フレームワーク応用実習

04. ユーザー認証

株式会社ジードライブ

今回学ぶこと

- ユーザー認証の種類
- データベースを利用したユーザー認証

ユーザー認証

- ユーザー認証とは、サイトにアクセスしてきているユーザーを確認すること
  - 特定のページやコンテンツへのアクセスを制御することができる

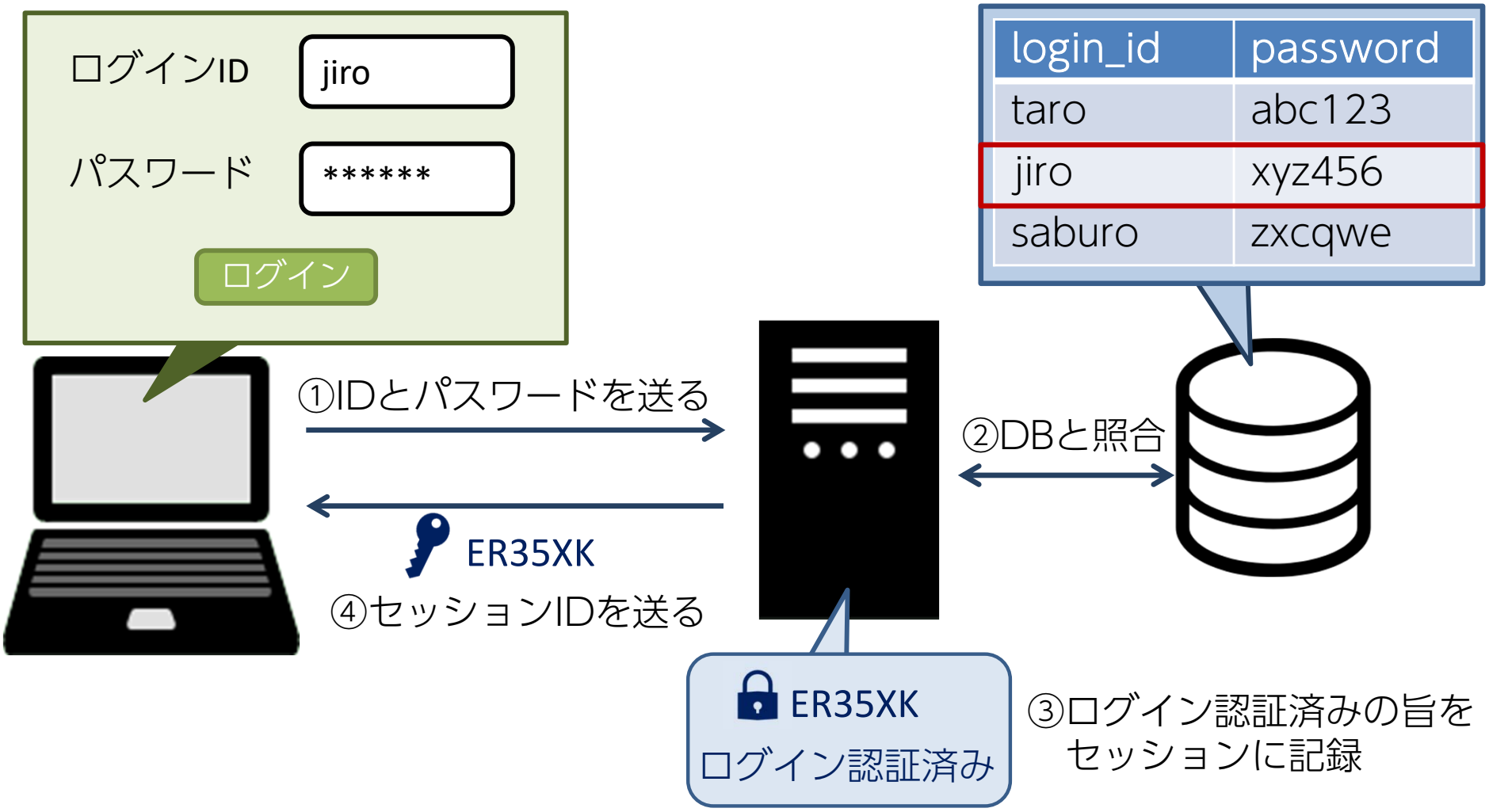
認証の方法	説明
BASIC認証	HTTPで定義される認証方法。未公開のサイトに対する簡易的な閲覧制限などに利用することができる
SNS認証	X(旧Twitter)やFacebookといったSNSアカウントによって認証を行う。ユーザーにとっては、新たにユーザーIDを作成し、管理するという手間を省くことができ、利用しやすい
独自の認証	JavaとDBを組み合わせてログイン機能を作成し、ユーザー認証を行う。システム管理画面や会員専用ページへのアクセス制限などを行う

Javaを利用した独自の認証

① ログインIDとパスワードをデータベースに保管しておく
② ログインフォームを用意し、ログインIDとパスワードが、DBに保管されているものと一致するかを確認する
③ IDとパスワードに問題がなければ、セッションに認証済みである旨を記録する
④ ログインが必要なページでは、セッションに認証済みである旨が記録されているか否かを確認する
  - 認証済みでない場合はエラーページを表示したり、ログインページへリダイレクトしたりする

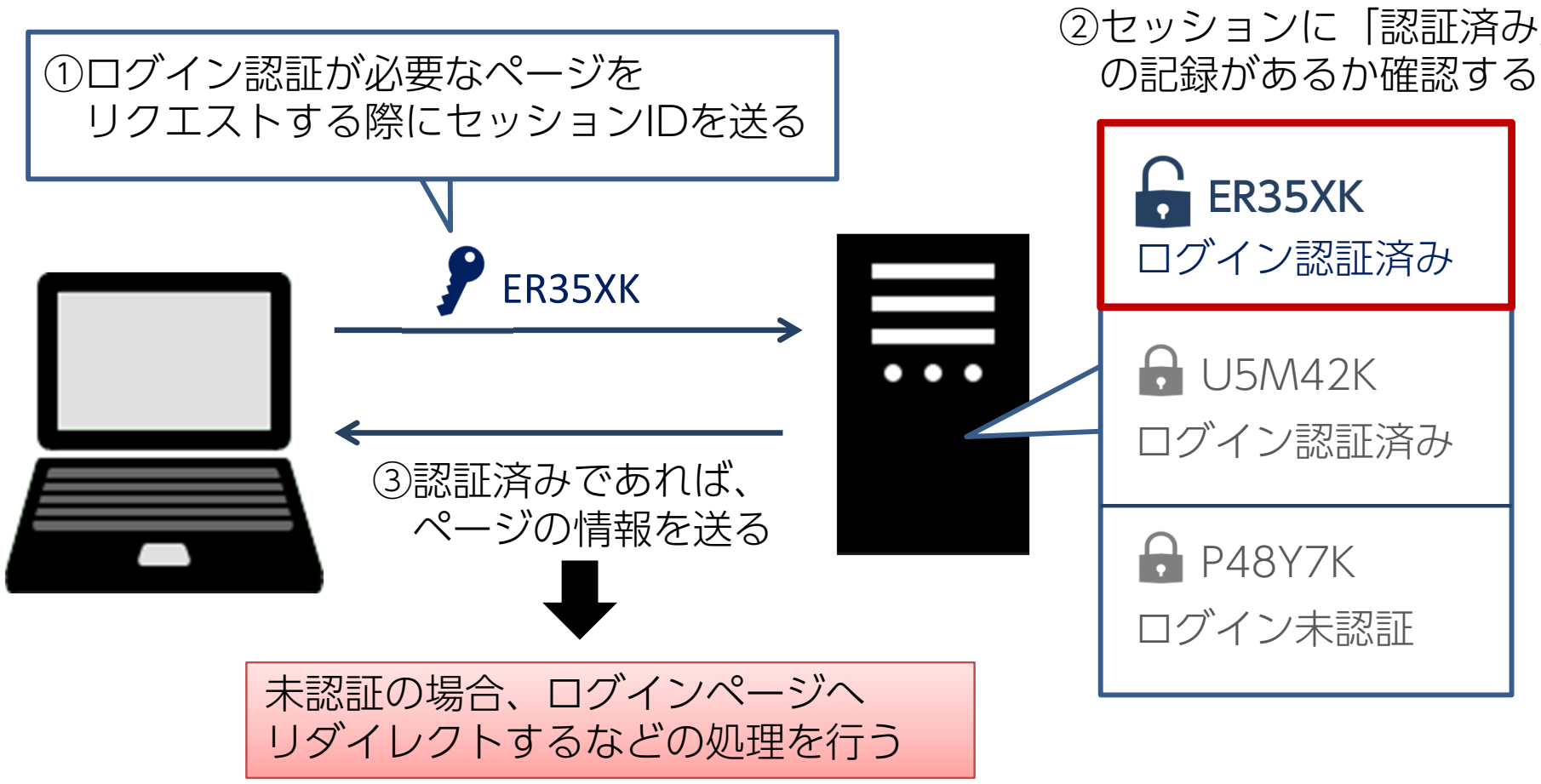
セッションを利用した独自の認証

- ログインの認証の流れ



セッションを利用した独自の認証

- ログイン認証の確認



認証情報を格納するテーブル

- 認証情報を格納するテーブルには、ログインIDとパスワードのカラムが最低限必要
  - システムの要件として、ユーザーに関する他の情報(ユーザーの種別など)がある場合はそれらのカラムも用意する

例：loginテーブル

login_id	password
taro	catchABall123
newton	fallingRedApple789
kaguya	princessBamb00!

パスワード情報の保護

- パスワードをそのままの文字列 (平文：ひらぶん) でデータベースに保存するのは危険
  - 万が一データベースの中身が第三者に漏えいした場合に、ユーザーのパスワードが知られてしまう
  - 他のサイトでも同じパスワードを使い回しているユーザーが多い現状では、１つのパスワードが漏えいすると、そのユーザーが利用する他のサイトでも「なりすまし」の被害に遭う可能性がある

パスワード情報の保護

- パスワードは何らかの暗号化、または**ハッシュ化**(不可逆な変換)を行って保存するのが望ましい
- 一般的には、**ハッシュ関数**を使い、ある文字列から復元不可能な文字列を生成してデータベースに保存する
  - この方法を使ってパスワードを保存しておくと、サイト管理者でもユーザーのパスワードを知ることはできない

login_id	password
taro	cc3336549cf5473a01b59fa301b4ce298fcdb962988c32d59a1c3eefa7d5366c ↑「catchABall123」をハッシュ化したもの

- ハッシュ関数の例
  - MD5, SHA-1, SHA-2, PBKDF2, **BCrypt**, HMAC

ハッシュ関数の例

- MD5
  - 任意の長さの文字列から、128bit (16進数で32桁)のハッシュ値を生成する関数

    例：hello という文字列のMD5値(16進数表記)

    ⇒ 5d41402abc4b2a76b9719d910017c592
- SHA-1
  - 任意の長さの文字列から、160bit (16進数で40桁)のハッシュ値を生成する関数

    例：「hello」という文字列のSHA-1値 (16進表記)

    ⇒ aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d

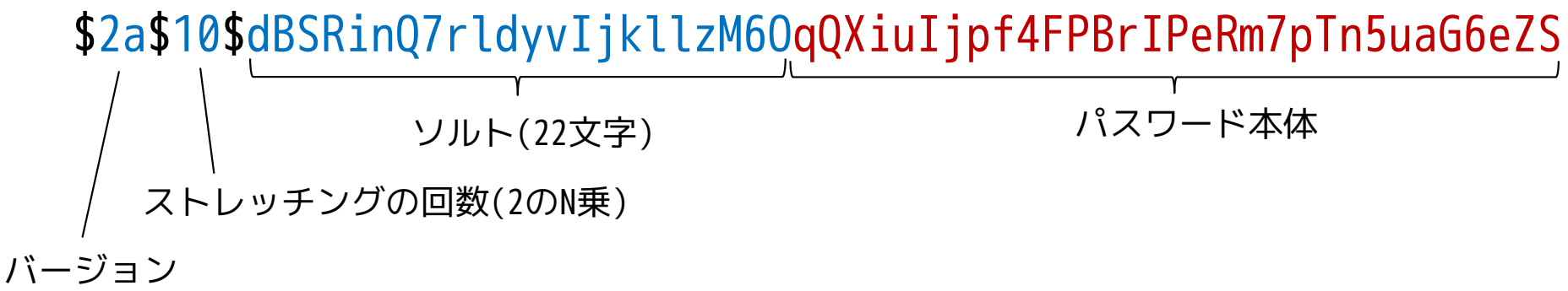
ハッシュ関数の選定

- 従来はパスワードをハッシュ化する方法として MD5, SHA-1, SHA-2などが使われてきた
  - MD5やSHA-1は脆弱性が発見された
  - SHA-2も「総当たり攻撃」がしやすいなどの弱点がある
- 現在では、よりパスワードのハッシュ化に適した関数(PBKDF2, BCrypt, HMAC)を使用することが推奨されている

BCrypt

- Blowfishという暗号化方式を利用したハッシュ関数で、**60字**の文字列が生成される
  - BSD(Unixをベースとした OS)を始めとした様々なシステムでの認証ハッシュ関数として利用されている
  - 元になる文字列が同じであっても、計算するたびに出力される文字列が異なる

例：「abcd」をBCryptでハッシュ化したもの



Bcryptハッシュの計算：https://toolbase.cc/text/bcrypt

jBCryptの導入

- Javaのコード内でBcryptのハッシュ関数を使いたい場合は、jBCryptというライブラリを導入するとよい
  - より包括的なセキュリティ強化を望む場合は、Spring Securityの導入を検討する

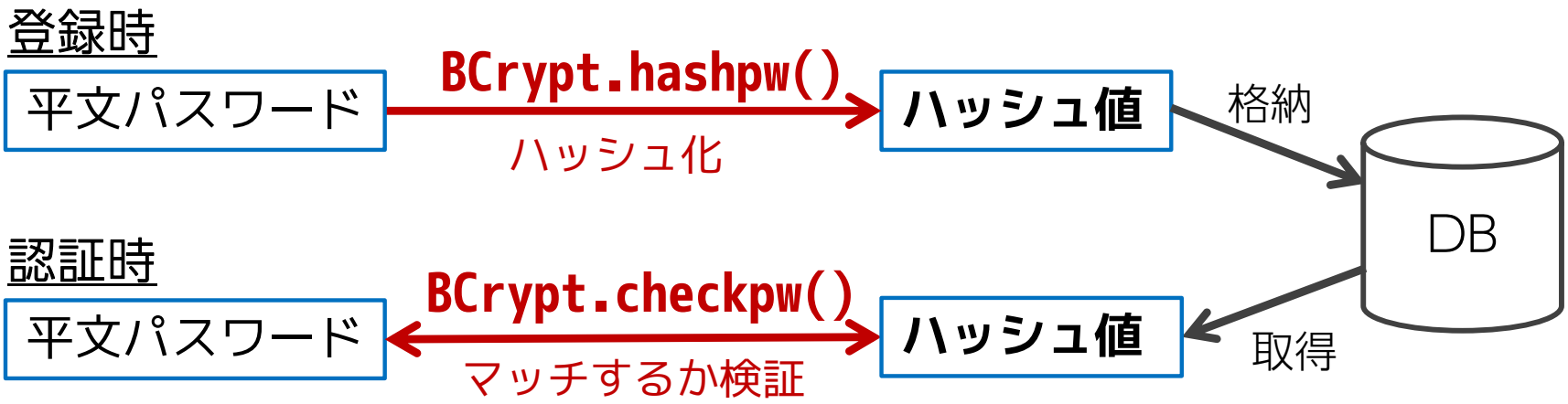
jBCryptを導入する際のpom.xmlへの追記

```
<dependency>
    <groupId>de.svenkubiak</groupId>
    <artifactId>jBCrypt</artifactId>
    <version>0.4.3</version>
</dependency>
```

https://mvnrepository.com/artifact/de.svenkubiak/jBCrypt/0.4.3

jBCryptの利用

- DBにパスワードを登録する際は、BCrypt.hashpw()メソッドを使い、パスワードをハッシュ化する
- 認証時は、ユーザーが入力した平文のパスワードとDBから取り出したハッシュ化されたパスワードが一致するかをBCrypt.checkpw()メソッドで検証する



登録時の記述例

```
@PostMapping("/register")
public String register(@RequestParam String userId,
                       @RequestParam String pass) {

    // パスワードをハッシュ化
    String hashed = BCrypt.hashpw(pass, BCrypt.gensalt());

    // ハッシュ化されたパスワードをデータベースに登録
    userMapper.insert(userId, hashed);
    ...
```

平文のパスワード

ハッシュ化されたパスワード

ソルトの生成

認証時の記述例

```
@PostMapping("/login")
public String login(@RequestParam String userId,
                    @RequestParam String pass) {

    // ユーザーIDを元にDBからパスワード情報を取得
    // → DB内のパスワードはハッシュ化されている
    String hashed = userMapper.selectPasswordByUserId(userId);

    if(BCrypt.checkpw(pass, hashed)) {
        System.out.println("認証成功");
        ...
```

平文のパスワード

ハッシュ化されたパスワード

練習

- 練習04