3,spring.factories 启动加载原理 >>里面说明

2.3 setListeners

this.setListeners(this.getSpringFactoriesInstances(ApplicationListener.class)); 设置相关的监听器

3.主方法 run 执行分析

详细源码见下面：

```
public ConfigurableApplicationContext run(String... args) {
        StopWatch stopWatch = new StopWatch();
        stopWatch.start();
        ConfigurableApplicationContext context = null;
        Collection<SpringBootExceptionReporter> exceptionReporters = new ArrayList();
        this.configureHeadlessProperty();
        SpringApplicationRunListeners listeners = this.getRunListeners(args);
        listeners.starting();

        Collection exceptionReporters;
        try {
            ApplicationArguments applicationArguments = new DefaultApplicationArguments(args);
            ConfigurableEnvironment environment = this.prepareEnvironment(listeners, applicationArguments);
            this.configureIgnoreBeanInfo(environment);
            Banner printedBanner = this.printBanner(environment);
            context = this.createApplicationContext();
            exceptionReporters = this.getSpringFactoriesInstances(SpringBootExceptionReporter.class, new
        Class[]{ConfigurableApplicationContext.class}, context);
            this.prepareContext(context, environment, listeners, applicationArguments, printedBanner);
            this.refreshContext(context);
            this.afterRefresh(context, applicationArguments);
            stopWatch.stop();
            if (this.logStartupInfo) {
            (new StartupInfoLogger(this.mainApplicationClass)).logStarted(this.getApplicationLog(),
        stopWatch);
            }
            listeners.started(context);
            this.callRunners(context, applicationArguments);
        } catch (Throwable var10) {
            this.handleRunFailure(context, var10, exceptionReporters, listeners);
            throw new IllegalStateException(var10);
        }

        try {
            listeners.running(context);
            return context;
        } catch (Throwable var9) {
            this.handleRunFailure(context, var9, exceptionReporters, (SpringApplicationRunListeners)null);
            throw new IllegalStateException(var9);
        }
}
```

源代码分析：

3.1，StopWatch

新建一个 StopWatch 后面主要调用里面的 start ，stop ，用来记录执行时间

start 方法： this.startTimeMillis = System.currentTimeMillis();

stop 方法：

```
long lastTime = System.currentTimeMillis() - this.startTimeMillis;
this.totalTimeMillis += lastTime;
```

3.2 configureHeadlessProperty

主要设置 java.awt.headless 属性，源码如下

```
private void configureHeadlessProperty() {
    System.setProperty("java.awt.headless", System.getProperty("java.awt.headless",
Boolean.toString(this.headless)));
}
```

3.3，获取 SpringApplicationRunListeners ，并且执行他们的 starting 方法

相关代码：

```
SpringApplicationRunListeners listeners = this.getRunListeners(args);
listeners.starting();
```

3.4， 调用 SpringApplicationRunListeners 的 prepareEnvironment 方法

```
ConfigurableEnvironment environment = this.prepareEnvironment(listeners, applicationArguments);
```

具体执行代码：

```
private ConfigurableEnvironment prepareEnvironment(SpringApplicationRunListeners listeners,
ApplicationArguments applicationArguments) {
    ConfigurableEnvironment environment = this.getOrCreateEnvironment();
    this.configureEnvironment((ConfigurableEnvironment)environment,
applicationArguments.getSourceArgs());
    listeners.environmentPrepared((ConfigurableEnvironment)environment);
    this.bindToSpringApplication((ConfigurableEnvironment)environment);
    if (this.webApplicationType == WebApplicationType.NONE) {
```

```
            environment = (new
            EnvironmentConverter(this.getClassLoader())).convertToStandardEnvironmentIfNecessary((Configu
        rableEnvironment)environment);
            }

        ConfigurationPropertySources.attach((Environment)environment);
        return (ConfigurableEnvironment)environment;
    }
```

3.5，设置 configureIgnoreBeanInfo

```
            System.setProperty("spring.beaninfo.ignore", ignore.toString());
```

3.6 打印 spring boot banner 图

```
        Banner printedBanner = this.printBanner(environment);
```

提示：我们在自己的项目启动时可以设置 个性化的 banner

3.7，createApplicationContext

主要是创建 ApplicationContext ，实现类为：
AnnotationConfigServletWebServerApplicationContext

3.8，执行 prepareContext

在这里会调用 listeners.contextLoaded

3.9,执行 refreshContext--重点

```
调用 this.refreshContext(context);
AbstractApplicationContext refresh () 核心处理逻辑：
public void refresh() throws BeansException, IllegalStateException {
        Object var1 = this.startupShutdownMonitor;
```

```
        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);
              this.finishRefresh();
        } catch (BeansException var9) {
        if (this.logger.isWarnEnabled()) {
        this.logger.warn("Exception encountered during context initialization - cancelling refresh attempt: "
+ var9);
        }

        this.destroyBeans();
        this.cancelRefresh(var9);
        throw var9;
        } finally {
        this.resetCommonCaches();
        }

}
}
```

3.9.1prepareRefresh

这里会初始化 servletContextInitParams，并且验证格式

```
        this.initPropertySources();
        this.getEnvironment().validateRequiredProperties();
```

初始化 property 具体执行的代码为：StandardServletEnvironment 类的 initPropertySources()方法

```
        WebApplicationContextUtils.initServletPropertySources(this.getPropertySources(),
servletContext, servletConfig);
```

3.9.2 prepareBeanFactory

主要是注册一些 bean

3.9.3 initApplicationEventMulticaster

初始化 ApplicationEvent 广播器，后面会广播 ApplicationEvent 事件

3.9.4 initMessageSource

初始化国际化文件 ，涉及到资源的国际化处理

3.9.5 onRefresh

其中 onRefresh 调用的子类 ServletWebServerApplicationContext 的方法，具体为：

```
protected void onRefresh() {
      super.onRefresh();

      try {
            this.createWebServer();
      } catch (Throwable var2) {
            throw new ApplicationContextException("Unable to start web server", var2);
       }
}
```

可以看出这里会执行 createWebServer ，在里面会执行 onStartup 具体处理逻辑：

```
private void createWebServer() {
      WebServer webServer = this.webServer;
      ServletContext servletContext = this.getServletContext();
      if (webServer == null && servletContext == null) {
            ServletWebServerFactory factory = this.getWebServerFactory();
            this.webServer = factory.getWebServer(new
      ServletContextInitializer[]{this.getSelfInitializer()});
      } else if (servletContext != null) {
            try {
                  this.getSelfInitializer().onStartup(servletContext);
            } catch (ServletException var4) {
                  throw new ApplicationContextException("Cannot initialize servlet context", var4);
            }
      }
      this.initPropertySources();
```

```
}
```

上面代码中先创建 tomcat 工厂 TomcatServletWebServerFactory ，具体执行代码为：

```
public WebServer getWebServer(ServletContextInitializer... initializers) {
      Tomcat tomcat = new Tomcat();
      File baseDir = this.baseDirectory != null ? this.baseDirectory : this.createTempDir("tomcat");
      tomcat.setBaseDir(baseDir.getAbsolutePath());
      Connector connector = new Connector(this.protocol);
      tomcat.getService().addConnector(connector);
      this.customizeConnector(connector);
      tomcat.setConnector(connector);
      tomcat.getHost().setAutoDeploy(false);
      this.configureEngine(tomcat.getEngine());
      Iterator var5 = this.additionalTomcatConnectors.iterator();

      while(var5.hasNext()) {
            Connector additionalConnector = (Connector)var5.next();
            tomcat.getService().addConnector(additionalConnector);
      }

      this.prepareContext(tomcat.getHost(), initializers);
      return this.getTomcatWebServer(tomcat);
}
```

最后调用 构造函数创建 TomcatWebServer，执行里面的 initialize 方法。

```
public TomcatWebServer(Tomcat tomcat, boolean autoStart) {
      this.monitor = new Object();
      this.serviceConnectors = new HashMap();
      Assert.notNull(tomcat, "Tomcat Server must not be null");
      this.tomcat = tomcat;
      this.autoStart = autoStart;
      this.initialize();
}
```

在 initialize 方法里面 会调用 this.tomcat.start();

3.10 finishRefresh

会 发布一个 ContextRefreshedEvent 的event。

3.11 registerListeners 执行逻辑：

（1）添加监听器 addApplicationListener

（2）添加监听器 bean addApplicationListenerBean

（3）广播 ApplicationEvent

```
ApplicationEvent earlyEvent = (ApplicationEvent)var9.next();
this.getApplicationEventMulticaster().multicastEvent(earlyEvent);
```

3.12，执行 afterRefresh，目前暂未实现

3.13， 调用 SpringApplicationRunListeners 中的 started

```
listeners.started(context);
```

3.14 执行 callRunners

```
this.callRunners(context, applicationArguments);
```

里面有 2 种类型的 runner ，一个是 ApplicationRunner 另外一个是 commandLineRunner ，执行他们的 run 方法。。

3.15，执行 SpringApplicationRunListeners 中的 running

```
listeners.running(context);
```

至此 spring boot 启动完成

阿伦读源码 allengent@163.com