

Day5：ログイン機能と管理者権限

Spring Securityと単体テストの実装

本日のゴール



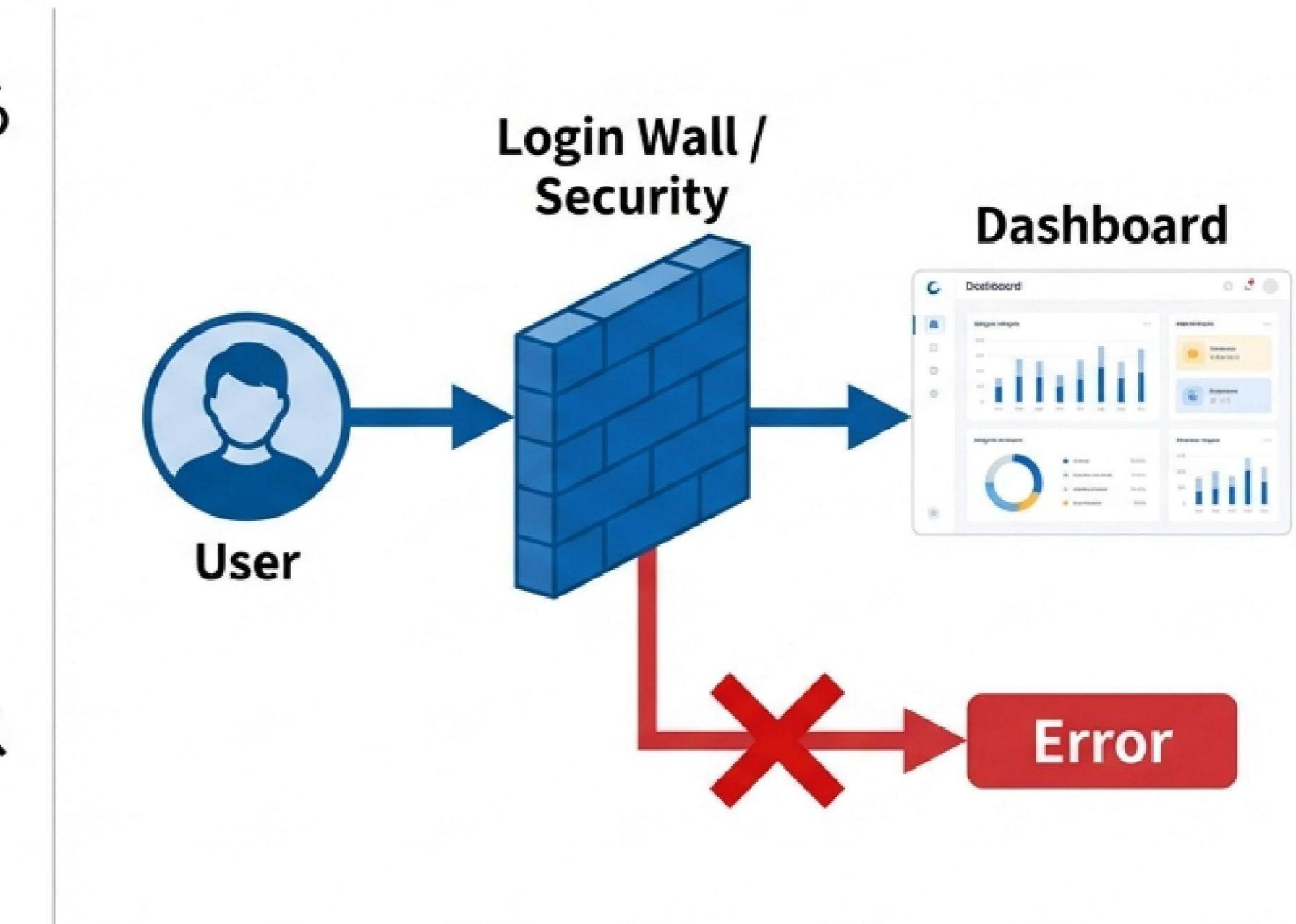
Secure: Spring Securityによる
ログイン認証の実装。



Role: 一般ユーザーと管理者
(Admin) の権限の使い分け。



Verify: JUnitを用いたサービス
ロジックの自動テスト。



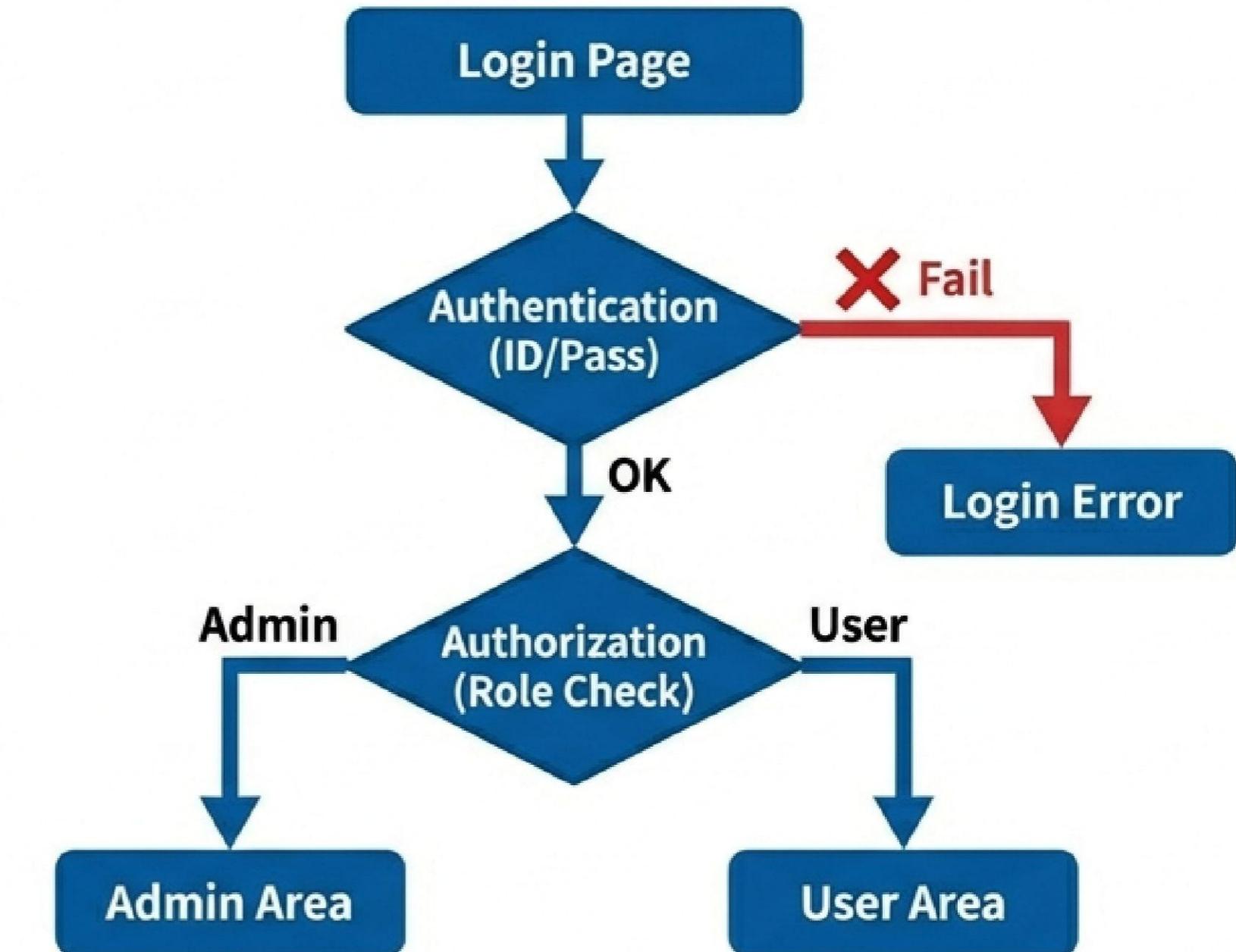
Spring Securityの概要

認証 (Authentication)

「あなたは誰ですか？」
IDとパスワードで本人確認を行う
プロセス。

認可 (Authorization)

「何ができますか？」
一般ユーザーか管理者か、役割
(Role) に応じたアクセス制御。



依存関係の追加

① spring-boot-starter-security

ログイン・ログアウト機能、認証基盤を提供。

② spring-boot-starter-validation

入力チェック（バリデーション）機能。

③ spring-boot-starter-test

JUnitを含むテスト用ライブラリ（testスコープ）。

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

SecurityConfig - 権限設定

requestMatchers

- `/admin/**`: 管理者 (ADMIN) のみ
アクセス可能。
- `/login`、`/css/**`: 誰でもアクセス
可能 (permitAll)。
- その他: 認証が必要
(authenticated)。

```
http.authorizeHttpRequests(auth -> auth
    .requestMatchers("/admin/**").hasRole("ADMIN")
    .requestMatchers("/login", "/css/**").permitAll()
    .anyRequest().authenticated()
);
```

formLogin

ログイン画面の挙動設定。

User管理 - データとロジック

① パスワードのハッシュ化

セキュリティ上の理由から、生パスワードは保存しない。BCryptなどでハッシュ化してDBに保存。

```
// Password Encoding
String encodedPassword = passwordEncoder.encode(rawPassword);
user.setPassword(encodedPassword);

// User Entity
@Column(unique = true)
private String username;
```

② UserService

ログイン時のユーザー情報取得処理と、ユーザー登録時の重複チェック。

管理者機能の実装

AdminAttendanceController

管理者専用のコントローラー。

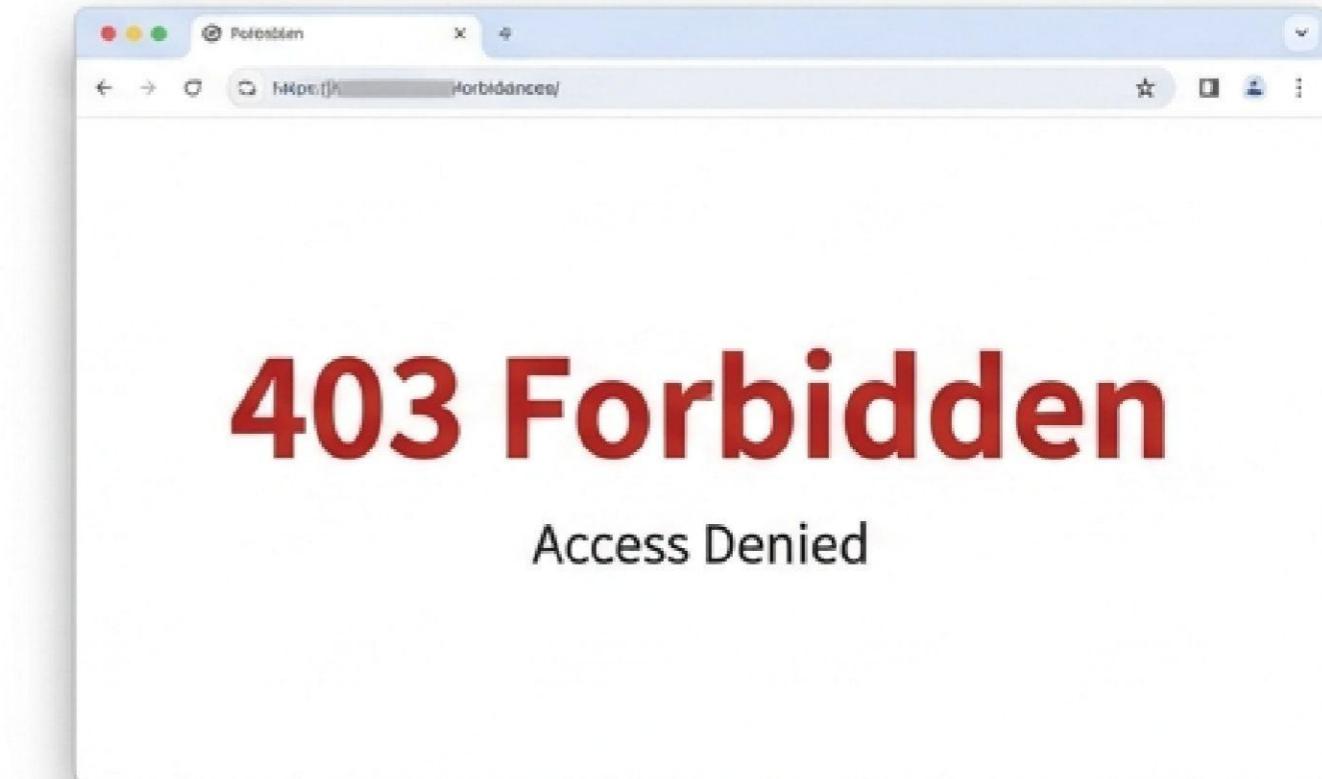
アクセス制御

一般ユーザーが`/admin`配下にアクセスすると、Spring Securityがブロックする。

403 Forbidden

権限不足を表すHTTPステータスコード。

```
@RequestMapping("/admin/attendances")
public class AdminAttendanceController { ... }
```



画面表示の制御

sec:authorize

Thymeleafでセキュリティ状態に応じて表示を切り替える属性。

管理者のみに「ユーザー管理」メニューを表示。

ログイン中のユーザー名を表示。

```
<div sec:authorize="hasRole('ADMIN')">
    <a href="/users">ユーザー管理</a>
</div>
```



自動テスト (JUnit)

テストの目的

- 修正時の安心感を担保する（リグレッション防止）。
- 手動確認の手間を減らす。

検証内容

- 正常系：出勤が成功すること。
- 異常系：二重出勤でエラーになること。

```
@Test  
AttendanceServiceTest.java  
  
void clockIn_success() {  
    // Setup & Execute  
    service.clockIn(user);  
    // Verify  
    verify(repository, times(1)).save(any());  
}
```



動作確認シナリオ

1. ログイン確認: user1 (一般) / admin (管理者) でログイン可能か。
2. 権限確認: user1 で `/users` にアクセス -> 403エラー (成功)。
3. 権限確認: admin で `/users` にアクセス -> 一覧表示 (成功)。
4. テスト実行: `mvn test` コマンドで全テストがPassすること。

User1

Access Denied

403 Forbidden

Me nim-nahimted win awern.aocor Access Denied | Error Forbidden.

Admin

ID	Name	Role	Action
1	user1	ADMIN	
2	user2	user1	
3	admin	ADMIN	
4	admin	admin	
5	felpo	Admin	
6	Mainirhtarmana	Admin	

つまずきポイント

よくあるエラー

- 口グインできない：DBにデータが入っているか確認（DataSeeder）。
- 403エラーが消えない：DBのロール名が`ROLE_ADMIN`になっているか確認。
- 依存関係エラー：Mavenキャッシュの不整合。`.m2` フォルダのクリーンアップが必要。

```
$ mvn -U clean spring-boot:run
```

```
SELECT * FROM users;
```

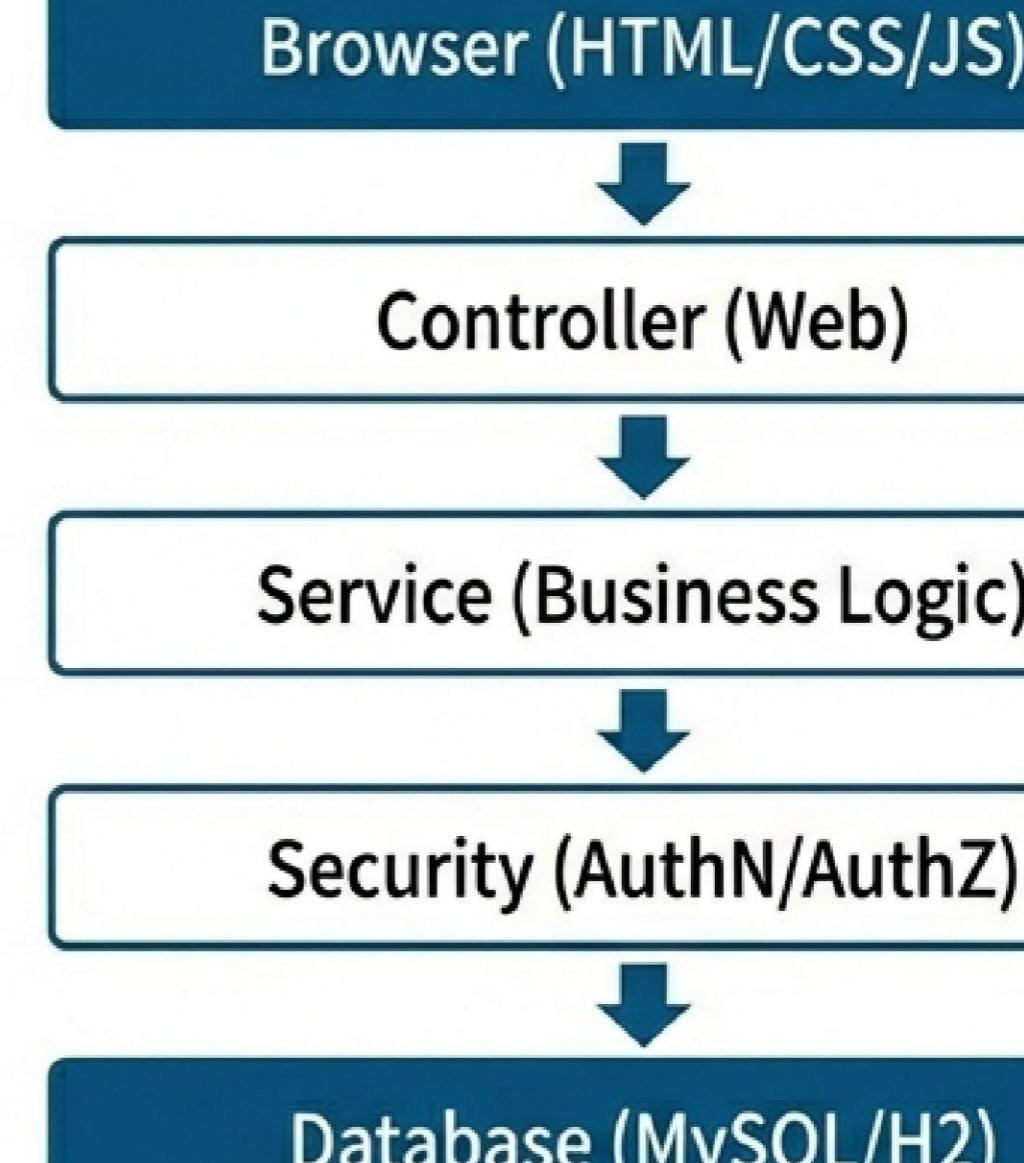
id	username	role
1	admin	ROLE_ADMIN

研修全体のまとめ

Course Timeline

- Day 0: HTML/Java基礎
- Day 1: Spring Boot & MVC
- Day 2-3: Database & Business Logic
- Day 4: List Views
- Day 5: Security & Testing

Full Stack Result



Webアプリケーション開発の基礎習得、お疲れ様でした。