

服务端开发-Spring Security

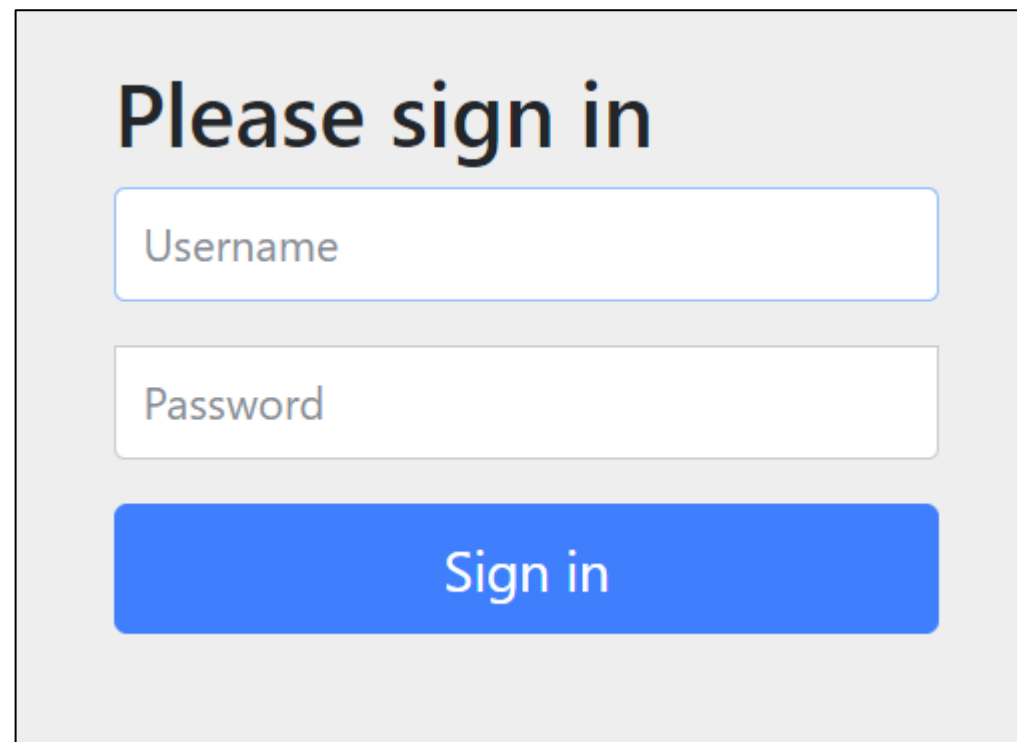
陶召胜

Spring Security

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

加了security starter后自动获得登录界面

- 用户名: user
- 密码查看日志: Using generated security password: cb81f1d1-c11e-4b0a-b728-92d591ffa9c5



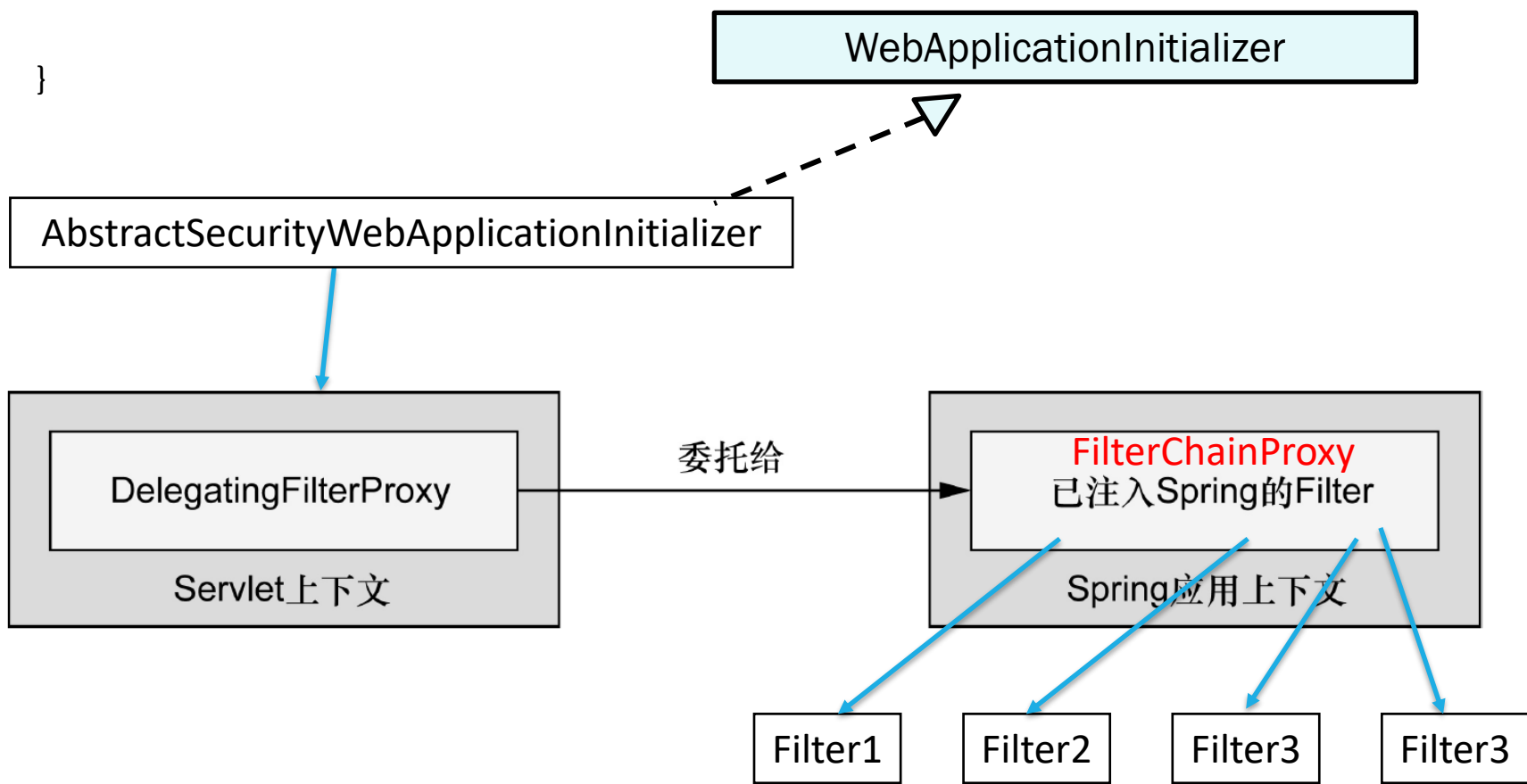
The image shows a login interface with a light gray background. At the top, the text "Please sign in" is displayed in a large, bold, black font. Below this, there are two input fields: the first is labeled "Username" and the second is labeled "Password". Both fields are white with a thin blue border. At the bottom of the form is a blue button with the text "Sign in" in white.

关于Cookie

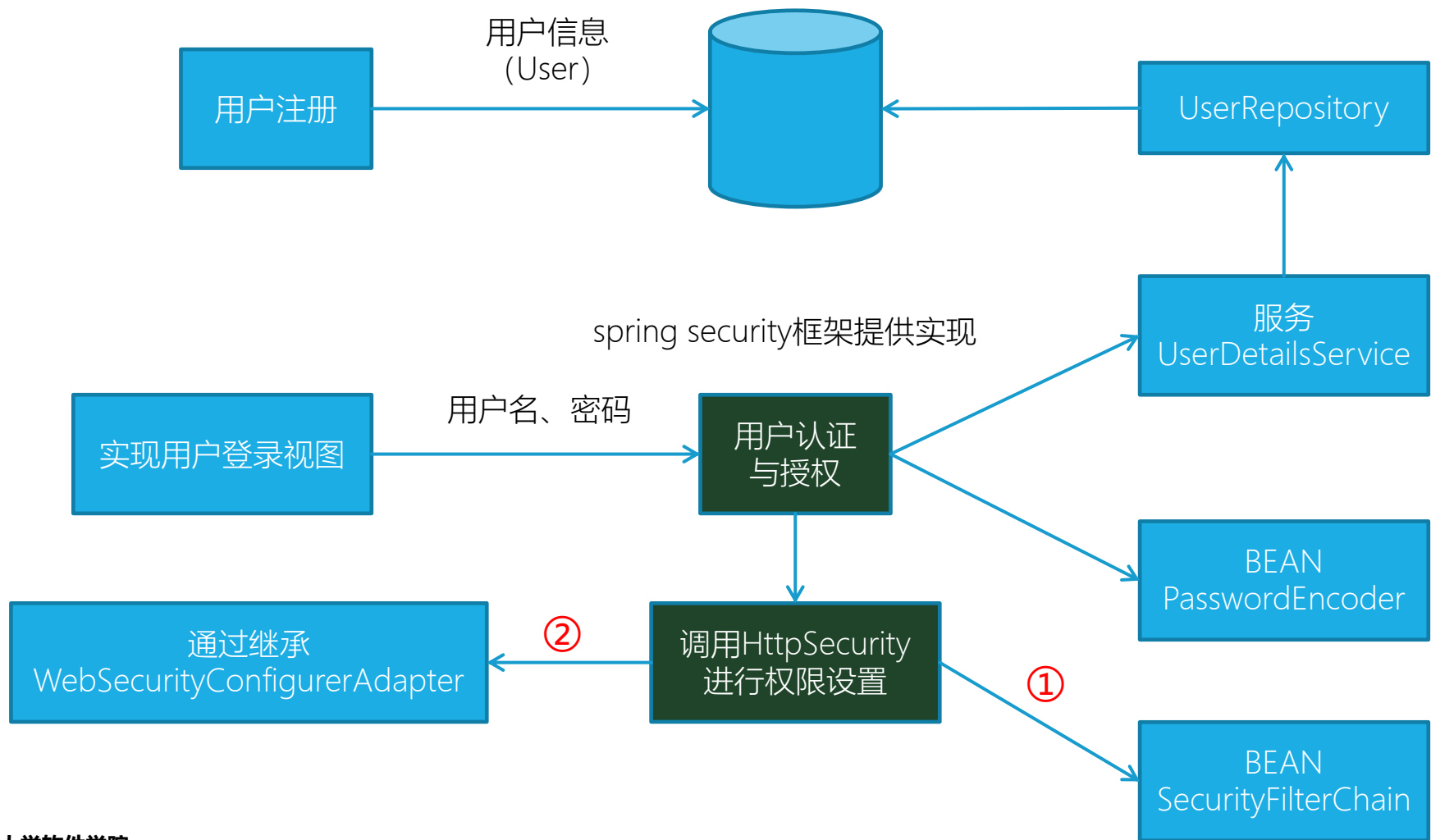
X 标头 载荷 预览 响应 启动器 时间 Cookie	
▼ 常规	
请求网址:	http://localhost:8080/login
请求方法:	POST
状态代码:	● 302 Found
远程地址:	[::1]:8080
引荐来源网址政策:	strict-origin-when-cross-origin
▼ 响应标头	<input type="checkbox"/> 原始
Cache-Control:	no-cache, no-store, max-age=0, must-revalidate
Connection:	keep-alive
Content-Length:	0
Date:	Sat, 14 Oct 2023 07:07:50 GMT
Expires:	0
Keep-Alive:	timeout=60
Location:	http://localhost:8080/design
Pragma:	no-cache
Set-Cookie:	JSESSIONID=D4FBE5FDADBF3B54B874CB18A716223B; Path=/; HttpOnly
X-Content-Type-Options:	nosniff
X-Frame-Options:	SAMEORIGIN
X-Xss-Protection:	1; mode=block

Web请求拦截

```
public class SecurityWebInitializer extends AbstractSecurityWebApplicationInitializer  
{  
  
}  
}
```



开发要做什么



两种配置

- 纯Java配置类

@Configuration

```
public class SecurityConfig {
```

- 继承自WebSecurityConfigurerAdapter

@Configuration

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

纯Java配置类

- 提供密码转换器：PasswordEncoder
- 提供接口实现：UserDetailsService，用于从用户名获取用户信息

密码转码器

- **NoOpPasswordEncoder**: 不编码密码，而保持明文，因为它不会对密码进行哈希化，所以永远不要在真实场景中使用它
- **StandardPasswordEncoder**: 使用SHA-256对密码进行哈希化。这个实现现在已经不推荐了，不应该在新的实现中使用它
- **Pbkdf2PasswordEncoder**: 使用基于密码的密钥派生函数2 (PBKDF2)
- **BCryptPasswordEncoder**: 使用bcrypt强哈希函数对密码进行编码
- **SCryptPasswordEncoder**: 使用scrypt强哈希函数对密码进行编码

用户信息存储

- 内存用户存储
- JDBC用户存储
- LDAP用户存储
 - ✓ LDAP 就是一个轻量目录访问协议, 全称是 (Lightweight Directory Access Protocol), 是基于X.500标准的轻量级目录访问协议, 我们可以理解成是一个目录数据库

使用Spring Data存储库来保存用户

- 定义领域对象：User，实现了UserDetails接口
 - ✓ UserDetailsService接口方法：UserDetails loadUserByUsername(String username)
- 定义持久化接口：UserRepository，增加自定义方法：findByUsername

注册用户

- RegistrationController

保护Web请求

@Bean

```
public SecurityFilterChain filterChain(HttpSecurity http)
```

权限分类

- Authority, 权限
- Role, 角色, ==>>>权限, 加前缀: ROLE_

创建自定义登录页

- 当需要认证时转向的登录页: `.loginPage("/login")`
- 视图控制器, 定义login请求对应的视图: `registry.addViewController("/login");`
- 登录的post请求由Spring Security自动处理, 名称默认: `username`、`password`, 可配置

启用HTTP Basic认证

- 启用HTTP Basic认证
- httpBasic ()

The screenshot shows a REST client interface with the following details:

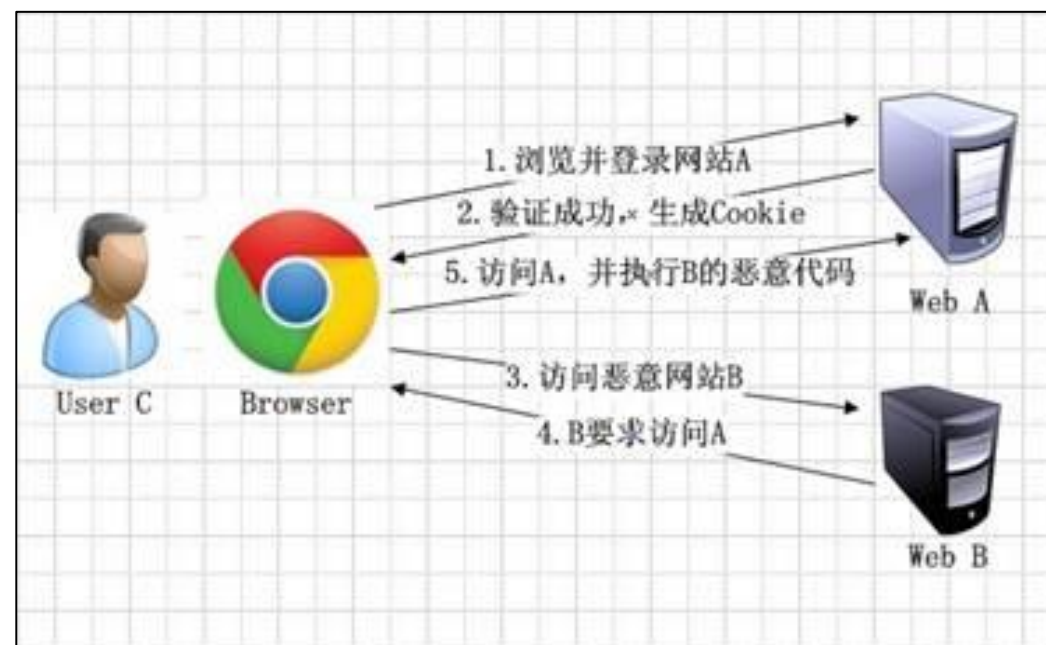
- URL Bar:** Method is **POST**, URL is `http://localhost:8080/admin/deleteOrders`. Buttons for **Save** and **Send** are present.
- Tab Bar:** Includes **Params**, **Auth** (selected), **Headers (9)**, **Body**, **Pre-req.**, **Tests**, **Settings**, and **Cookies**.
- Auth Section:**
 - Type:** A dropdown menu is set to **Basic Auth**.
 - Username:** The field contains the text `txf`.
 - Password:** The field contains the text `123456` and has a warning icon.
 - Description:** A note states: "The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)".

安装postman

- <https://www.postman.com/downloads/>

CSRF攻击

- CSRF, 跨站请求伪造, Cross-site request forgery
- 默认开启
- 如果要关闭: `.csrf().disable()`
- 如果要忽略: `.ignoringAntMatchers("/admin/**")`



实现方法级别的安全

@Configuration

@EnableGlobalMethodSecurity

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

```
@PreAuthorize("hasRole('ADMIN')")
```

获取当前登录的用户

1. DesignTacoController

参数: Principal principal

```
String username = principal.getName();
```

2. OrderController

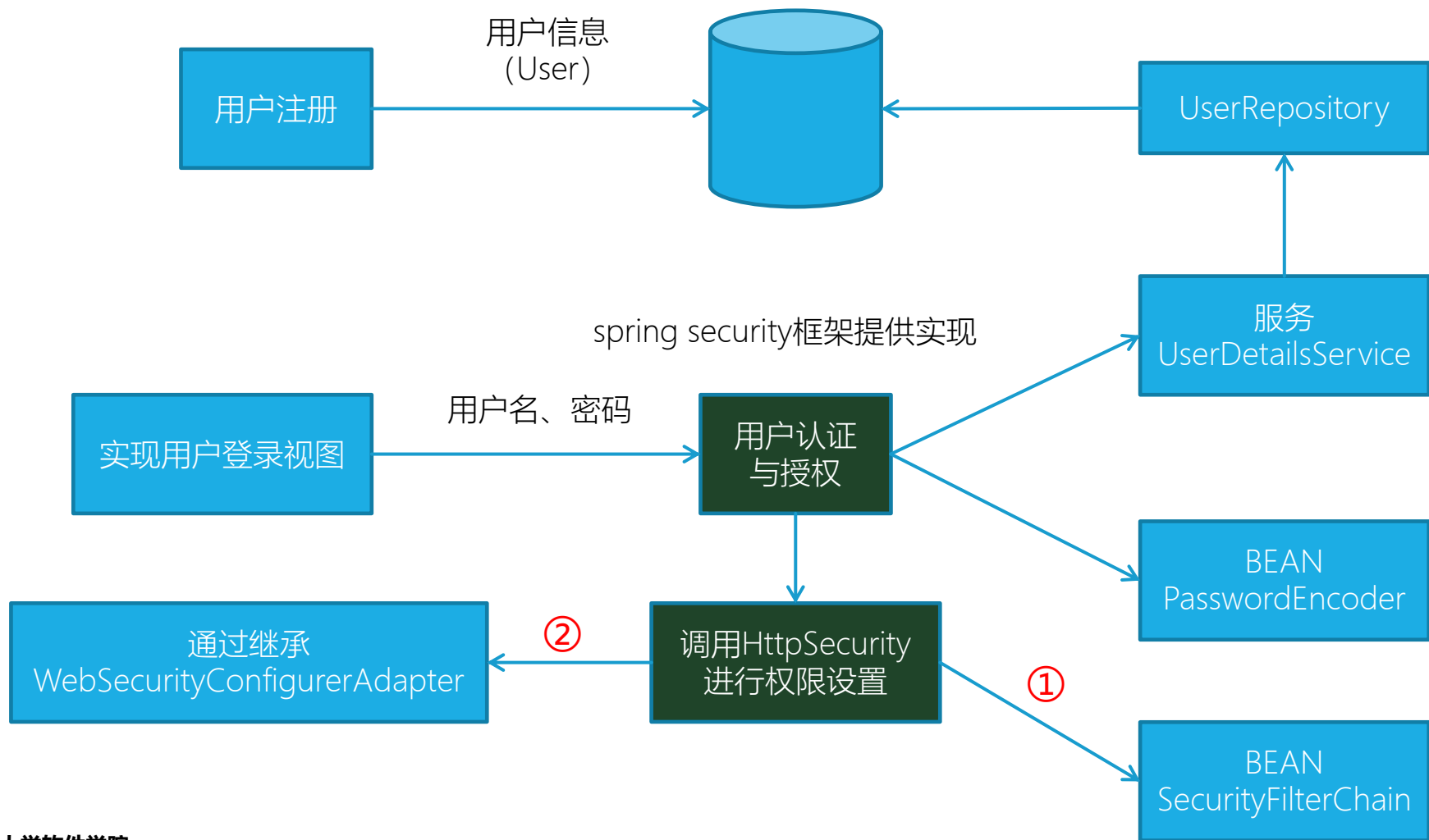
@AuthenticationPrincipal User user

3. 安全上下文获取

```
Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
```

```
User user = (User) authentication.getPrincipal();
```

总结



作业

- 给上节课的作业增加用户认证

Spring Boot Contacts

First Name:

Last Name:

Phone #:

Email:

- san zhang : 13312341234, zhangsan@163.com
- si li : 13345674567, lisi@163.com

要求1：使用mongodb内存数据库

<dependency>

<groupId>de.flapdoodle.embed</groupId>

<artifactId>de.flapdoodle.embed.mongo</artifactId>

</dependency>

要求2：使用默认登录页面



A mockup of a default login page. It features a light gray background with a white rounded rectangle in the center. Inside the rectangle, the text "Please sign in" is displayed in a bold, dark gray font. Below this text are two input fields: the first contains the text "buzz", and the second contains a series of dots followed by a vertical cursor line. At the bottom of the white rectangle is a blue button with the text "Sign in" in white.

要求3：密码无需加密

- `NoOpPasswordEncoder.getInstance()`

要求4：使用内存用户

auth

```
.inMemoryAuthentication()  
  .withUser("buzz")  
    .password("infinity")  
    .authorities("ROLE_USER")  
  .and()  
  .withUser("woody")  
    .password("bullseye")  
    .authorities("ROLE_USER");
```

提交

- 所有源代码，压缩成一个文件。不包含编译后的class文件（删除target目录）

谢谢观看！

