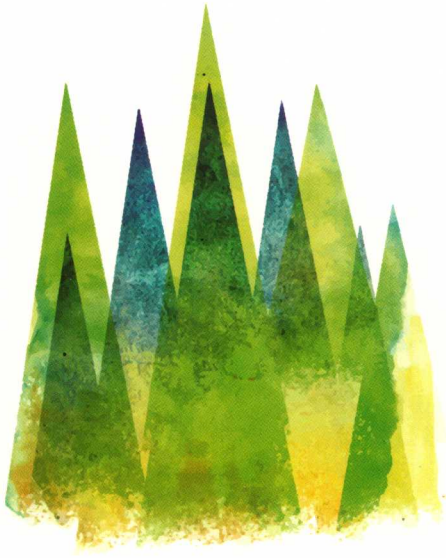
Broadview®
www.broadview.com.cn

包含Spring Security Java配置、Spring Security安全防护和源码导读
详细讲解OAuth2实战，并简单剖析部分OAuth2核心源码



Spring Security实战

陈木鑫◎编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Spring Security实战

陈木鑫◎编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

Spring Security 是一个强大且高度可定制的安全框架，致力于为 Java 应用提供身份认证和授权。

本书通过 4 部分内容由浅入深地介绍了 Spring Security 的方方面面。第 1 部分主要讲解 Spring Security 的基本配置；第 2 部分剖析 Web 项目可能遇到的安全问题，并讲解如何使用 Spring Security 进行有效防护；第 3 部分详细介绍 OAuth，并使用 Spring Social 整合 Spring Security，实现 QQ 快捷登录；第 4 部分重点介绍 Spring Security OAuth 框架，剖析 Spring Security OAuth 的部分核心源码。

本书适合有一定 Java 基础的读者，以及希望在项目中应用 Spring Security 的开发人员阅读。

图书在版编目（CIP）数据

Spring Security 实战 / 陈木鑫编著. —北京：电子工业出版社，2019.8
ISBN 978-7-121-37143-1

Ⅰ. ①S… Ⅱ. ①陈… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2019）第 152390 号

责任编辑：安　娜
印　　刷：天津千鹤文化传播有限公司
装　　订：天津千鹤文化传播有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：17.75　字数：348 千字
版　　次：2019 年 8 月第 1 版
印　　次：2019 年 8 月第 1 次印刷
定　　价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　言

Spring Security 的前身是 Acegi Security，在被收纳为 Spring 子项目后正式更名为 Spring Security。在笔者成书时，Spring Security 已经升级到 5.1.3.RELEASE 版本，不仅新增了原生 OAuth 框架，还支持更加现代化的密码加密方式。可以预见，在 Java 应用安全领域，Spring Security 会成为首先被推崇的安全解决方案。

虽然 Spring Security 有强大的功能，但它同时也有很高的学习成本。它囊括了身份认证的各种应用场景以及 Web 安全的大量知识，仅官方参考手册就有数十万字，并且还省略了诸多实现细节。许多开发人员在面对这样的“庞然大物”时无从入手，更因为对其不够了解而在实际项目中不敢轻易采用。

本书由浅入深、抽丝剥茧地讲解了 Spring Security 的典型应用场景，另外，还分析了部分核心源码，以及许多开发语言之外的安全知识。通过本书，读者不仅可以学习如何应用 Spring Security，还可以学习借鉴它的实现思路，以将这种实现思路应用到其他开发场景中。

本书读者

本书主要面向有一定 Java 基础的读者，以及希望在实际项目中应用 Spring Security 的开发人员。

本书内容

本书共分为 4 个部分。

第 1 部分（第 1 章至第 3 章）主要介绍 Spring Security 的基本配置，包括默认配置、简单表单认证，以及基于数据库模型的认证与授权。

第 2 部分（第 4 章至第 11 章）主要介绍各种定制化的配置场景，剖析 Web 项目可能遇到

的安全问题，并讲解如何使用 Spring Security 进行有效防护，部分章节还配备了详细的源码导读。

第 3 部分（第 13 章）将登录用户的数据来源从系统内转移到社交平台，详细介绍了 OAuth，并使用 Spring Social 整合 Spring Security，实现 QQ 快捷登录，满足一般性的项目需求。

第 4 部分（第 14 章）带领读者认识 Spring Security OAuth 框架，并基于该框架完整实现了 OAuth 客户端、OAuth 授权服务器以及 OAuth 资源服务器三种角色。除此之外，还简单剖析了 Spring Security OAuth 的部分核心源码，以帮助读者更好地理解 OAuth 框架。

致谢

首先感谢赵召同学（Andy）对第 4 部分的贡献。笔者在 Gitter 发言表明会写一本关于 Spring Security 的中文书后，赵召同学找到了我，并希望与我一起写作，但由于这本书实际上已基本成型，所以赵召同学贡献了第 4 部分，使得本书内容更加充实，再次感谢赵召同学的贡献。

其次也非常感谢博文视点公司的安娜编辑以及身边的朋友给予的鼓励。从下定决心编写这本书开始，笔者其实经历了非常多的折磨，不管是思路的枯竭还是耐心的消磨，都致使笔者几次三番萌生退意，但最终还是在不断的鼓励声中坚持了下来，成功为国内希望学习 Spring Security 的朋友奉上了一本中文版的教程，这份收获也应当属于他们。

陈木鑫

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- ◎ **提交勘误：**您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- ◎ **与读者交流：**在页面下方【读者评论】处留下您的疑问或观点，与其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/37143

目　录

第 1 部分

第 2 部分

第 3 部分

第 4 部分

第 1 部分

第1章
初识Spring Security

本书所有的示例都基于 Intellij IDEA 创建的 Spring Boot 项目，因此读者需要具备一定的 Spring 相关知识。

1.1 Spring Security 简介

Spring Security 的前身是 Acegi Security，在被收纳为 Spring 子项目后正式更名为 Spring Security。

在笔者成书时，Spring Security 已经升级到 5.1.3.RELEASE 版本，加入了原生 OAuth2.0 框架，支持更加现代化的密码加密方式。可以预见，在 Java 应用安全领域，Spring Security 会成为被首先推崇的解决方案，就像我们看到服务器就会联想到 Linux 一样顺理成章。

应用程序的安全性通常体现在两个方面：认证和授权。

认证是确认某主体在某系统中是否合法、可用的过程。这里的主体既可以是登录系统的用户，也可以是接入的设备或者其他系统。

授权是指当主体通过认证之后，是否允许其执行某项操作的过程。

这些概念并非 Spring Security 独有，而是应用安全的基本关注点。Spring Security 可以帮助我们更便捷地完成认证和授权。

Spring Security 支持广泛的认证技术，这些认证技术大多由第三方或相关标准组织开发。Spring Security 已经集成的认证技术如下：

◎ HTTP BASIC authentication headers：一个基于 IETF RFC 的标准。

- HTTP Digest authentication headers：一个基于 IETF RFC 的标准。
- HTTP X.509 client certificate exchange：一个基于 IETF RFC 的标准。
- LDAP：一种常见的跨平台身份验证方式。
- Form-based authentication：用于简单的用户界面需求。
- OpenID authentication：一种去中心化的身份认证方式。
- Authentication based on pre-established request headers：类似于 Computer Associates SiteMinder，一种用户身份验证及授权的集中式安全基础方案。
- Jasig Central Authentication Service：单点登录方案。
- Transparent authentication context propagation for Remote Method Invocation (RMI) and HttpInvoker：一个 Spring 远程调用协议。
- Automatic "remember-me" authentication：允许在指定到期时间前自行重新登录系统。
- Anonymous authentication：允许匿名用户使用特定的身份安全访问资源。
- Run-as authentication：允许在一个会话中变换用户身份的机制。
- Java Authentication and Authorization Service：JAAS，Java 验证和授权 API。
- Java EE container authentication：允许系统继续使用容器管理这种身份验证方式。
- Kerberos：一种使用对称密钥机制，允许客户端与服务器相互确认身份的认证协议。

除此之外，Spring Security 还引入了一些第三方包，用于支持更多的认证技术，如 JOSSO 等。如果所有这些技术都无法满足需求，则 Spring Security 允许我们编写自己的认证技术。因此，在绝大部分情况下，当我们有 Java 应用安全方面的需求时，选择 Spring Security 往往是正确而有效的。

Internet 工程任务组（Internet Engineering Task Force，IETF）是推动 Internet 标准规范制定的最主要的组织。请求注解（Request For Comments，RFC）包含大多数关于 Internet 的重要文字资料，被称为“网络知识圣经”。

在授权上，Spring Security 不仅支持基于 URL 对 Web 的请求授权，还支持方法访问授权、对象访问授权等，基本涵盖常见的大部分授权场景。

很多时候，一个系统的安全性完全取决于系统开发人员的安全意识。例如，在我们从未听过 SQL 注入时，如何意识到要对 SQL 注入做防护？关于 Web 系统安全的攻击方式非常多，诸如 XSS、CSRF 等，未来还会暴露出更多的攻击方式，我们只有在充分了解其攻击原理后，才能提出完善而有效的防护策略。在笔者看来，学习 Spring Security 并非局限于降低 Java 应用的

安全开发成本，通过 Spring Security 了解常见的安全攻击手段以及对应的防护方法也尤为重要，这些是脱离具体开发语言而存在的。

1.2 创建一个简单的 Spring Security 项目

本节创建一个简单的 Spring Security 项目，带领大家初步领略 Spring Security 带来的便利。下面我们就完整地“走”一遍创建项目的流程。

通过 Intellij IDEA 创建 Spring Boot 项目的方式有许多种，其中最简单的方式就是使用 Spring Initializr 工具，省略了在 https://start.spring.io 中生成并导入 Intellij IDEA 的过程。Eclipse 也提供了一个可以实现类似功能的插件：STS（Spring Tool Suite），感兴趣的读者可以自行了解。此处我们单击“Next”按钮进入下一个页面，如图 1-1 所示。

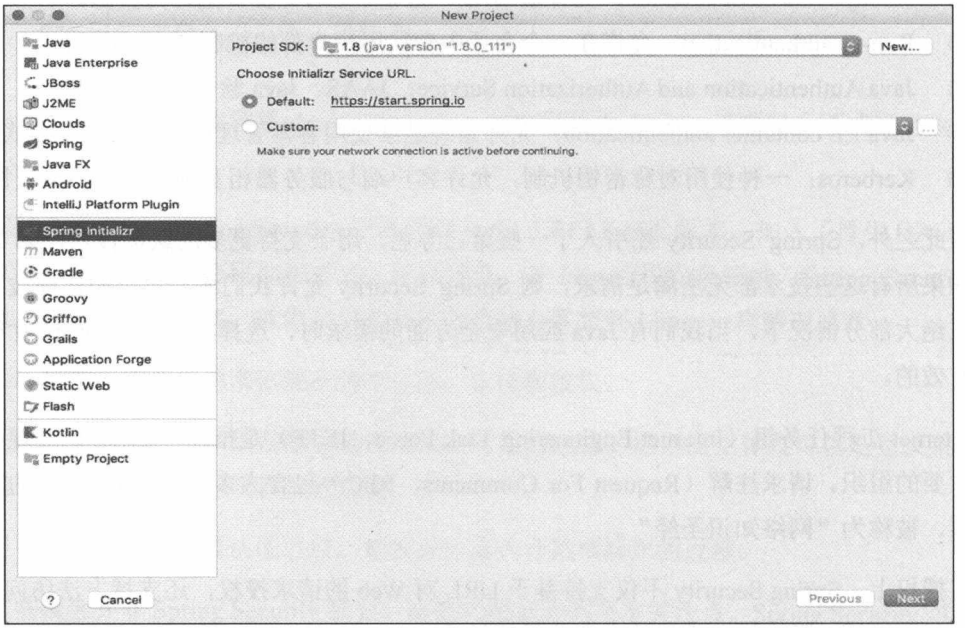


图 1-1

在图 1-2 所示页面中，除可以设置项目的 Group、Artifact 这些基本信息外，还有其他几个配置可选。例如，对于 Type 属性，可以选择 Maven Project 或者 Gradle Project 作为项目管理工具；对于 Language 属性，可以选择使用 Java、Kotlin 或 Groovy 作为开发语言。

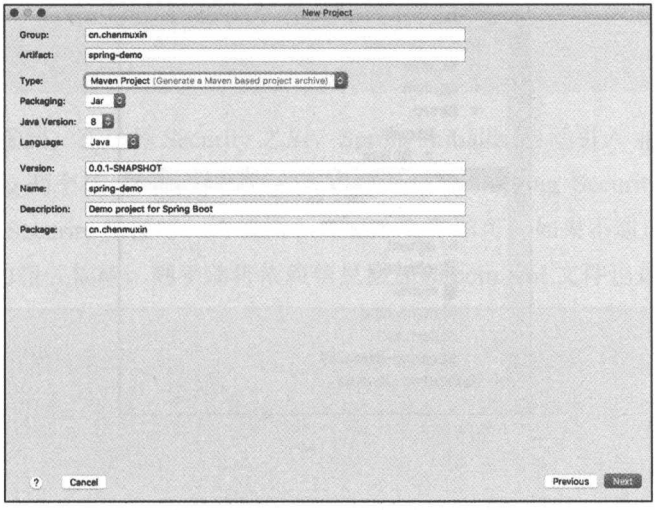


图 1-2

Spring Initializr 将根据我们的选择自动构建项目骨架，选好之后单击“Next”按钮进入下一个页面，如图 1-3 所示。

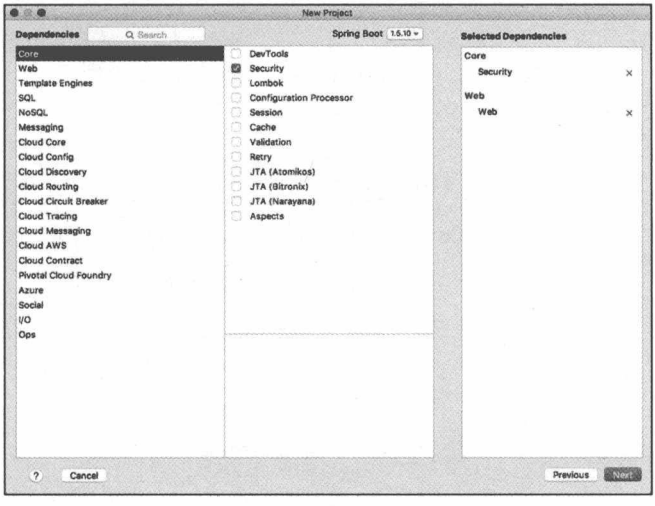


图 1-3

Spring Initializr 允许我们提前选定一些常用的项目依赖，此处我们选择 Security 作为构建 Spring Security 项目的最小依赖，选择 Web 作为 Spring Boot 构建 Web 应用的核心依赖。

当项目创建完成后，可以得到如图 1-4 所示的目录结构。

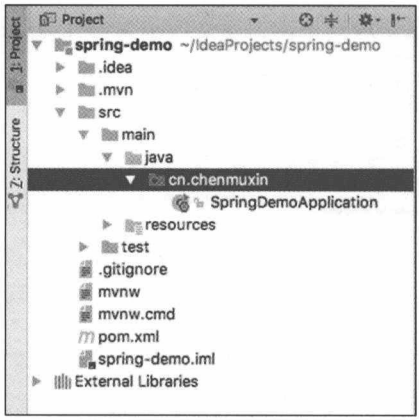


图 1-4

打开 pom.xml 文件，看看 Spring Initializr 引入了哪些依赖。

```
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-aop</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-config</artifactId>
                <exclusions>
                        <exclusion>
                                <groupId>aopalliance</groupId>
                                <artifactId>aopalliance</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
        <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-web</artifactId>
                <exclusions>
                        <exclusion>
                                <groupId>aopalliance</groupId>
                                <artifactId>aopalliance</artifactId>
                        </exclusion>
```

```
                </exclusions>
        </dependency>
</dependencies>
```

从代码中可以看到，在选择 Security 之后，Spring Initializr 自动引入 spring-security-web 和 spring-security-config 两个核心模块，这正是官方建议引入的 Spring Security 最小依赖。当需要引入更多的 Spring Security 特征时，再编辑 pom.xml 文件即可。如果不通过 Spring Initializr 添加 Spring Security 的相关依赖，则手动将依赖信息添加到 pom.xml 文件也是可以的。

```
<dependencies>
    <!-- ... 其他依赖 ... -->
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-web</artifactId>
        <version>4.2.4.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
        <version>4.2.4.RELEASE</version>
    </dependency>
</dependencies>
```

同理，通过 Gradle 管理的项目只需引入 spring-security-web 和 spring-security-config 两个核心模块。

```
dependencies {
        compile
'org.springframework.security:spring-security-web:4.2.4.RELEASE'
        compile
'org.springframework.security:spring-security-config:4.2.4.RELEASE'
}
```

下面打开程序的入口类 SpringDemoApplication，声明一个 hello 路由。

```
@RestController
@SpringBootApplication
public class SpringDemoApplication {

        @GetMapping("/")
        public String hello() {
                return "hello, spring security";
        }
```