第01课：SpringBoot 入门

Spring 与 SpringBoot 目前在 Java 项目中基本是标配了，极大简化了开发的时间和难度。尤其是 SpringBoot，各种框架整合 SpringBoot 都非常方便。在享受完方便的开发后，是时候来学习下背后的机制了。

对于框架的学习，学会使用只是入门的第一步，掌握其工作原理这才是需要更加深入学习的，技术类框架都是大同小异的，只要掌握了一种框架，这样即使再多类似的框架，也是非常容易上手使用的。

SpringBoot 可以非常方便的的集成各种框架，比如整合 Web 环境只需要导入对应的依赖：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

启动 SpringBoot 项目只需要在对应的入口类中添加注解 @SpringBootApplication()：

```
@SpringBootApplication()
public class Application {
    public static void main(String[] args) throws Exception {
        SpringApplication.run(Application.class, args);
    }
}
```

这样一个 SpringBoot 项目就启动了，非常简单，一行代码就解决了。当项目中需要获取配置文件时，这时 @ConfigurationProperties 就派上用场了，只需要在对应的 Bean 上添加 @ConfigurationProperties 并配置固定的前缀就可以将对应的属性注入到 Bean 中，比如：

```
@Component
@ConfigurationProperties(prefix = "myconfig")
public class MyConfig {
    private String name;
    private Integer age;
    private String desc;
    //get/set省略
    @Override
    public String toString() {
        return "MyConfig [name=" + name + ", age=" + age + ", desc=" + desc + "]";
    }
}
```

在配置文件中添加配置：

```
myconfig.name=test
myconfig.age=22
myconfig.desc=这是我的测试描述
```

这样项目启动后，MyConfig 对象就自动注入了对应的属性值。测试代码：

```
@SpringBootApplication()
public class Application {
    public static void main(String[] args) throws Exception {
        ConfigurableApplicationContext application = SpringApplication.run(Applica
tion.class, args);
        MyConfig config = application.getBean(MyConfig.class);
        System.out.println(config);
        application.close();
    }
}
```

输出：

```
MyConfig [name=test, age=22, desc=这是我的测试描述]
```

SpringBoot 还提供了多种 EnableXX 注解，每种注解对应实现了不同的功能，使用时只需要在入口类中添加对应的注解就可以了，比如：

```
@EnableAspectJAutoProxy
@EnableConfigurationProperties
@EnableCaching
@EnableWebMvc
```

这样就非常轻松的整合 cache、aspectj、MVC 配置、属性配置的功能。那么是否想过这些注解是如何实现对应的功能了？中间用到了什么原理呢？为什么有的需要添加注解，有的直接引入依赖就可以运行了呢？

本次课程将以理论为基础、实践并行的方式讲解 Spring 与 SpringBoot 常用的功能原理。让读者更快更好更轻松的掌握其使用方法的同时掌握原理。

本次课程的内容将分为一下几个模块进行，逐渐深入的方式讲解 SpringBoot 框架的执行机制。

- 如何在 Bean 初始化回调前后进行自定义操作

本章将讲解如何在 Bean 初始化回调前后进行自定义操作，什么时候 InitializingBean 回调，什么时候 @PostConstruct 会得到执行，如果不想要使用注解和添加实现接口时，如何在 Bean 初始化回调前后进行自定义操作。

- @ConfigurationProperties 实现原理与实战

通过上一篇的引入，以实践的方式解析 @ConfigurationProperties 工作机制和原理，掌握以后会感觉其他注

解都是套路般的实现。

- Spring 各种 Aware 注入的原理与实战

讲解 Spring 中提供的各种 Aware 接口工作原理与自定义 Aware 实战。

- @EnableAutoConfiguration 原理实战

SpringBoot 中所有的 AutoConfiguration 框架都需要依赖于 @EnableAutoConfiguration 注解，如果没有此注解，那么 SpringBoot 将不在有那么多的便利性，本文将从源码的角度剖析 @EnableAutoConfiguration 工作原理，讲解为什么有的框架需要添加对应的 Enable 注解，有的框架为什么只要引入对应的依赖就可以了。

- Spring 中 Bean 扫描实战

在框架类型的项目中，对 Bean 的扫描是必不可少的，本文将从实践的方式讲解如何扫描出我们需要的 Bean，如何对接口和抽象类进行扫描，如何脱离出 Spring 注解体系对 Bean 进行扫描，为下一篇文章中实现自己的 Enable* 框架打下基础。

- `Enable*` 框架实现原理与实战

通过《@EnableAutoConfiguration 原理实战》一文，掌握了 @EnableAutoConfiguration 的工作原理，以 MyBatis 整合源码进行分析，对 SpringBoot 提供的各种 Enable* 框架进行细致分析与自定义实现，以及学习如何使用 Spring 的工厂 Bean 与工厂 Bean 的 JDK 动态代理方式实现和 CGlib 方式实现。

通过以上几篇文章的学习后，想必读者应该已经掌握了 SpringBoot 常用注解的工作原理，将来在项目中需要对项目进行设计开发时，这些知识必然会派上用场的。

之所以写这一系列教程，主要是因为在查看 SpringBoot 相关注解源码时，发现了非常多实用的类和实现方式，这些实现方式将会在后续的项目中起到非常重要的作用，可以帮助读者设计出更好的架构。这些知识之所以共享出来，是为需要学习的开发者提供一个导向，帮助他们更好的吸收 SpringBoot 与 Spring 框架。

在开发越来越简便的今天，可以非常快速地入门一个新的框架，但如果只是停留在使用的角度上，那么相对于新手程序员有什么优势呢？同样的使用方式，大家都会，其竞争力在哪里？在和其他开发者或和项目经理解释项目技术的时候，难道还是满口代码么？而且作为老员工，当给新手开发解说项目框架如何使用时，被一句“为什么这么用？”就卡住了，是不是很尴尬。

作为开发人员，想要提升自身能力的一个非常好的方式就是看源码，在源码中能找到非常多有用的实现原理和思路，可以学习到大牛架构师是如何编码、设计架构的。最重要的是各种框架的原理都是在源码中体现的。当学某一个框架的时候，首先要了解它的原理，知道大概流程是做什么的，对后续的开发和学习都是非常有帮助的。了解原理后就需要继续深入学习了，通过看源码是非常好的一种方式，可以了解到一个框架的精髓。而且类似框架的原理都是想通的，只要掌握了一种框架，对于其他类似的框架是非常容易上手并理解的。

任何语言和框架，学习其原理都是必须掌握的。只有掌握了其中原理才能说个所以然来，才能更好的解决和避免项目中的各种坑。这就是为什么面试时面试官喜欢问 Java 底层原理和 JVM 实现原理。

本课程中所有涉及到的代码都是基于 SpringBoot 1.5.8.Release 版本来写的。