第1-2课：Spring Boot 2.0 都更新了什么（下）

彩蛋

Spring Boot 2.0 支持了动态 Gif 的启动 Logo 打印。

在 Spring Boot 1.0 项目中 src/main/resources 路径下新建一个 banner.txt 文件，文件中写入一些字符，启动项目时就会发现默认的 Banner 被替换了，到了 Spring Boot 2.0 现在可以支持 Gif 文件的打印，Spring Boot 2.0 在项目启动的时候，会将 Gif 图片的每一个画面，按照顺序打印在日志中，所有的画面打印完毕后，才会启动 Spring Boot 项目。

项目的启动 Banner 有什么用呢，在一些大的组织或者公司中，可以利用这个特性定制自己专属的启动画面，增加团队对品牌的认同感。

1.0 升级 2.0 API 变化

从 Spring Boot 1.0 升级到 2.0 之后，有很多的 API 已经过时，在使用的时候需要注意。

启动类 SpringBootServletInitializer

Spring Boot 部署到 Tomcat 中去启动时需要在启动类添加 SpringBootServletInitializer，2.0 和 1.0 有区别。

```
// 1.0
import org.springframework.boot.web.support.SpringBootServletInitializer;
// 2.0
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class UserManageApplication extends SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder applicat
ion) {
        return application.sources(UserManageApplication.class);
    }

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

Spring Boot 2.0 默认不包含 log4j，建议使用 slf4j

```
import org.apache.log4j.Logger;
protected Logger logger = Logger.getLogger(this.getClass());
```

改为：

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
protected Logger logger =  LoggerFactory.getLogger(this.getClass());
```

Thymeleaf 3.0 默认不包含布局模块

Spring Boot 2.0 中 spring-boot-starter-thymeleaf 包默认并不包含布局模块，使用 Thymeleaf 对页面进行布局时，需要单独添加布局模块，如下：

```
<dependency>
   <groupId>nz.net.ultraq.thymeleaf</groupId>
   <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>
```

layout:decorator 标签在 3.0 过期，推荐使用新的标签 layout:decorate 进行页面布局。

配置文件

大量的 Servlet 专属的 server.* 被移到了 server.servlet.* 下：

Old property	New property
server.context-parameters.*	server.servlet.context-parameters.*
server.context-path	server.servlet.context-path
server.jsp.class-name	server.servlet.jsp.class-name
server.jsp.init-parameters.*	server.servlet.jsp.init-parameters.*
server.jsp.registered	server.servlet.jsp.registered
server.servlet-path	server.servlet.path

原 spring.http.* 或 spring.server.* 下的一些参数，例如我用到了文件上传参数，已修改为 sping.servlet.multipart 下。

WebMvcConfigurerAdapter 过期

Spring Boot 2.0 中将原来的 WebMvcConfigurerAdapter 替换为 WebMvcConfigurer。

1.0 中的用法：

```
public class MyWebMvcConfigurerAdapter extends WebMvcConfigurerAdapter
```

2.0 中的用法：

```
public class MyWebMvcConfigurerAdapter implements WebMvcConfigurer
```

Spring Boot JPA 变化

去掉了 xxxOne() 方法

以前的 findOne() 方法其实就是根据传入的 ID 来查找对象，所以在 Spring Boot 2.0 的 Repository 中我们可以添加 findById(long id) 来替换使用。

例如：

```
User user=userRepository.findOne(Long id)
```

改为手动在 userRepository 手动添加 findById(long id) 方法，使用时将 findOne() 调用改为 findById(long id)：

```
User user=userRepository.findById(long id)
```

delete() 方法和 findOne() 类似也被去掉了，可以使用 deleteById(Long id) 来替换，还有一个不同点是 deleteById(Long id) 默认实现返回值为 void。

```
Long deleteById(Long id);
```

改为：

```
//delete 改为 void 类型
void deleteById(Long id);
```

当然我们还有一种方案可以解决上述的两种变化，就是自定义 SQL（如下），但是没有上述方案简单，不建议使用。

```
@Query("select t from Tag t where t.tagId = :tagId")
Tag getByTagId(@Param("tagId") long tagId);
```

需要指定主键的自增策略

Spring Boot 2.0 需要指定主键的自增策略，这个和 Spring Boot 1.0 有所区别，1.0 会使用默认的策略，如果不指定自增策略会报错。

```
@Id
@GeneratedValue(strategy= GenerationType.IDENTITY)
private long id;
```

分页组件 PageRequest 变化

在 Spring Boot 2.0 中，方法 new PageRequest(page, size, sort) 已经过期不再推荐使用，推荐使用以下方式来构建分页信息：

```
Pageable pageable =PageRequest.of(page, size, Sort.by(Sort.Direction.ASC,"id"));
```

跟踪了一下源码发现 PageRequest.of() 方法，内部还是使用的 new PageRequest(page, size, sort)，只是最新的写法更简洁一些。

```
public static PageRequest of(int page, int size, Sort sort) {
    return new PageRequest(page, size, sort);
}
```

JPA 关联查询

在使用 Spring Boot 1.0 时，使用 JPA 关联查询时我们会构建一个接口对象来接收结果集，类似如下：

```
public interface CollectView{
   Long getId();
   Long getUserId();
   String getProfilePicture();
   String getTitle();
}
```

在使用 Spring Boot 1.0 时，如果没有查询到对应的字段会返回空，在 Spring Boot 2.0 中会直接报空指针异常，对结果集的检查会更加严格一些。

> 这只是目前升级过程中发现的一些问题，不代表 Spring Boot 2.0 升级中的所有问题，在随后的课程中会再一一介绍。

是否选择升级

通过以上内容可以看出 Spring Boot 2.0 相对于 1.0 增加了很多新特性，并且最重要的是 Spring Boot 2.0 依赖的 JDK 最低版本是 1.8，估计国内大多互联网公司还没有这么激进。另外一个新的重大版本更新之后，难免会有一些小 Bug 什么的，往往需要再发布几个小版本之后，才会慢慢稳定下来。

因此我的建议是，如果不是特别需要使用 Spring Boot 2.0 上面提到的新特性，就尽量不要着急进行升级，等 Spring Boot 2.0 彻底稳定下来后再使用。如果想要升级也请先从早期的版本升级到 Spring Boot 1.5X 系列之后，再升级到 Spring Boot 2.0 版本，Spring Boot 2.0 的很多配置内容和 Spring Boot 1.0 不一致需要注意。

Spring Boot 1.0 发布之后给我们带来了全新的开发模式，Spring Boot 2.0 发布标志着 Spring Boot 已经走向成熟，对 Java 领域带来的变革已经开启！

总结

可以看出 Spring Boot 2.0 是历时 4 年开发出来的巨作，在 Spring Boot 1.0 的基础上进行了大量的优化，淘汰了很多过期的 API，同时引入了一大批最新的技术，这些新技术在未来的一段时间内都具有引导性。在我们学习 Spring Boot 2.0 的同时，需要同时学习 Spring Boot 2.0 引入的一些新技术，不建议大家在生产环境直接进行升级，等 Spring Boot 2.0 进一步稳定之后再升级替换。