# Spring Security学习文档

## Spring Security简介

layers mentioned above, together with

managerial factors that encompass every layer. A non-exhaustive list of such managerial factors would

include security bulletin monitoring, patching, personnel vetting, audits, change control, engineering

management systems, data backup, disaster recovery, performance benchmarking, load monitoring,

centralised logging, incident response procedures etc.

With Spring Security being focused on helping you with the enterprise application security layer, you will

find that there are as many different requirements as there are business problem domains. A banking

application has different needs from an ecommerce application. An ecommerce application has different

needs from a corporate sales force automation tool. These custom requirements make application

security interesting, challenging and rewarding.

Please read Part II, “Getting Started”, in its entirety to begin with. This will introduce you to the framework

and the namespace-based configuration system with which you can get up and running quite quickly.

To get more of an understanding of how Spring Security works, and some of the classes you might need

to use, you should then read Part III, “Architecture and Implementation”. The remaining parts of this

guide are structured in a more traditional reference style, designed to be read on an as-required basis.

We’d also recommend that you read up as much as possible on application security issues in general.

Spring Security is not a panacea which will solve all security issues. It is important that the application

is designed with security in mind from the start. Attempting to retrofit it is not a good idea. In particular,

if you are building a web application, you should be aware of the many potential vulnerabilities such

as cross-site scripting, request-forgery and session-hijacking which you should be taking into account

from the start. The OWASP web site (http://www.owasp.org/) maintains a top ten list of web application

vulnerabilities as well as a lot of useful reference information.

We hope that you find this reference guide useful, and we welcome your feedback and suggestions.

Finally, welcome to the Spring Security community.

以上是Spring官方的描述！

下面我们看下万能的百度是如何解释的：

**Spring Security是一个能够为基于Spring的企业应用系统提供声明式的安全访问控制解决方案的安全框架。它提供了一组可以在Spring应用上下文中配置的Bean，充分利用了Spring IoC，DI（控制反转Inversion of Control ,DI:Dependency Injection 依赖注入）和AOP（面向切面编程）功能，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作。**

简单来说，Spring Security就是提供了一套安全框架，供我们在项目中使用，仅此而已！

## 开始使用吧

### 添加Spring Security Maven支持，当然了，必须有Spring Core的支持先；

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-acl</artifactId>

<version>3.2.6.RELEASE</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>3.2.6.RELEASE</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-core</artifactId>

<version>3.2.6.RELEASE</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>3.2.6.RELEASE</version>

<scope>compile</scope>

</dependency>

### web.xml添加Spring Security支持

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<listener>

<listener-class>org.springframework.security.web.session.HttpSessionEventPublisher</listener-class>

</listener>

过滤器springSecurityFilterChain作用：启用Spring Security，**名字不能更改**！

监听器的作用：监听并销毁退出用户的session;

web.xml中的配置完成！

### applicationContext-security.xml配置

<?xml version="1.0" encoding="UTF-8"?>

<beans:beans **xmlns=**"http://www.springframework.org/schema/security"

xmlns:beans="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/security

http://www.springframework.org/schema/security/spring-security-3.2.xsd">

</beans>

我们以 security 替换通常的 beans 作为默认使用标签头！

#### http配置

<http auto-config="true">

<intercept-url pattern="/\*\*" access="any"/>

</http>

注：auto-config="true" 等于如下配置

<http>

<form-login/>

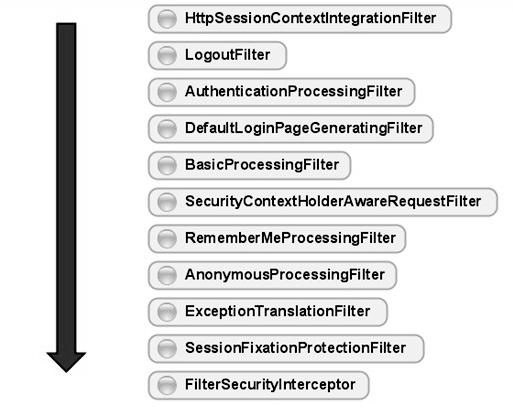
<http-basic/>

<logout/>

<intercept-url pattern="/\*\*" access="any"/>

</http>

当使用 auto-config="true" 时，使用的11个过滤器，但要在项目中真正使用，默认配置还是太单薄，不能满足需求！



如下是Spring Security 过滤器链，我们在使用中不会全部关注，只会关注或替换其中几个即可！

<xs:enumeration value="FIRST"/>

<xs:enumeration value="CHANNEL\_FILTER"/>

<xs:enumeration value="SECURITY\_CONTEXT\_FILTER"/>

<xs:enumeration value="CONCURRENT\_SESSION\_FILTER"/>

<xs:enumeration value="WEB\_ASYNC\_MANAGER\_FILTER"/>

<xs:enumeration value="HEADERS\_FILTER"/>

<xs:enumeration value="CSRF\_FILTER"/>

<xs:enumeration value="LOGOUT\_FILTER"/>

<xs:enumeration value="X509\_FILTER"/>

<xs:enumeration value="PRE\_AUTH\_FILTER"/>

<xs:enumeration value="CAS\_FILTER"/>

<xs:enumeration value="FORM\_LOGIN\_FILTER"/>

<xs:enumeration value="OPENID\_FILTER"/>

<xs:enumeration value="LOGIN\_PAGE\_FILTER"/>

<xs:enumeration value="DIGEST\_AUTH\_FILTER"/>

<xs:enumeration value="BASIC\_AUTH\_FILTER"/>

<xs:enumeration value="REQUEST\_CACHE\_FILTER"/>

<xs:enumeration value="SERVLET\_API\_SUPPORT\_FILTER"/>

<xs:enumeration value="JAAS\_API\_SUPPORT\_FILTER"/>

<xs:enumeration value="REMEMBER\_ME\_FILTER"/>

<xs:enumeration value="ANONYMOUS\_FILTER"/>

<xs:enumeration value="SESSION\_MANAGEMENT\_FILTER"/>

<xs:enumeration value="EXCEPTION\_TRANSLATION\_FILTER"/>

<xs:enumeration value="FILTER\_SECURITY\_INTERCEPTOR"/>

<xs:enumeration value="SWITCH\_USER\_FILTER"/>

<xs:enumeration value="LAST"/>

#### Demo配置

applicationContext-security.xml配置：

<http pattern="/resources/\*\*" security="none"/>

<http pattern="/\*\*/login.jsp" security="none"/>

<http pattern="/\*\*/login.action\*" security="none"/>

<http auto-config="false" use-expressions="true" entry-point-ref="loginUrlEntryPoint">

<intercept-url pattern="/\*\*" access="permitAll"/>

<remember-me data-source-ref="dataSource"

token-validity-seconds="604800"

remember-me-parameter="beauty.user"

user-service-ref="myUserDetailsService"/>

**<custom-filter before="FILTER\_SECURITY\_INTERCEPTOR" ref="selfDefinedFilterInvocationInterceptor"/>**

**<custom-filter position="FORM\_LOGIN\_FILTER" ref="selfDefinedFormLoginFilter"/>**

**<access-denied-handler ref="selfDefinedAccessDeniedHandler"/>**

<session-management invalid-session-url="/errors/sessionTimeout.jsp" session-authentication-error-url="/errors/secondLogin.jsp">

<concurrency-control max-sessions="1" error-if-maximum-exceeded="true"/>

</session-management>

<logout invalidate-session="true" logout-url="/j\_spring\_security\_logout" logout-success-url="/login.jsp?logout=true"/>

</http>

<authentication-manager alias="authenticationManager" erase-credentials="false">

<authentication-provider ref="**selfDefinedAuthenticationProvider**"/>

</authentication-manager>

applicationContext-security-bean.xml配置：略

对配置中的内容的解释：

* **<custom-filter before="FILTER\_SECURITY\_INTERCEPTOR" ref="selfDefinedFilterInvocationInterceptor"/>**

在spring security权限验证链之前，走自己的权限认证，这个很关键，我们的权限扩展都在这个地方实现；

* **<custom-filter position="FORM\_LOGIN\_FILTER" ref="selfDefinedFormLoginFilter"/>**

自定义实现spring security的登陆验证机制，用户名不存在、密码错误等，同时配置remember-me的持久化功能；

* **<access-denied-handler ref="selfDefinedAccessDeniedHandler"/>**

自定义实现权限认证不通过（无权限操作）的处理；

* **<authentication-provider ref="selfDefinedAuthenticationProvider"/>**

自定义实现spring security登陆功能；

其它的功能需要同学们自已研究了！

#### 资源-权限-角色-用户 表设计

Spring Security基本资源对认证的管理，所以我们需要收集所有需要添加权限的资源信息，通常是url，建议可以通过filter收集所有访问的url，然后配置相应权限！

我在设计的过程中没有使用group，而是采用role来替代，同学们可以自行研究基于group的权限管理；



## Demo演示

略