# SpringApplication.run()方法

最终返回的是ConfigurableApplicationContext

主要有两个步骤，首先创建SpringApplicaiton对象，然后再调用run方法。

## stopWatch.start();

计时工具，直到stopWatch.stop();为一个任务，调用stopWatch.prettyPrint();即可返回所有任务的执行情况，包括执行时间。里面的方法挺多。

可以多次启动和关闭。每一次启动和关闭就是一个任务的时间。

## SpringApplicationRunListeners listeners = getRunListeners(args);

## listeners.starting();

## createApplicationContext()

创建ConfigurableApplicationContext

通过Class.forName(类路径)获取下面这个类的对象，

AnnotationConfigServletWebServerApplicationContext这个类对象。

BeanUtils.instantiateClass对这个类的对象处理。

(ConfigurableApplicationContext) BeanUtils.*instantiateClass*(contextClass);

将这个对象返回。

## getSpringFactoriesInstances

## prepareContext()

做context的准备工作，主要是对容器做一些预设置。

参数：context, environment, listeners, applicationArguments,

printedBanner

### postProcessApplicationContext(context)

向beanFactory中添加了一个beanNameGenerator

用来生成扫描到的Bean在容器中的名字。

### applyInitializers(context)

调用ApplicationContextInitializer中的initialize(context)方法.

ApplicationContextInitializer是一个回调接口，它会在ConfigurableApplicationContext的refresh()方法调用之前被调用,做一些容器的初始化工作

### listeners.contextPrepared(context);

## refreshContext(context);

调用refresh(context);方法。

### Refresh(context)

调用AbstractApplicationContext中的refresh方法。到这里就和基于xml配置到了一个地方。Xml的方式也是到了这里开始最重要的处理。

## afterRefresh(context, applicationArguments);

## stopWatch.stop();

## listeners.started(context);

## callRunners(context, applicationArguments);

## listeners.running(context);