法規掰掰！
Spring Authorization Server
的 OTP 整合快攻

Sam Wang

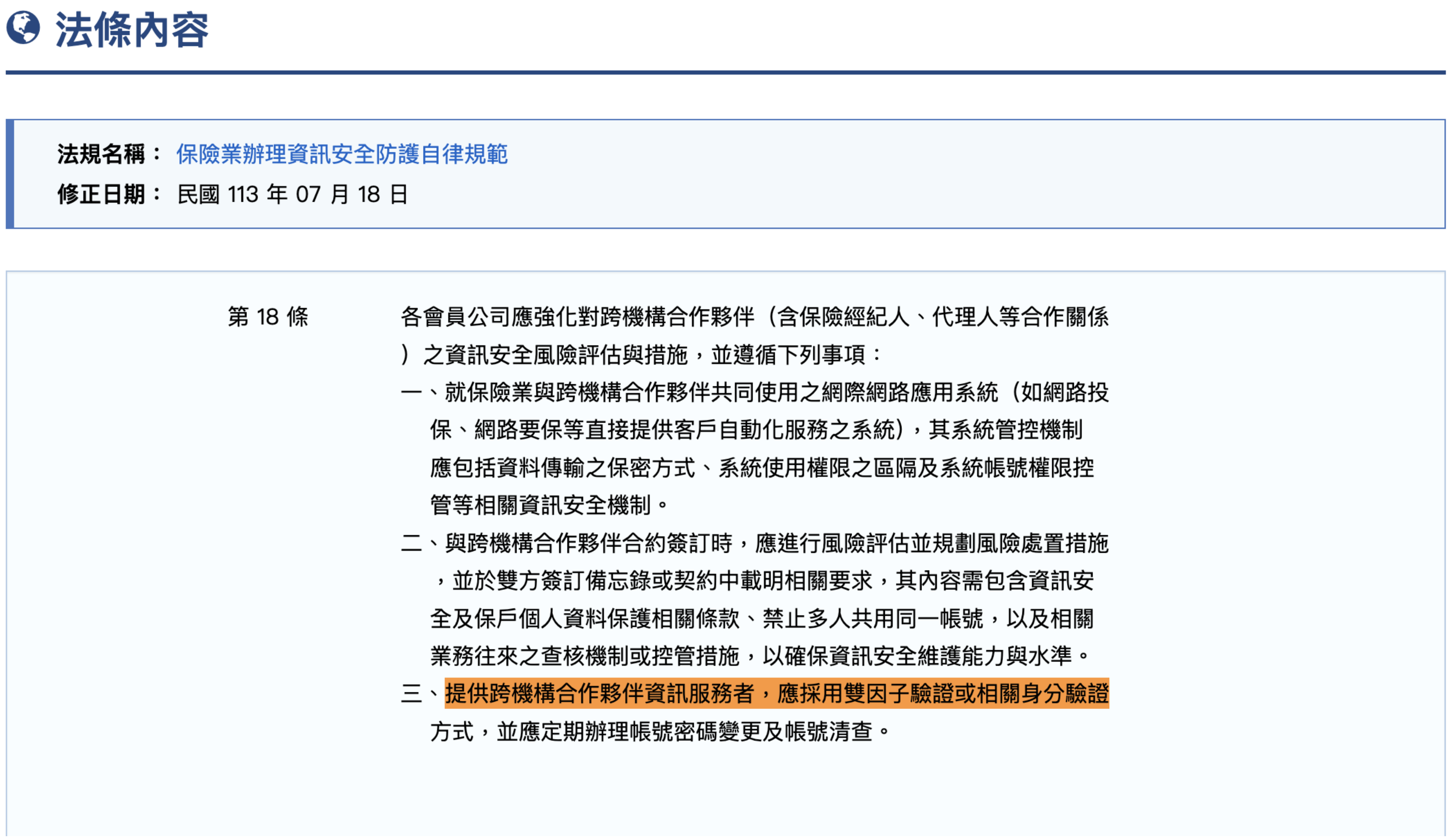
松凌科技 SoftLeader

今日議程

- 背景: 為何需要 2FA 與 OIDC code flow 簡介
- 起點: 如何快速啟動一個 Spring-Authorization-Server with OIDC
- 預設行為: Spring 預設的登入流程是什麼樣子？
- 挑戰: 我們該在哪裡「插隊」加入 2FA？
- 解決方案: 一個針對 Form Login 的快速實現

TQRCG

Why I need 2FA ?



法條內容

法規名稱： 保險業辦理資訊安全防護自律規範
修正日期： 民國 113 年 07 月 18 日

第 18 條

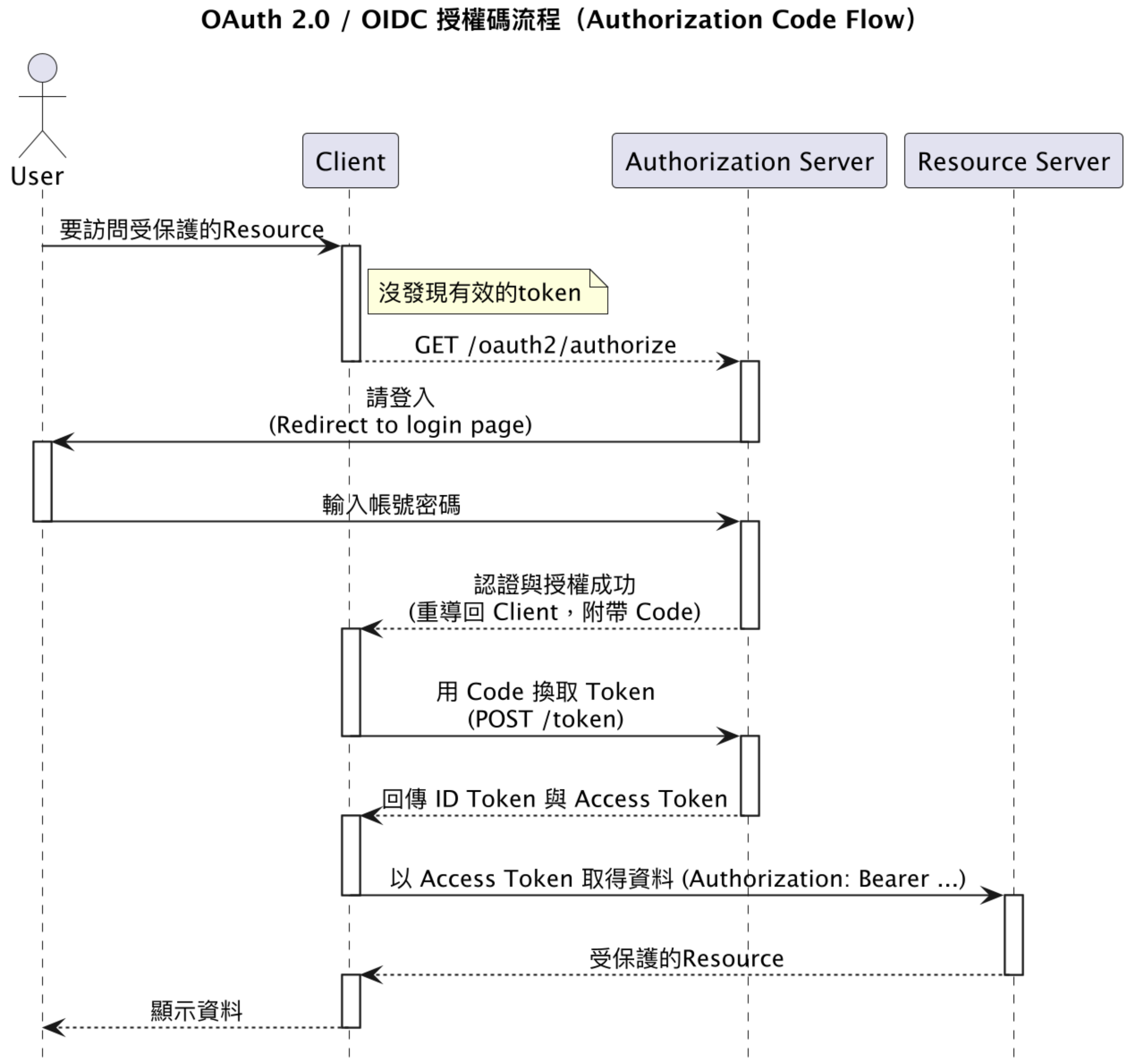
各會員公司應強化對跨機構合作夥伴（含保險經紀人、代理人等合作關係）之資訊安全風險評估與措施，並遵循下列事項：

一、就保險業與跨機構合作夥伴共同使用之網際網路應用系統（如網路投保、網路要保等直接提供客戶自動化服務之系統），其系統管控機制應包括資料傳輸之保密方式、系統使用權限之區隔及系統帳號權限控管等相關資訊安全機制。

二、與跨機構合作夥伴合約簽訂時，應進行風險評估並規劃風險處置措施，並於雙方簽訂備忘錄或契約中載明相關要求，其內容需包含資訊安全及保戶個人資料保護相關條款、禁止多人共用同一帳號，以及相關業務往來之查核機制或控管措施，以確保資訊安全維護能力與水準。

三、提供跨機構合作夥伴資訊服務者，應採用雙因子驗證或相關身分驗證方式，並應定期辦理帳號密碼變更及帳號清查。

OIDC Authorization Code Flow 概覽

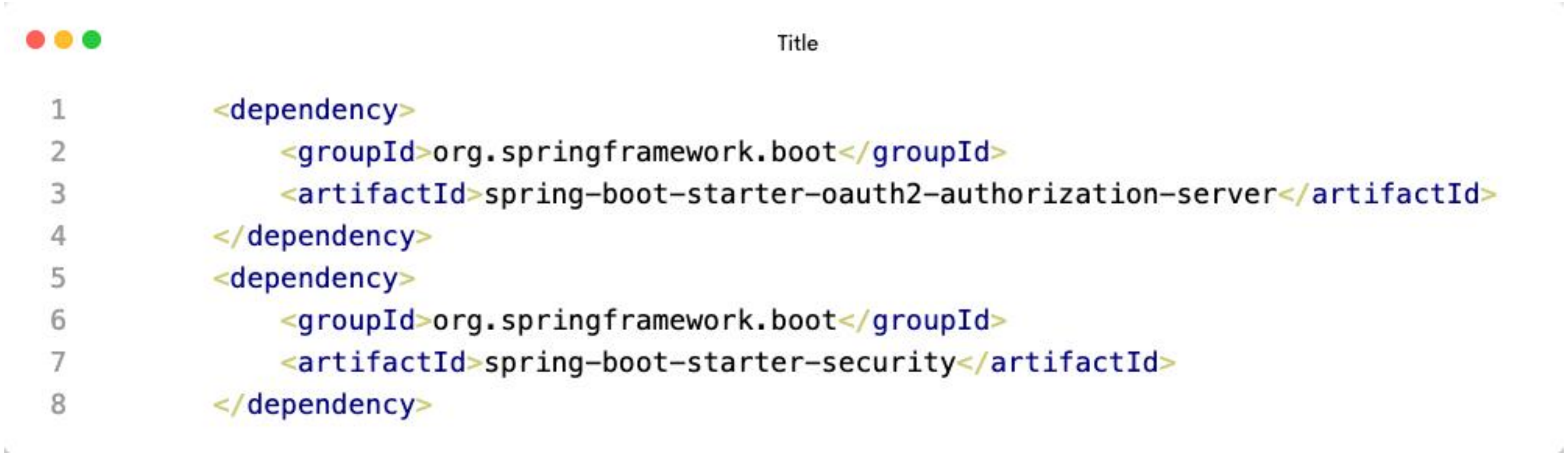


Why Spring-Authorization-Server ?

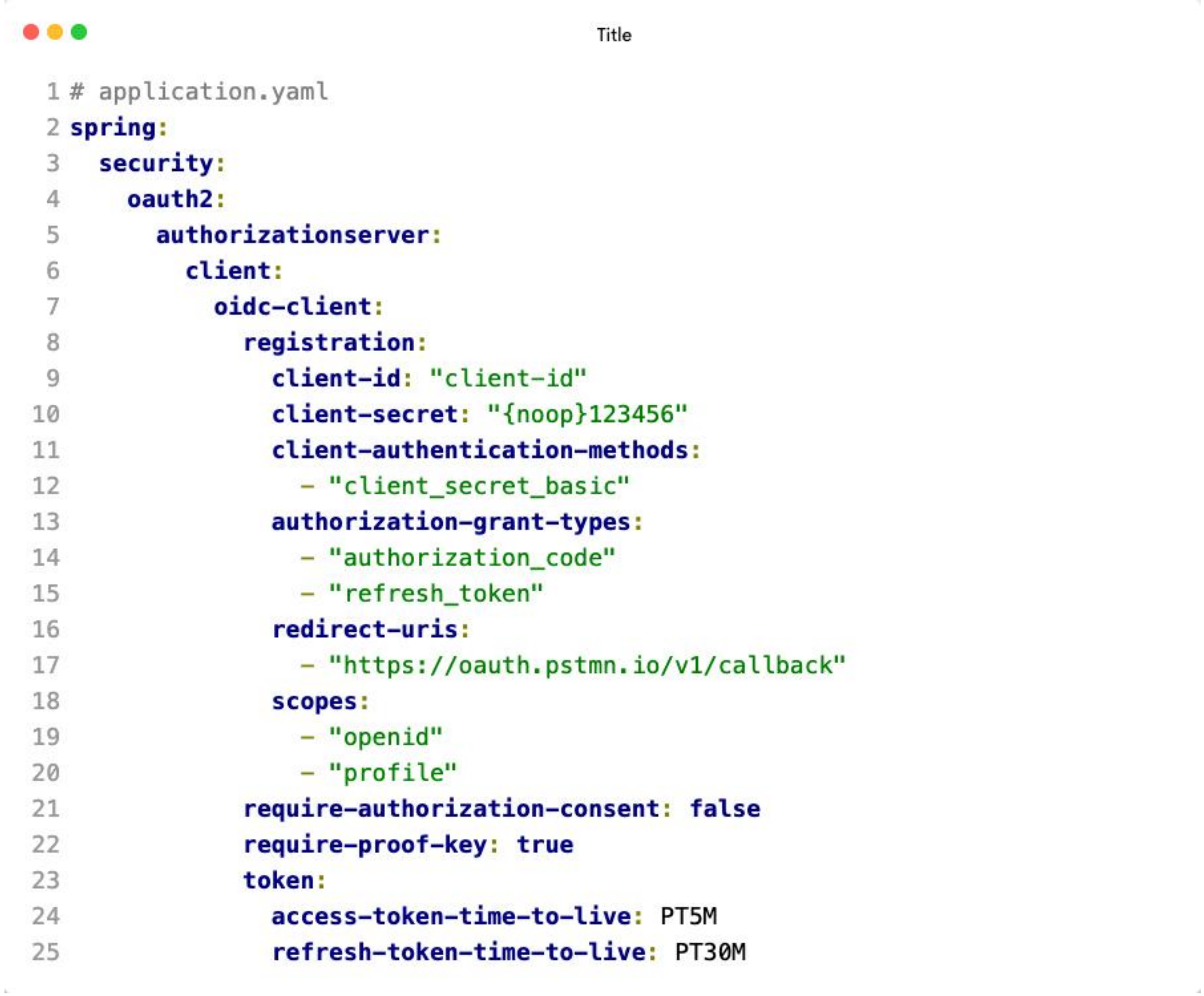
dex

KEYCLOAK

快速啟用 OIDC Server

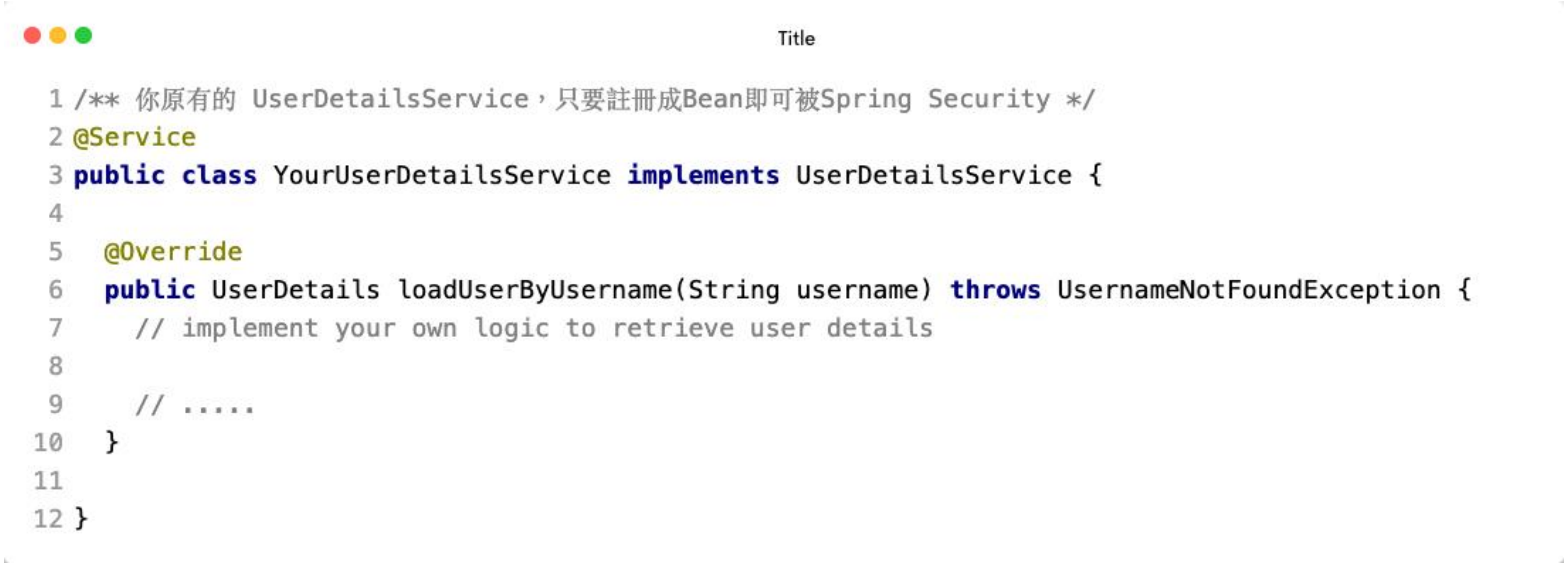


```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-authorization-server</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```



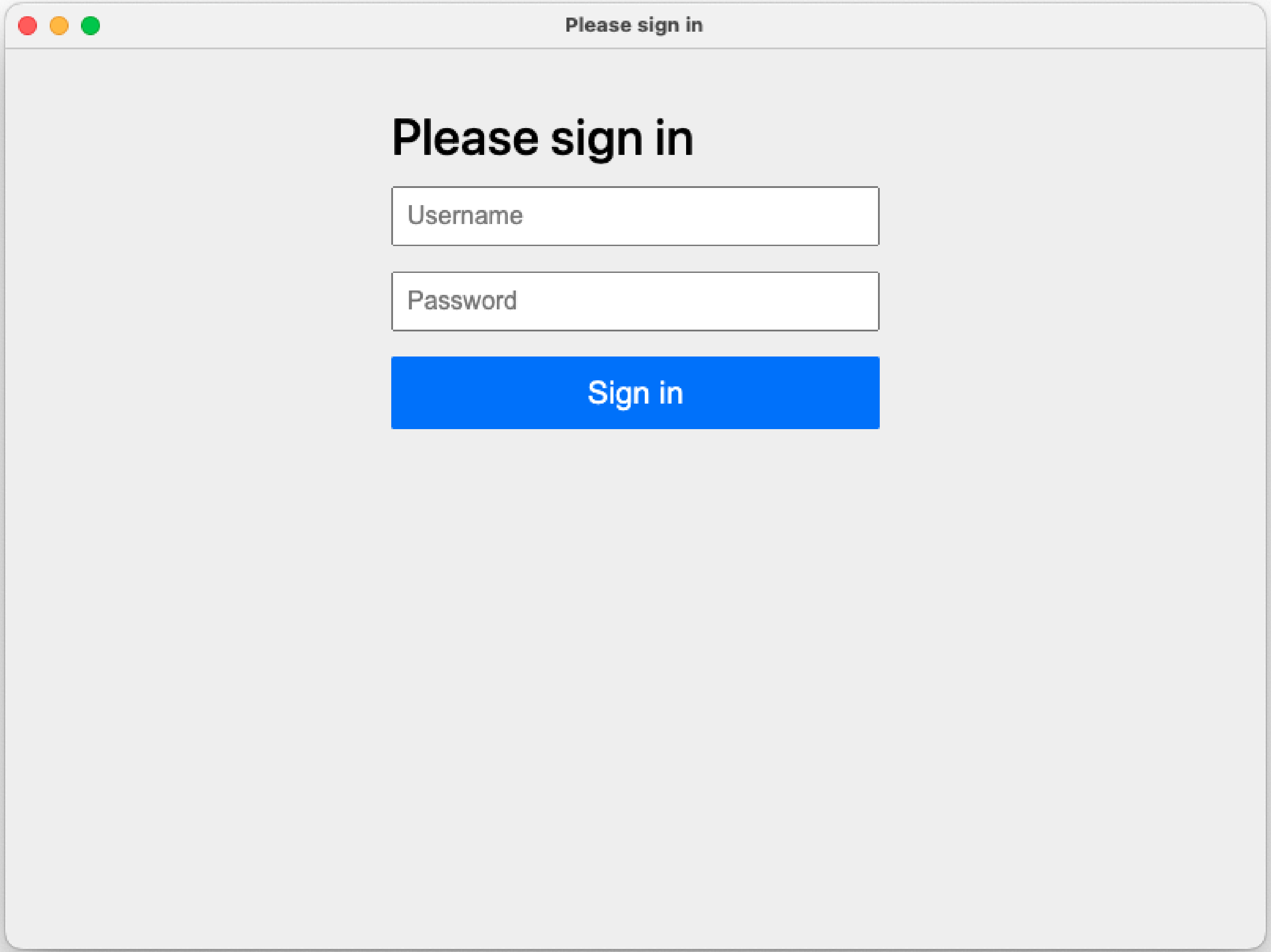
```yaml
# application.yaml
spring:
  security:
    oauth2:
      authorizationserver:
        client:
          oidc-client:
            registration:
              client-id: "client-id"
              client-secret: "{noop}123456"
              client-authentication-methods:
                - "client_secret_basic"
              authorization-grant-types:
                - "authorization_code"
                - "refresh_token"
              redirect-uris:
                - "https://oauth.pstmn.io/v1/callback"
              scopes:
                - "openid"
                - "profile"
            require-authorization-consent: false
            require-proof-key: true
            token:
              access-token-time-to-live: PT5M
              refresh-token-time-to-live: PT30M
```

UserDetailsService

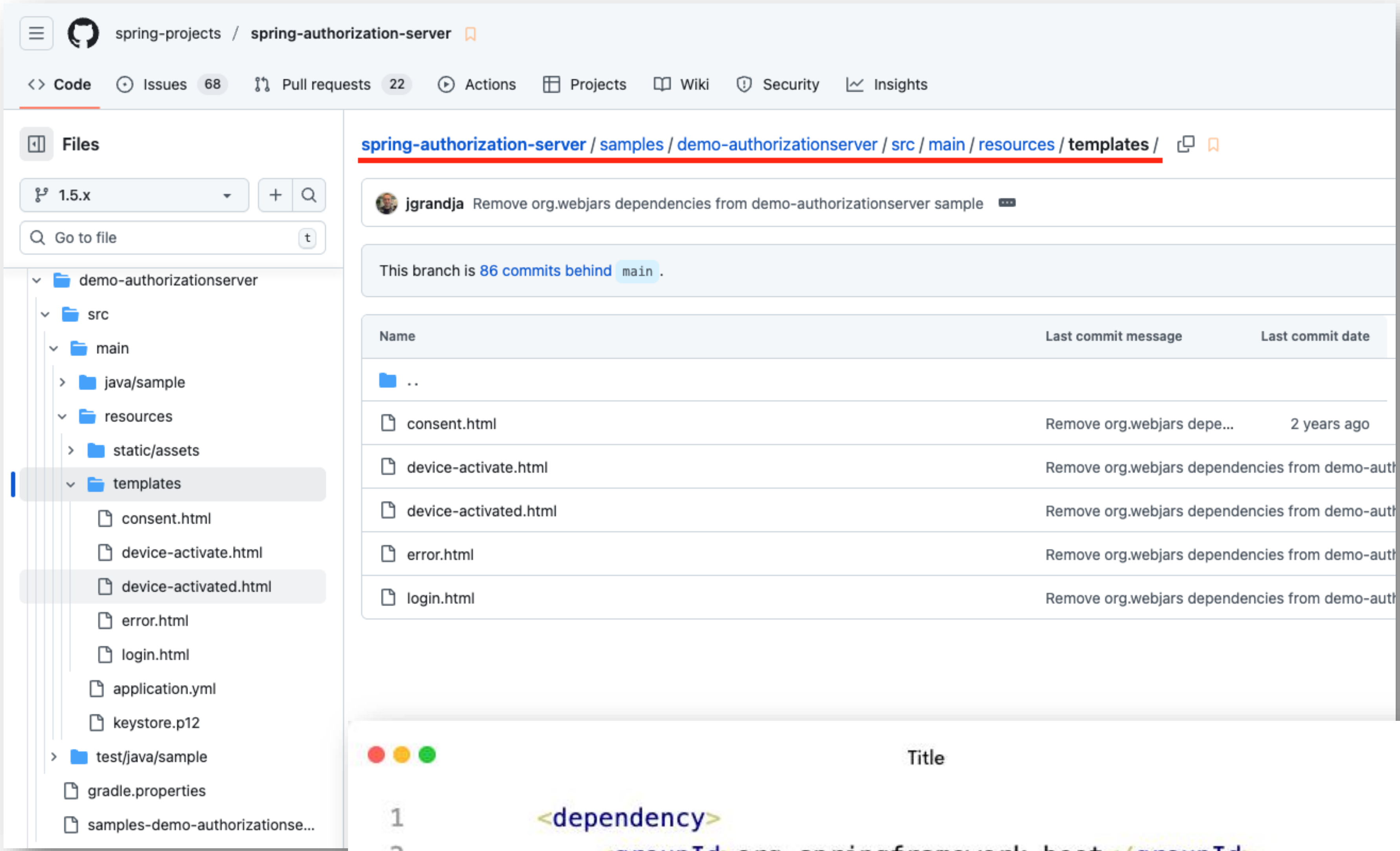


```java
/** 你原有的 UserDetailsService，只要註冊成Bean即可被Spring Security */
@Service
public class YourUserDetailsService implements UserDetailsService {

  @Override
  public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    // implement your own logic to retrieve user details

    // .....
  }

}
```

Default Login Page



Please sign in


spring-projects / spring-authorization-server

Code · Issues 68 · Pull requests 22 · Actions · Projects · Wiki · Security · Insights

Files

1.5.x

Go to file

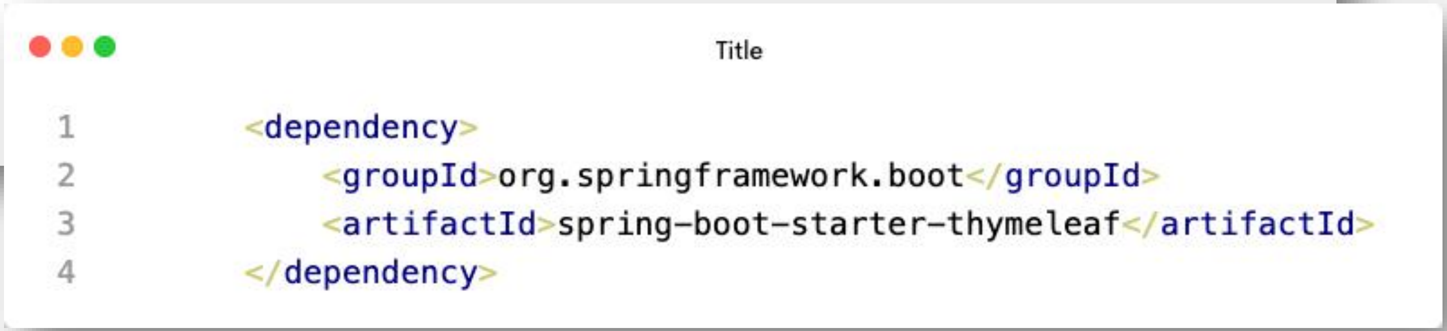
- demo-authorizationserver
  - src
    - main
      - java/sample
      - resources
        - static/assets
        - templates
          - consent.html
          - device-activate.html
          - device-activated.html
          - error.html
          - login.html
        - application.yml
        - keystore.p12
    - test/java/sample
  - gradle.properties
  - samples-demo-authorizationse...

spring-authorization-server / samples / demo-authorizationserver / src / main / resources / templates /

jgrandja Remove org.webjars dependencies from demo-authorizationserver sample

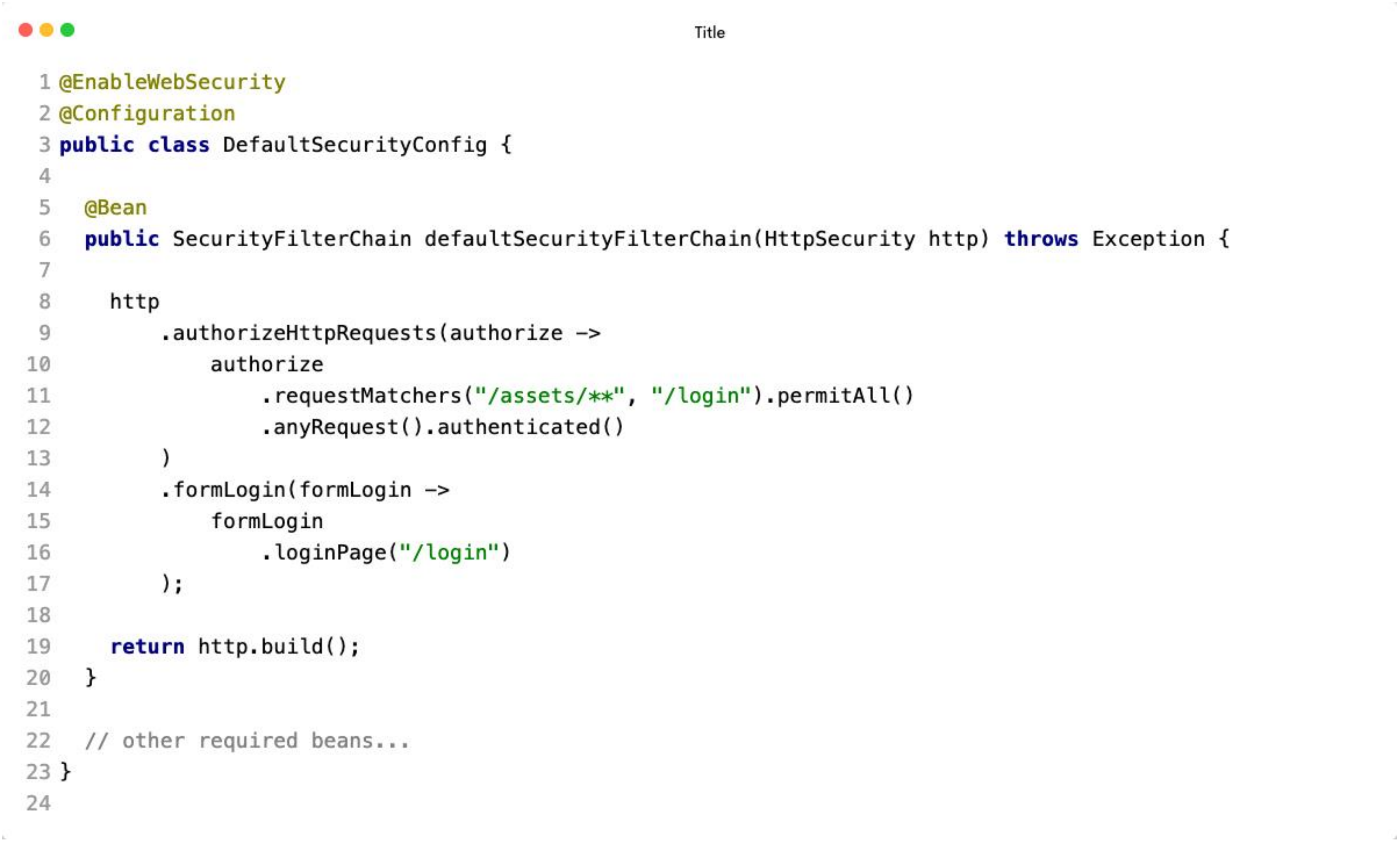
This branch is 86 commits behind main.

Name	Last commit message	Last commit date
..		
consent.html	Remove org.webjars depe...	2 years ago
device-activate.html	Remove org.webjars dependencies from demo-auth	
device-activated.html	Remove org.webjars dependencies from demo-auth	
error.html	Remove org.webjars dependencies from demo-auth	
login.html	Remove org.webjars dependencies from demo-auth	

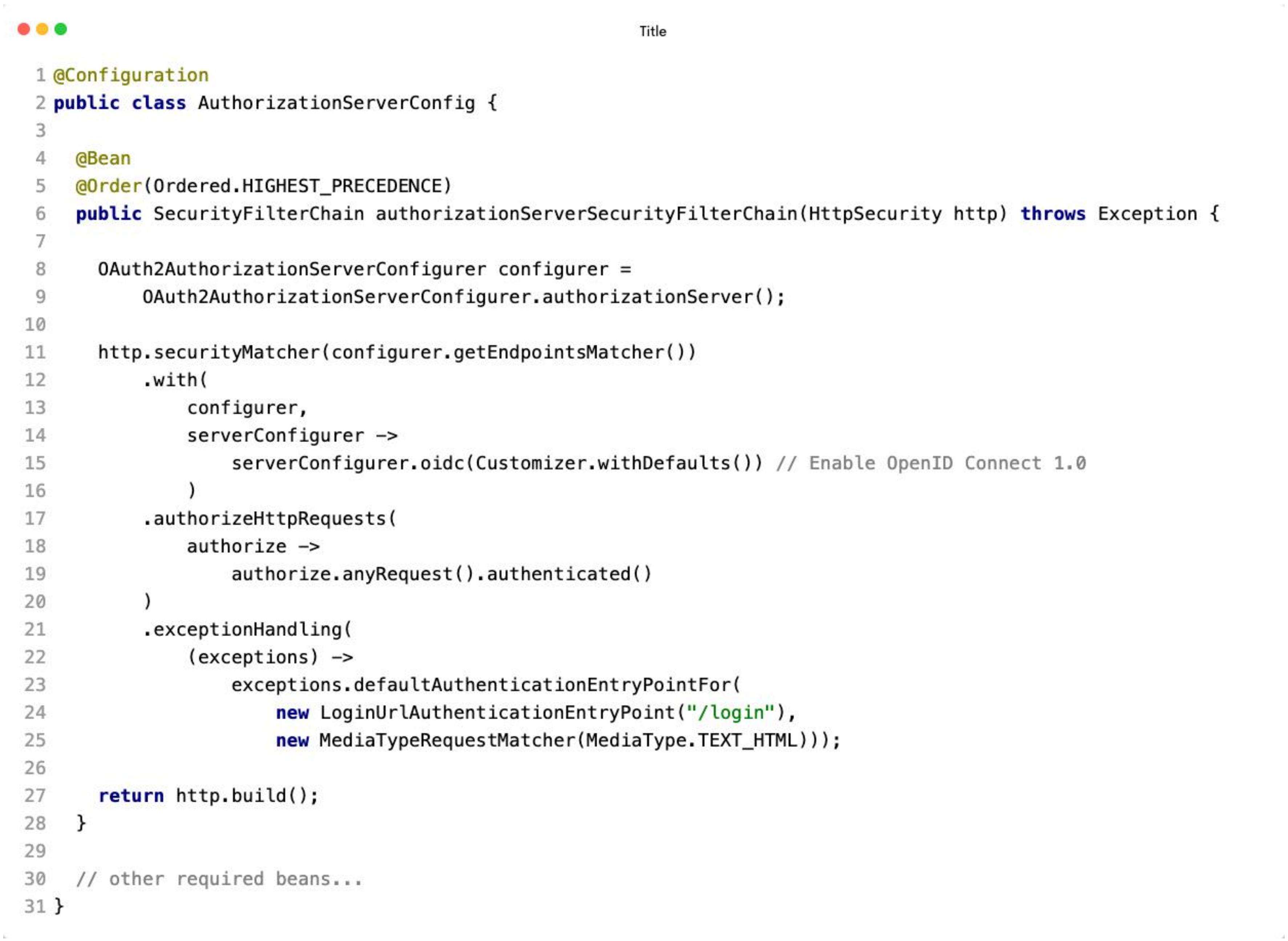

Title

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

DefaultSecurityFilterChain for login page



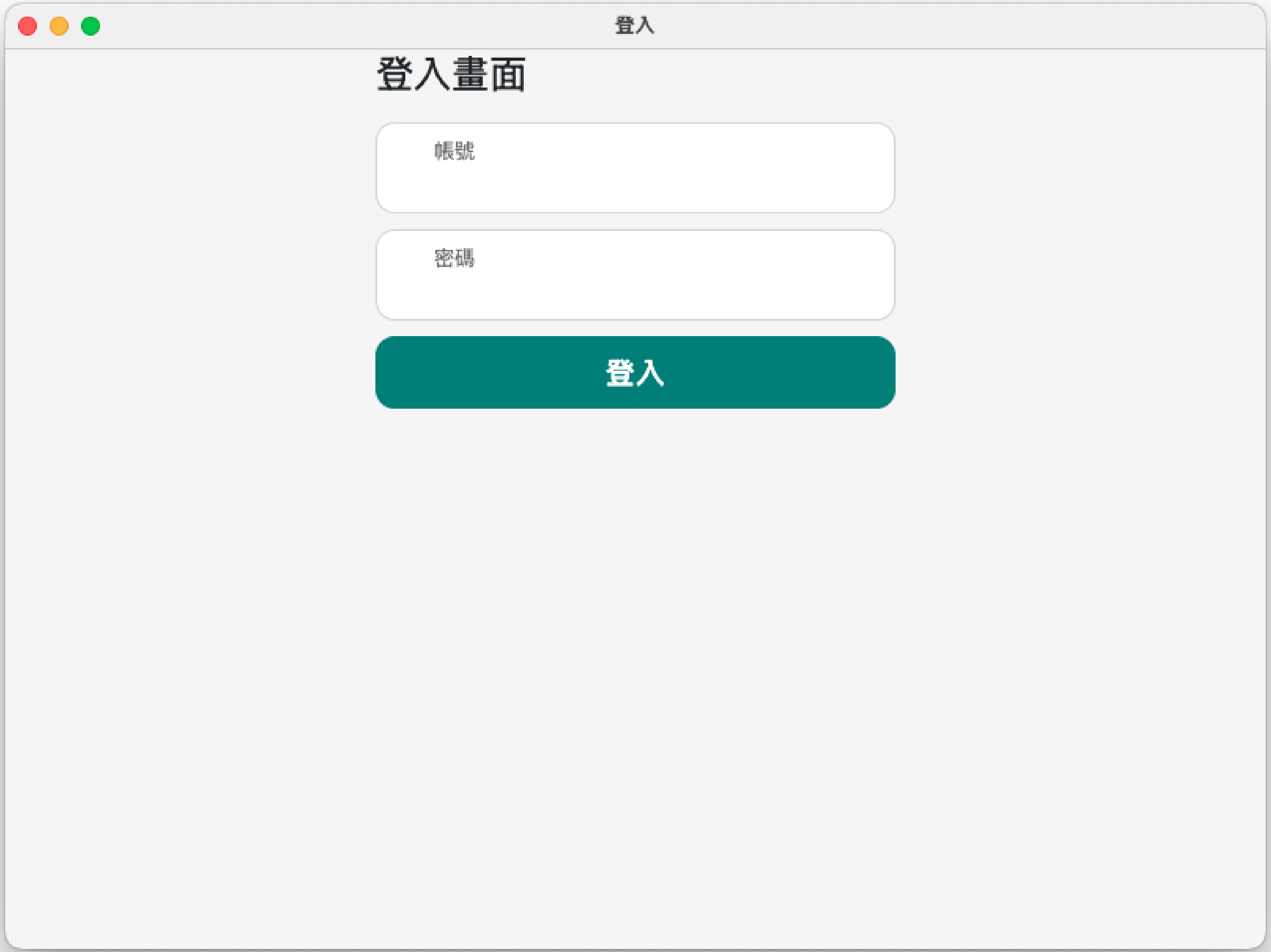
```java
@EnableWebSecurity
@Configuration
public class DefaultSecurityConfig {

  @Bean
  public SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {

    http
        .authorizeHttpRequests(authorize ->
            authorize
                .requestMatchers("/assets/**", "/login").permitAll()
                .anyRequest().authenticated()
        )
        .formLogin(formLogin ->
            formLogin
                .loginPage("/login")
        );

    return http.build();
  }

  // other required beans...
}

```



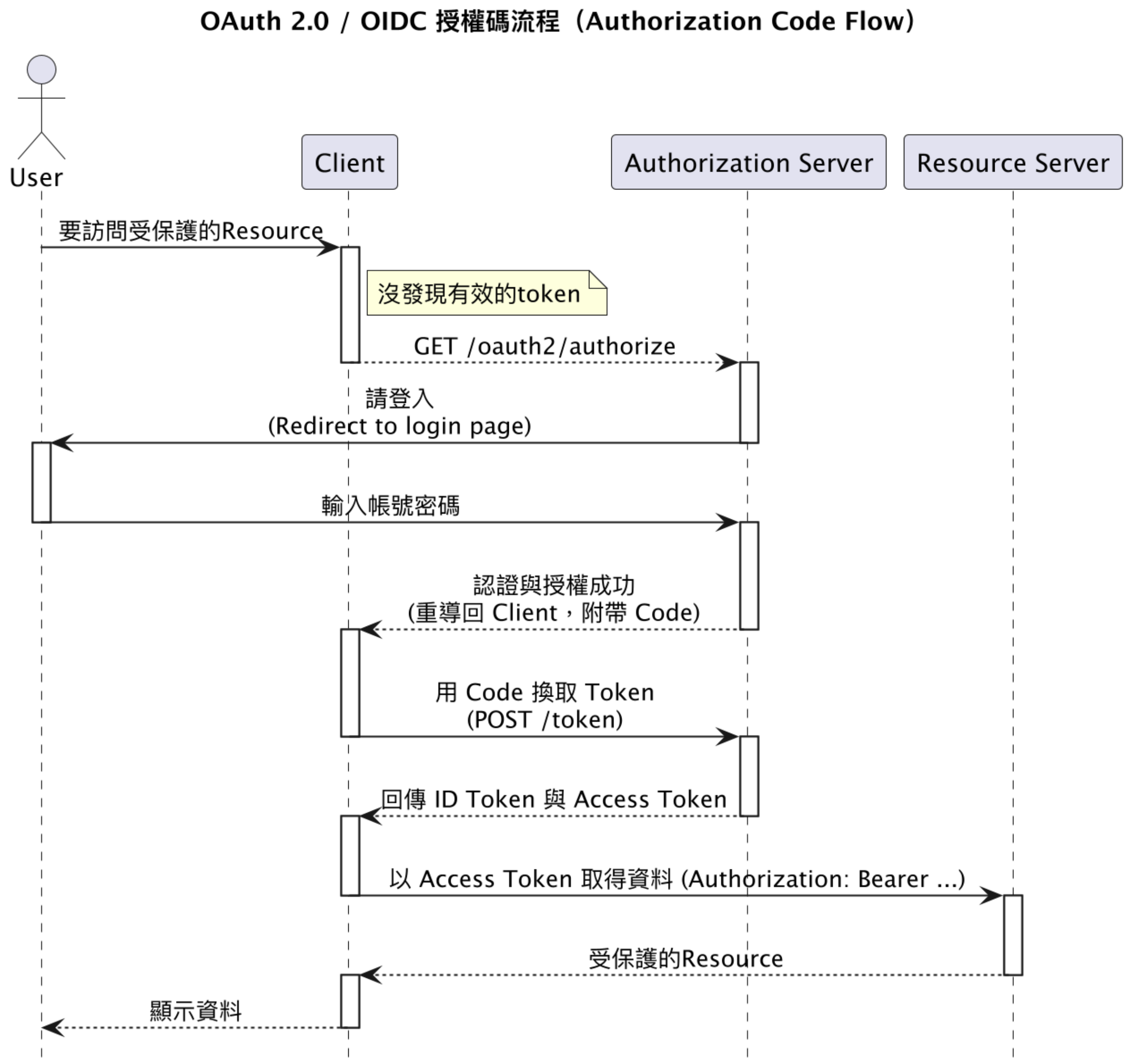
Title

```java
@Configuration
public class AuthorizationServerConfig {

  @Bean
  @Order(Ordered.HIGHEST_PRECEDENCE)
  public SecurityFilterChain authorizationServerSecurityFilterChain(HttpSecurity http) throws Exception {

    OAuth2AuthorizationServerConfigurer configurer =
        OAuth2AuthorizationServerConfigurer.authorizationServer();

    http.securityMatcher(configurer.getEndpointsMatcher())
        .with(
            configurer,
            serverConfigurer ->
                serverConfigurer.oidc(Customizer.withDefaults()) // Enable OpenID Connect 1.0
            )
        .authorizeHttpRequests(
            authorize ->
                authorize.anyRequest().authenticated()
        )
        .exceptionHandling(
            (exceptions) ->
                exceptions.defaultAuthenticationEntryPointFor(
                    new LoginUrlAuthenticationEntryPoint("/login"),
                    new MediaTypeRequestMatcher(MediaType.TEXT_HTML)));

    return http.build();
  }

  // other required beans...
}
```

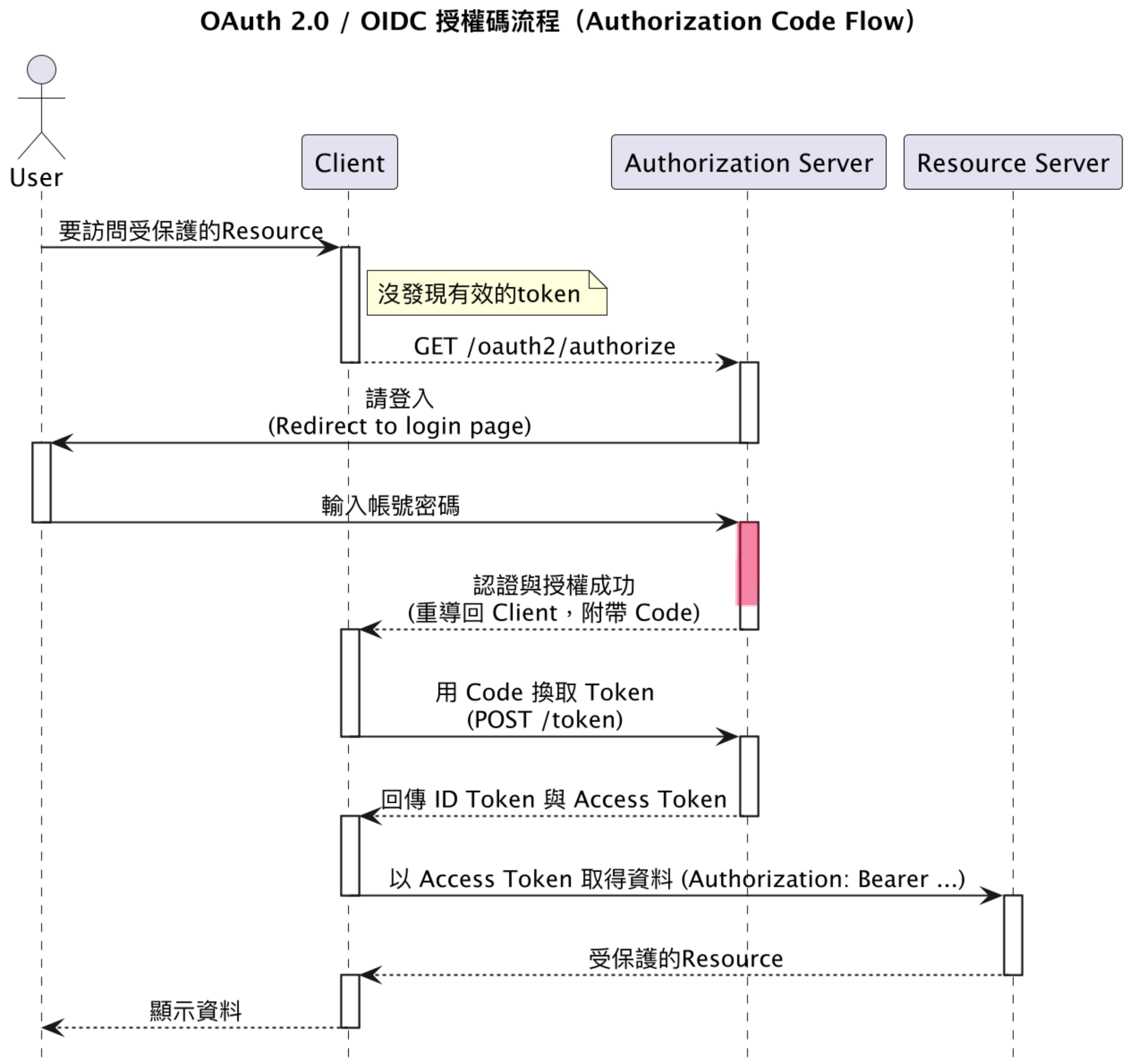
自己的登入畫面



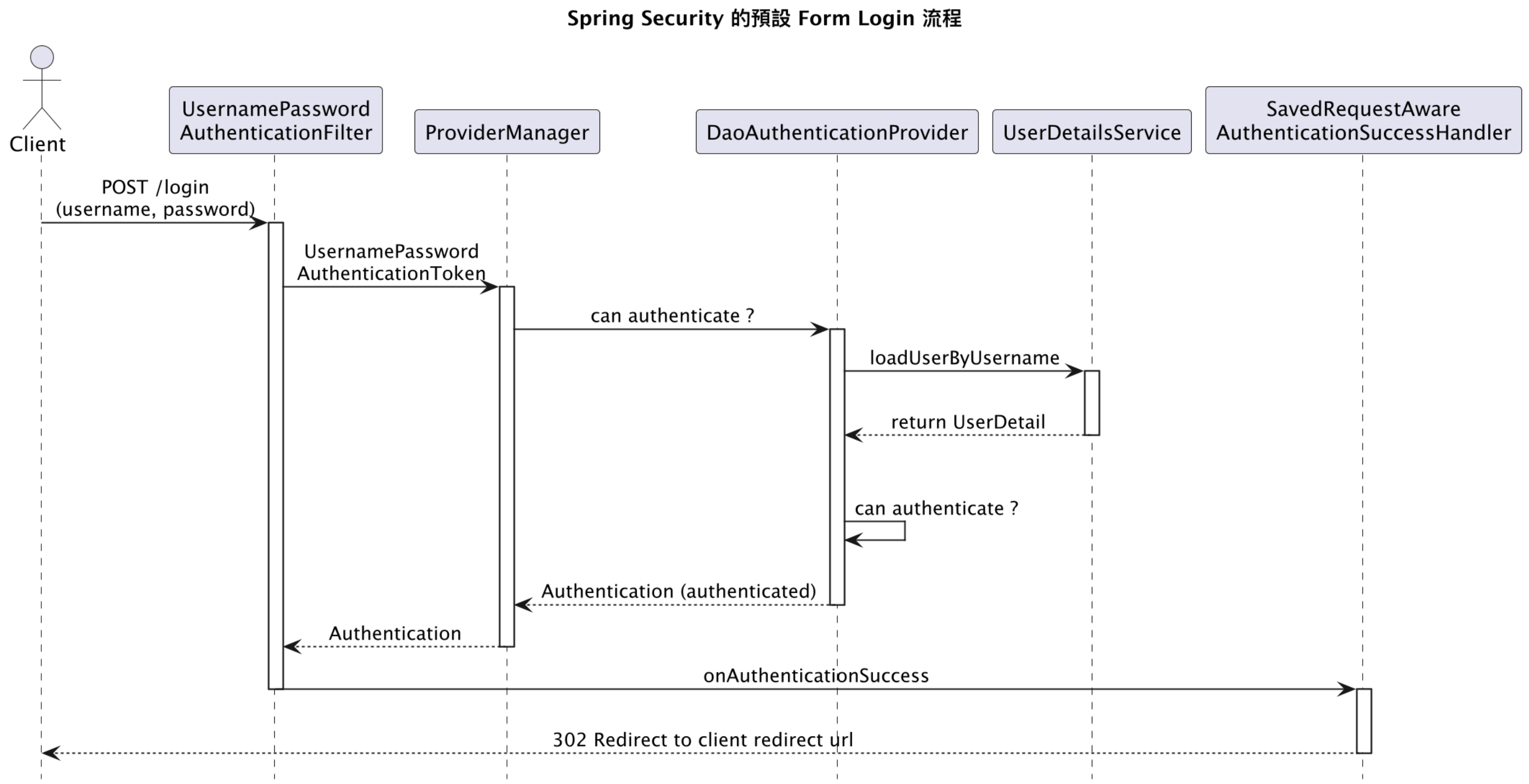
OIDC Authorization Code Flow 概覽



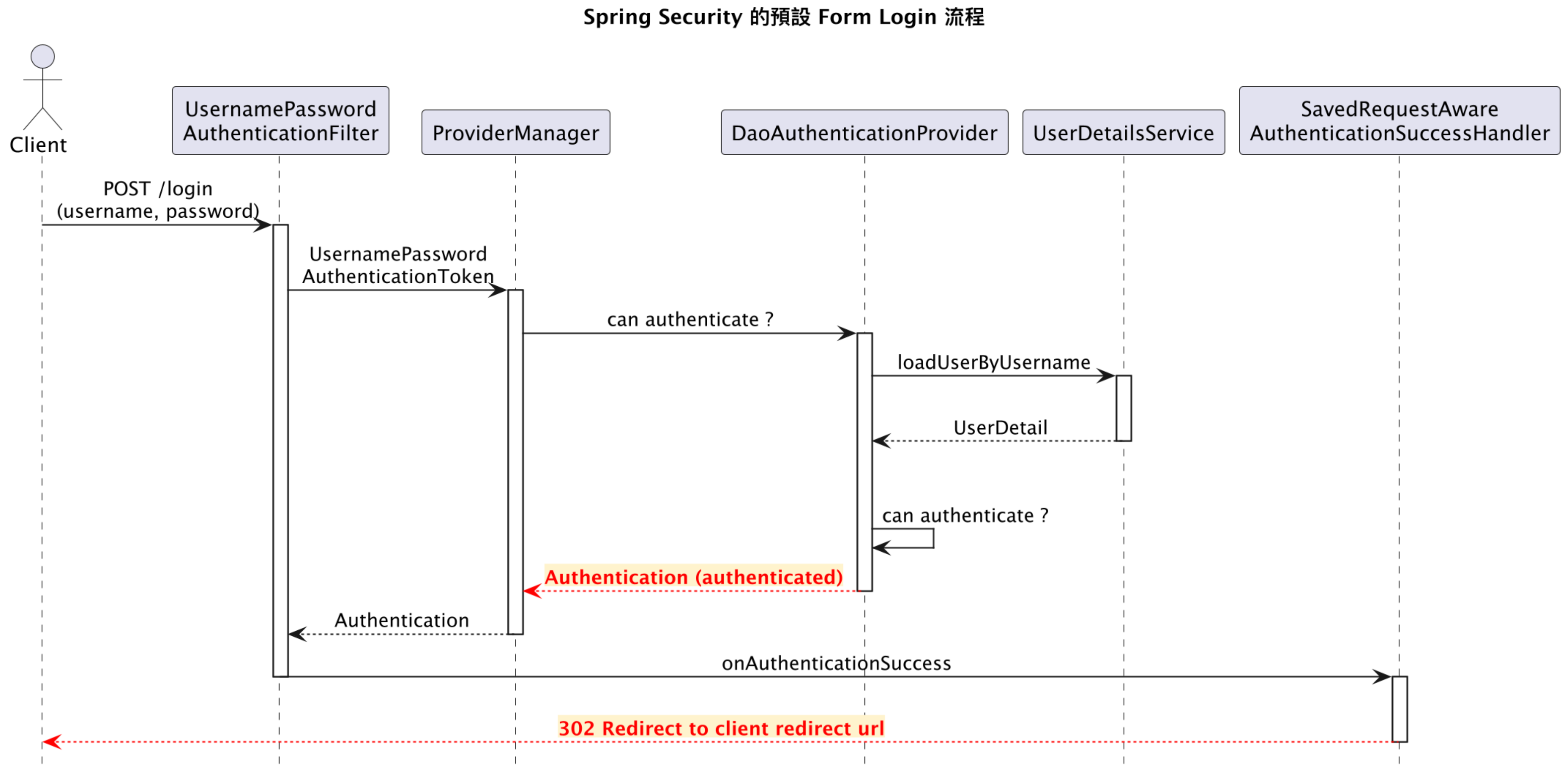
OIDC Authorization Code Flow 概覽



Spring Security 的預設 Form Login 流程


Spring Security 的預設 Form Login 流程
Client
UsernamePassword AuthenticationFilter
ProviderManager
DaoAuthenticationProvider
UserDetailsService
SavedRequestAware AuthenticationSuccessHandler
POST /login (username, password)
UsernamePassword AuthenticationToken
can authenticate ?
loadUserByUsername
return UserDetail
can authenticate ?
Authentication (authenticated)
Authentication
onAuthenticationSuccess
302 Redirect to client redirect url

Spring Security 的預設 Form Login 流程



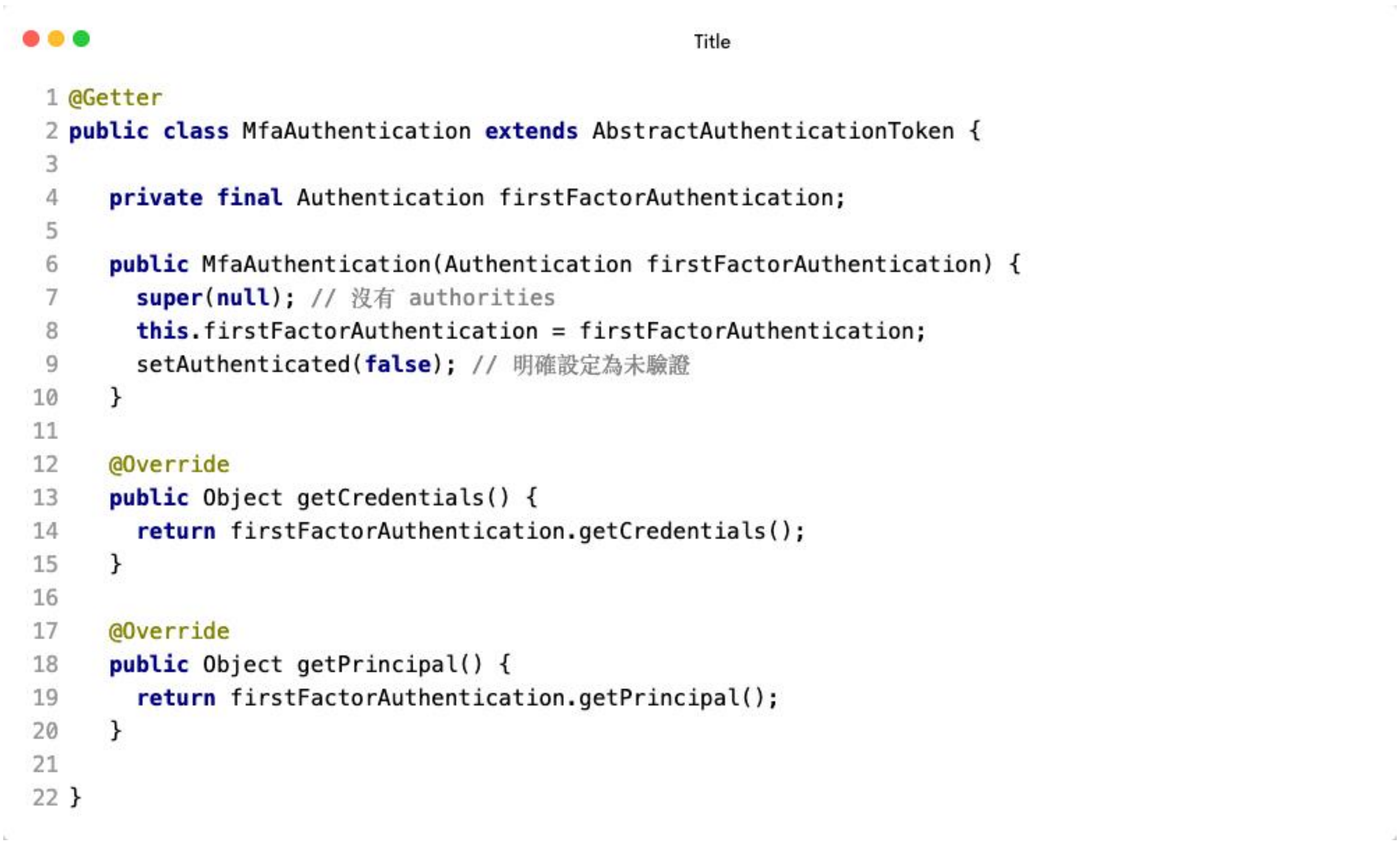
Spring Security 的預設 Form Login 流程
Client
UsernamePassword AuthenticationFilter
ProviderManager
DaoAuthenticationProvider
UserDetailsService
SavedRequestAware AuthenticationSuccessHandler
POST /login (username, password)
UsernamePassword AuthenticationToken
can authenticate ?
loadUserByUsername
UserDetail
can authenticate ?
Authentication (authenticated)
Authentication
onAuthenticationSuccess
302 Redirect to client redirect url

解決方案：客製化核心元件

- **CustomAuthenticationProvider:**
  - 取代預設的 DaoAuthenticationProvider。
  - 職責：只驗證帳號密碼。成功後，不回傳「完全認證」的 Authentication，而是回傳一個自訂的、**「半認證」**的 MfaAuthentication 物件。

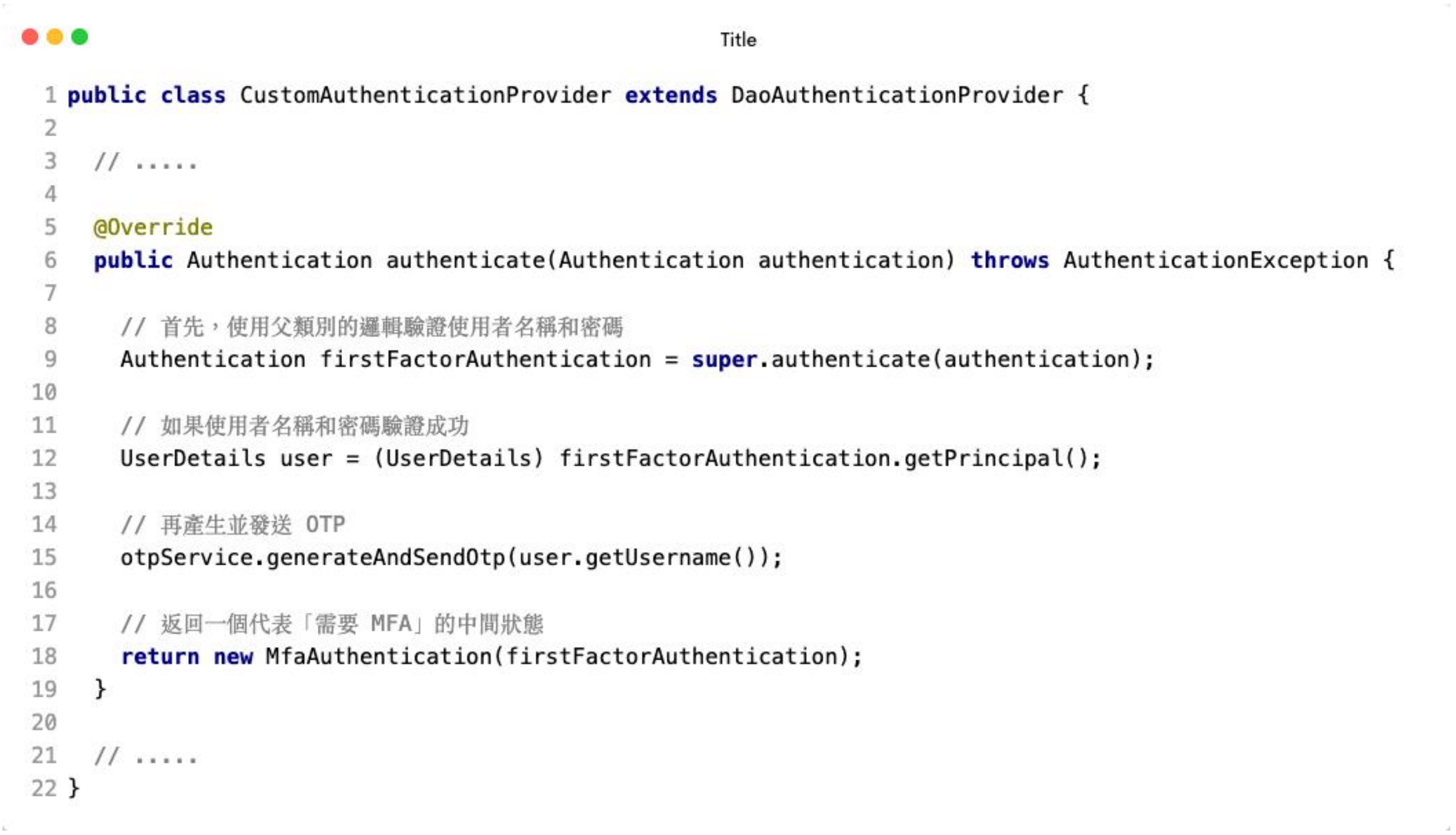
- **CustomLoginSuccessHandler**:
  - 取代預設的SavedRequestAwareAuthenticationSuccessHandler。
  - 職責：不讓他去原本要去的地方，改去mfa驗證畫面。

自訂的MfaAuthentication


Title

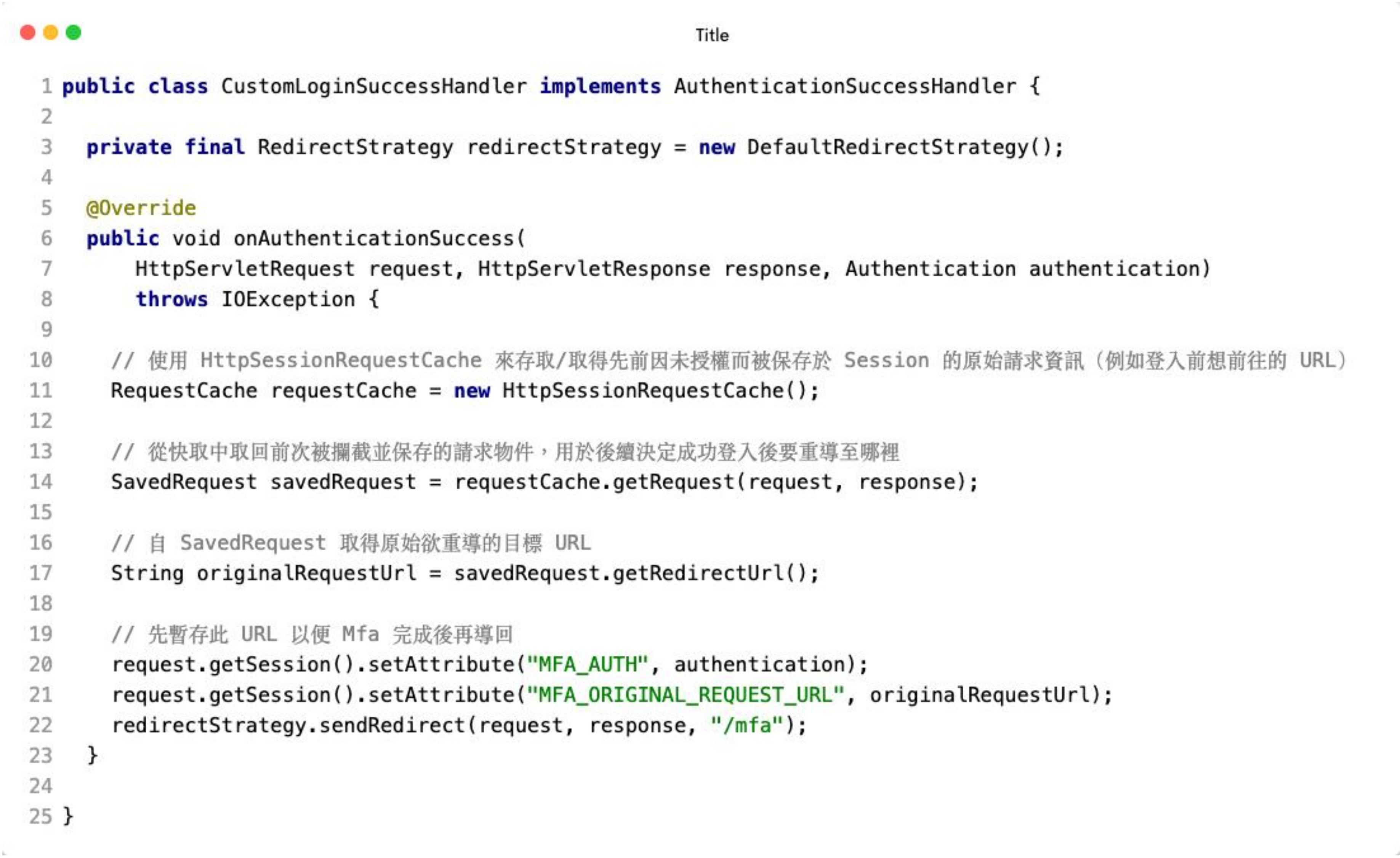
```java
@Getter
public class MfaAuthentication extends AbstractAuthenticationToken {

   private final Authentication firstFactorAuthentication;

   public MfaAuthentication(Authentication firstFactorAuthentication) {
     super(null); // 沒有 authorities
     this.firstFactorAuthentication = firstFactorAuthentication;
     setAuthenticated(false); // 明確設定為未驗證
   }

   @Override
   public Object getCredentials() {
     return firstFactorAuthentication.getCredentials();
   }

   @Override
   public Object getPrincipal() {
     return firstFactorAuthentication.getPrincipal();
   }

}
```

CustomAuthenticationProvider


Title

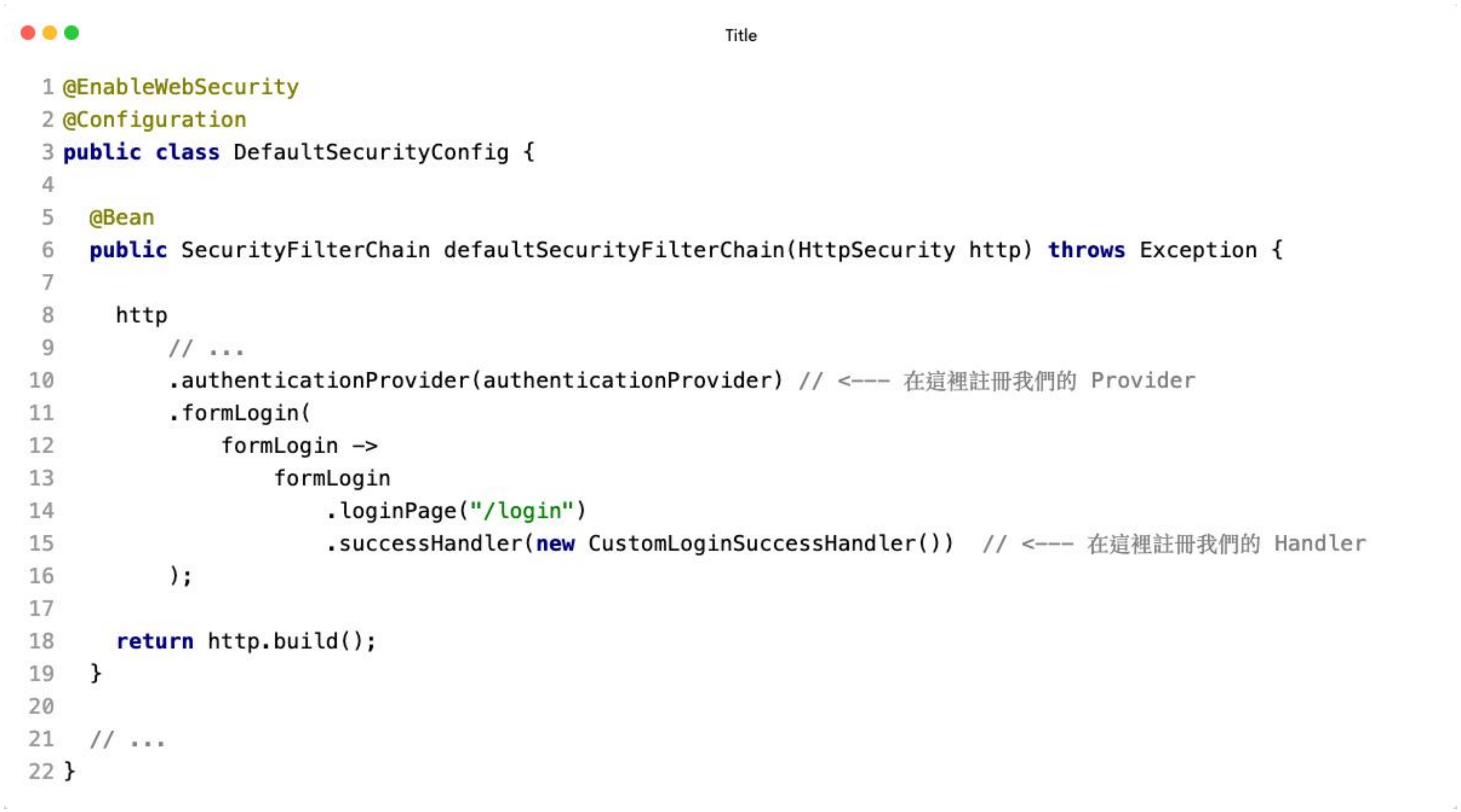
```java
public class CustomAuthenticationProvider extends DaoAuthenticationProvider {

  // .....

  @Override
  public Authentication authenticate(Authentication authentication) throws AuthenticationException {

    // 首先，使用父類別的邏輯驗證使用者名稱和密碼
    Authentication firstFactorAuthentication = super.authenticate(authentication);

    // 如果使用者名稱和密碼驗證成功
    UserDetails user = (UserDetails) firstFactorAuthentication.getPrincipal();

    // 再產生並發送 OTP
    otpService.generateAndSendOtp(user.getUsername());

    // 返回一個代表「需要 MFA」的中間狀態
    return new MfaAuthentication(firstFactorAuthentication);
  }

  // .....
}
```

CustomLoginSuccessHandler



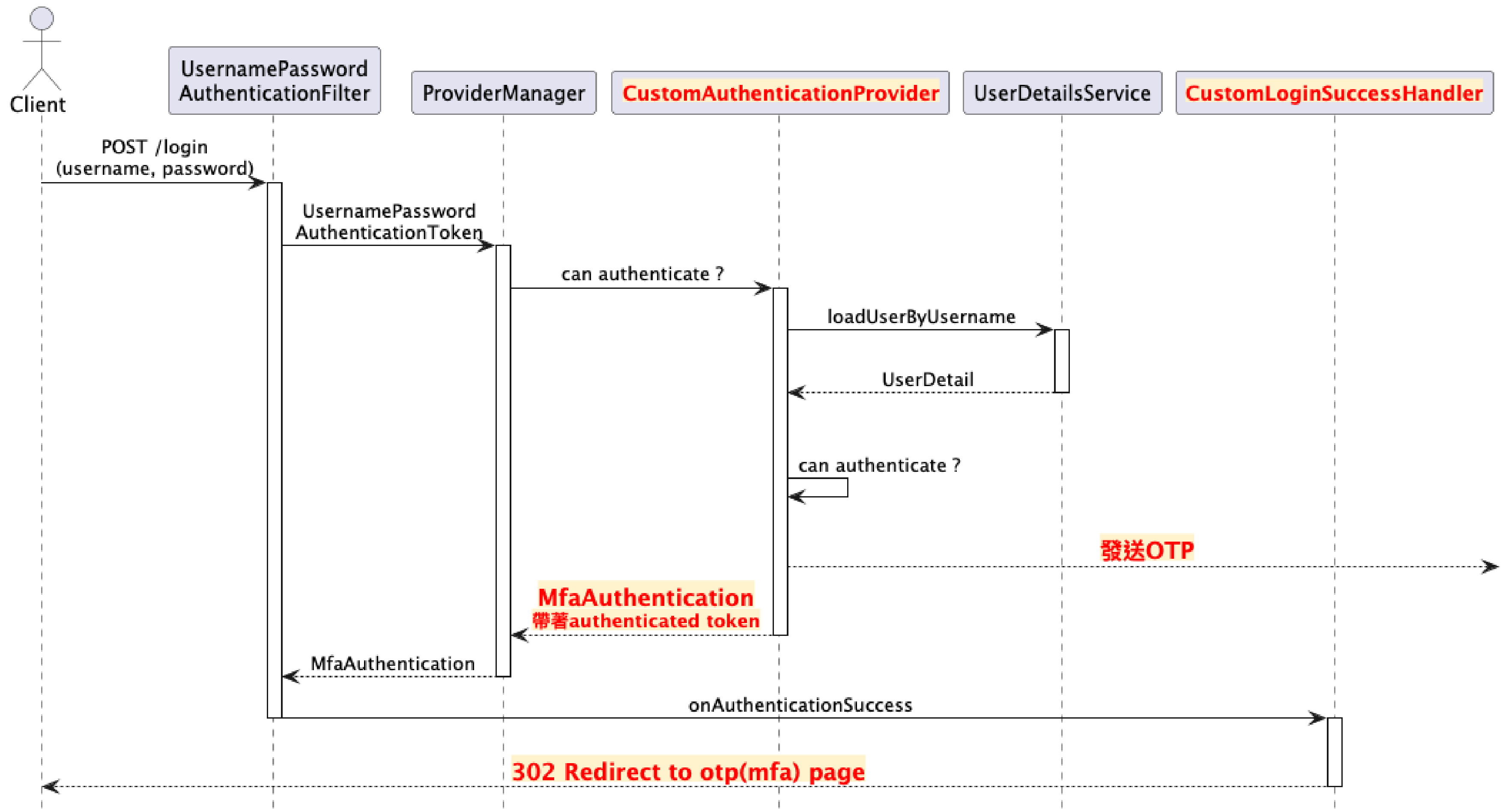
```java
public class CustomLoginSuccessHandler implements AuthenticationSuccessHandler {

  private final RedirectStrategy redirectStrategy = new DefaultRedirectStrategy();

  @Override
  public void onAuthenticationSuccess(
      HttpServletRequest request, HttpServletResponse response, Authentication authentication)
      throws IOException {

    // 使用 HttpSessionRequestCache 來存取/取得先前因未授權而被保存於 Session 的原始請求資訊（例如登入前想前往的 URL）
    RequestCache requestCache = new HttpSessionRequestCache();

    // 從快取中取回前次被攔截並保存的請求物件，用於後續決定成功登入後要重導至哪裡
    SavedRequest savedRequest = requestCache.getRequest(request, response);

    // 自 SavedRequest 取得原始欲重導的目標 URL
    String originalRequestUrl = savedRequest.getRedirectUrl();

    // 先暫存此 URL 以便 Mfa 完成後再導回
    request.getSession().setAttribute("MFA_AUTH", authentication);
    request.getSession().setAttribute("MFA_ORIGINAL_REQUEST_URL", originalRequestUrl);
    redirectStrategy.sendRedirect(request, response, "/mfa");
  }

}
```

啟用我們的客製元件

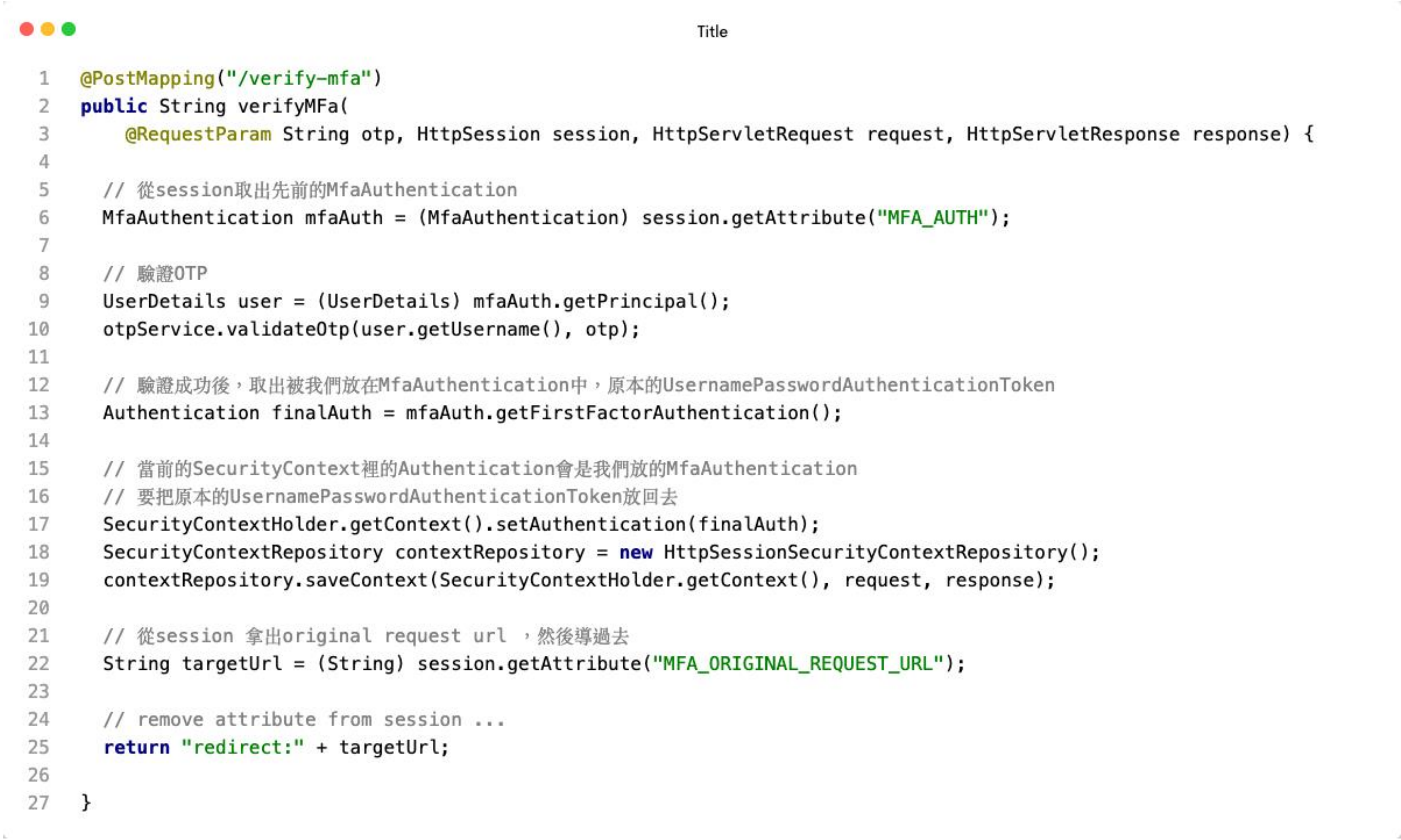

Title

```java
@EnableWebSecurity
@Configuration
public class DefaultSecurityConfig {

  @Bean
  public SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {

    http
        // ...
        .authenticationProvider(authenticationProvider) // <--- 在這裡註冊我們的 Provider
        .formLogin(
            formLogin ->
                formLogin
                    .loginPage("/login")
                    .successHandler(new CustomLoginSuccessHandler())  // <--- 在這裡註冊我們的 Handler
        );

    return http.build();
  }

  // ...
}
```

到此，已完成了....


Client
UsernamePassword AuthenticationFilter
ProviderManager
CustomAuthenticationProvider
UserDetailsService
CustomLoginSuccessHandler
POST /login (username, password)
UsernamePassword AuthenticationToken
can authenticate ?
loadUserByUsername
UserDetail
can authenticate ?
發送OTP
MfaAuthentication 帶著authenticated token
MfaAuthentication
onAuthenticationSuccess
302 Redirect to otp(mfa) page

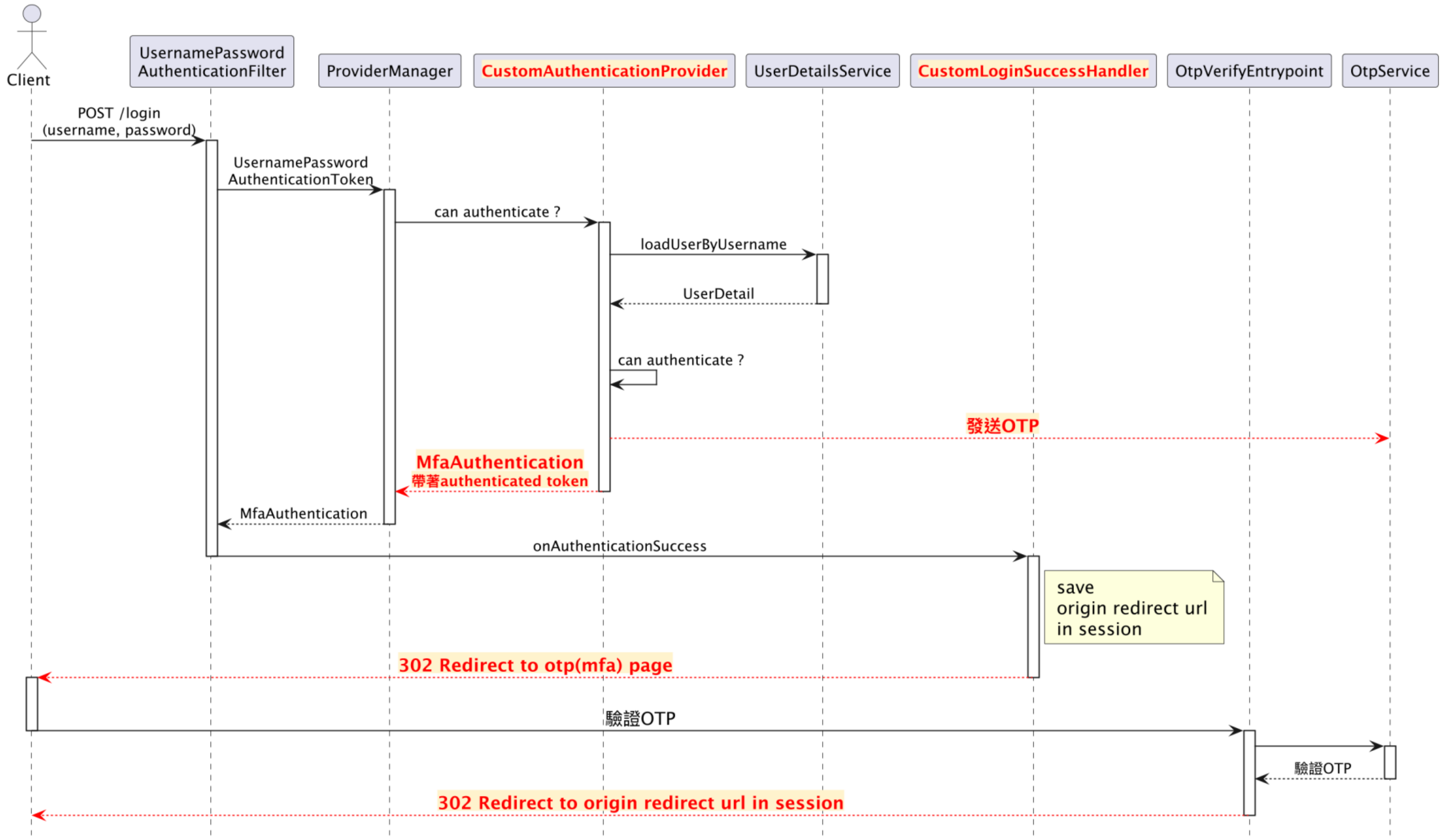
OtpVerifyEntrypoint



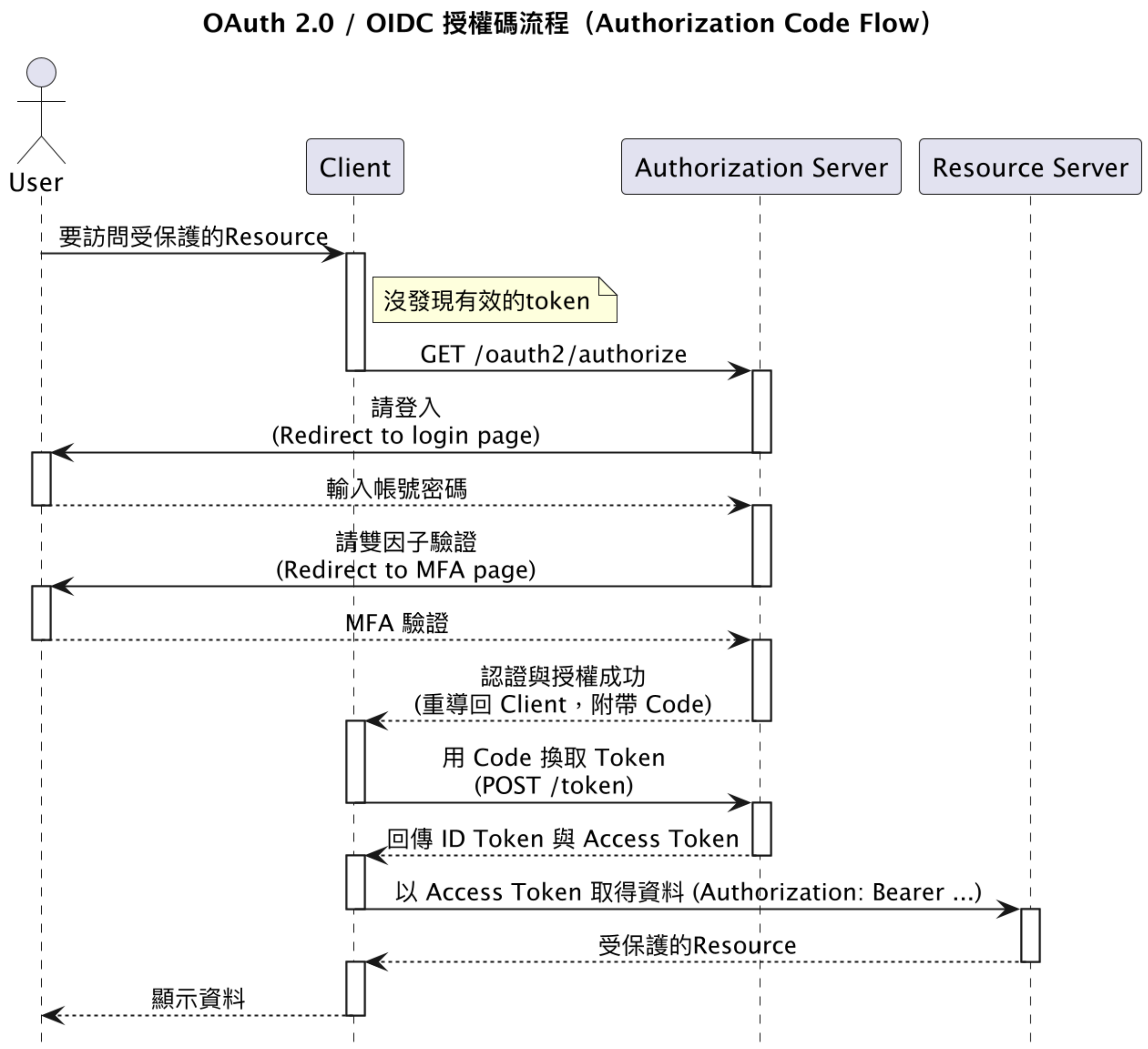
Title

```java
@PostMapping("/verify-mfa")
public String verifyMFa(
    @RequestParam String otp, HttpSession session, HttpServletRequest request, HttpServletResponse response) {

  // 從session取出先前的MfaAuthentication
  MfaAuthentication mfaAuth = (MfaAuthentication) session.getAttribute("MFA_AUTH");

  // 驗證OTP
  UserDetails user = (UserDetails) mfaAuth.getPrincipal();
  otpService.validateOtp(user.getUsername(), otp);

  // 驗證成功後，取出被我們放在MfaAuthentication中，原本的UsernamePasswordAuthenticationToken
  Authentication finalAuth = mfaAuth.getFirstFactorAuthentication();

  // 當前的SecurityContext裡的Authentication會是我們放的MfaAuthentication
  // 要把原本的UsernamePasswordAuthenticationToken放回去
  SecurityContextHolder.getContext().setAuthentication(finalAuth);
  SecurityContextRepository contextRepository = new HttpSessionSecurityContextRepository();
  contextRepository.saveContext(SecurityContextHolder.getContext(), request, response);

  // 從session 拿出original request url ，然後導過去
  String targetUrl = (String) session.getAttribute("MFA_ORIGINAL_REQUEST_URL");

  // remove attribute from session ...
  return "redirect:" + targetUrl;

}
```

調整後的 Form Login 流程


Client
UsernamePassword AuthenticationFilter
ProviderManager
CustomAuthenticationProvider
UserDetailsService
CustomLoginSuccessHandler
OtpVerifyEntrypoint
OtpService
POST /login (username, password)
UsernamePassword AuthenticationToken
can authenticate ?
loadUserByUsername
UserDetail
can authenticate ?
發送OTP
MfaAuthentication
帶著authenticated token
MfaAuthentication
onAuthenticationSuccess
save origin redirect url in session
302 Redirect to otp(mfa) page
驗證OTP
驗證OTP
302 Redirect to origin redirect url in session

調整後的Authorization Code Flow



Thank You