会话管理 Session Management

> HTTP 会话相关功能由过滤器委托给的 SessionManagementFilter 和 SessionAuthenticationStrategy 接口的组合处理。典型用途包括会话固定保护攻击预防、会话超时检测以及对经过身份验证的用户可以同时打开多少会话的限制

1. 超时检测 Detecting Timeouts

可以配置 Spring Security 以检测无效会话 ID 的提交并将用户重定向到适当的 URL。

XML

```xml
<http>
...
<session-management invalid-session-url="/invalidSession.htm" />
</http>
```

如果您使用此机制来检测会话超时，如果用户注销然后重新登录而不关闭浏览器，它可能会错误地报告错误。这是因为当您使会话无效时会话 cookie 不会被清除，即使用户已注销也会重新提交。您可以在注销时显式删除 JSESSIONID cookie，例如在注销处理程序中使用以下语法

XML

```xml
<!-- 这不能保证适用于每个 servlet 容器，因此您需要在您的环境中对其进行测试 -->
<http>
  <logout delete-cookies="JSESSIONID" />
</http>
```

📌 如果您在代理后面运行应用程序，您还可以通过配置代理服务器来删除会话 cookie。例如，使用 Apache HTTPD 的 mod_headers，以下指令将通过在对注销请求的响应中使 JSESSIONID cookie 过期来删除它（假设应用程序部署在路径 /tutorial 下）

XML

```xml
<LocationMatch '/tutorial/logout'>
```

```
Header always set Set-Cookie "JSESSIONID=;Path=/tutorial;Expires=Thu, 01 Ja
</LocationMatch>
```

2. 并发回话控制 Concurrent Session Control

如果您希望限制单个用户登录到您的应用程序的能力，Spring Security 通过以下简单的补充支持开箱即用。首先，您需要将以下侦听器添加到您的 web.xml 文件中，以使 Spring Security 更新有关会话生命周期事件的信息

XML

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

然后将以下行添加到您的应用程序上下文中, 这将阻止用户多次登录 – 第二次登录将导致第一次登录无效。

XML

```xml
<http>
...
<session-management>
    <concurrency-control max-sessions="1" />
</session-management>
</http>
```

通常您希望阻止第二次登录，在这种情况下您可以使用

XML

```xml
<http>
...
<session-management>
    <concurrency-control max-sessions="1" error-if-maximum-exceeded="true"
</session-management>
</http>
```

第二次登录将被拒绝。“拒绝”是指如果使用基于表单的登录，用户将被发送到 authentication-failure-url。如果通过另一种非交互机制进行第二次身份验证，例如“remember-me”，则会向客户端发送“unauthorized”（401）错误。如果您想使用错误页面，则可以将属性 session-authentication-error-url 添加到 session-management 元素。

3. 会话固定攻击防护 Session Fixation Attack Protection

> 会话固定攻击是一种潜在的风险，恶意攻击者有可能通过访问一个网站来创建一个会话，然后说服另一个用户用同一个会话登录（例如，通过向他们发送一个包含会话标识符作为参数的链接）。 Spring Security 通过创建新会话或以其他方式在用户登录时更改会话 ID 来自动防止这种情况发生

如果您不需要这种保护，或者它与其他一些要求冲突，您可以使用会话固定来控制行为 `<session-management>` 上的-protection 属性，有四个选项

- none – 不要做任何事情。原始会话将被保留
- newSession – 创建一个新的“干净”会话，而不复制现有的会话数据（仍然会复制 Spring Security 相关的属性）
- migrateSession – 创建一个新会话并将所有现有会话属性复制到新会话。这是 Servlet 3.0 或更早的容器中的默认值
- changeSessionId – 不创建新会话。而使用 Servlet 容器 HttpServletRequest#changeSessionId() 提供的会话固定保护。此选项仅在 `Servlet 3.1 (Java EE 7)` 和更新的容器中可用。在旧容器中指定它会导致异常。这是 Servlet 3.1 和更新的容器中的默认值

4. 会话管理过滤器 SessionManagementFilter

SessionManagementFilter 根据 SecurityContextHolder 的当前内容检查 SecurityContextRepository 的内容，以确定用户在当前请求期间是否已通过身份验证，通常通过非交互式身份验证机制，例如预身份验证或记住我[1] 。如果存储库包含安全上下文，则过滤器不执行任何操作。如果不是，并且线程本地 SecurityContext 包含一个（非匿名）身份验证对象，则过滤器假定它们已由堆栈中的前一个过滤器进行身份验证。然后它将调用配置的 SessionAuthenticationStrategy

如果用户当前未经过身份验证，过滤器将检查是否请求了无效的会话 ID（例如，由于超时），如果设置了，则会调用配置的 InvalidSessionStrategy。最常见的行为只是重定向到一个固定的 URL，这被封装在标准实现 SimpleRedirectInvalidSessionStrategy 中。后者也用于通过命名空间配置无效会话 URL，如前所述

> 回话管理过滤器会验证用户是否通过身份验证，如果 SpringSecurityContextHolder 中包含了一个非匿名的 SecurityContext，那么就会调用会话认证通过的安全策略，如果未通过用户身份验证，就会检测会话 Id 是否无效，如果配置了对应的策略则会去调用配置的会话验证策略

5. 会话认证策略 SessionAuthenticationStrategy

SessionAuthenticationStrategy 被 SessionManagementFilter 和 AbstractAuthenticationProcessingFilter 使用，因此如果您使用自定义的表单登录类，例如，您需要将它注入到这两者中。在这种情况下，结合命名空间和自定义 bean 的典型配置可能如下所示

XML

```xml
<http>
<custom-filter position="FORM_LOGIN_FILTER" ref="myAuthFilter" />
<session-management session-authentication-strategy-ref="sas"/>
</http>

<beans:bean id="myAuthFilter" class=
"org.springframework.security.web.authentication.UsernamePasswordAuthentica
    <beans:property name="sessionAuthenticationStrategy" ref="sas" />
    ...
</beans:bean>

<beans:bean id="sas" class=
"org.springframework.security.web.authentication.session.SessionFixationPro
```

如果您将 bean 存储在实现 HttpSessionBindingListener 的会话中，包括 Spring 会话范围的 bean，则使用默认值 SessionFixationProtectionStrategy 可能会导致问题。

6. 并发控制 Concurrency Control

> Spring Security 能够防止主体同时对同一应用程序进行身份验证超过指定次数。许多独立软件开发商利用这一点来强制执行许可，而网络管理员喜欢此功能，因为它有助于防止人们共享登录名。例如，您可以阻止用户“Batman”从两个不同的会话登录到 Web 应用程序。您可以让他们之前的登录失效，也可以在他们再次尝试登录时报告错误，从而阻止第二次登录。请注意，如果您使用第二种方法，则未明确注销的用户（例如，刚刚关闭浏览器）将无法再次登录，直到其原始会话到期。

不完整 查看【 会话管理-并发控制】

脚注列表

1. 通过在认证后执行重定向的机制进行的认证（如表单登录）将不会被 SessionManagementFilter 检测到，因为该过滤器不会在认证请求中被调用。在这些情况下，会话管理功能必须被单独处理。