フレームワーク基礎実習

07.クッキーとセッション

株式会社ジードライブ

今回学ぶこと

- データを保持する方法
- クッキーの利用
- セッションの利用

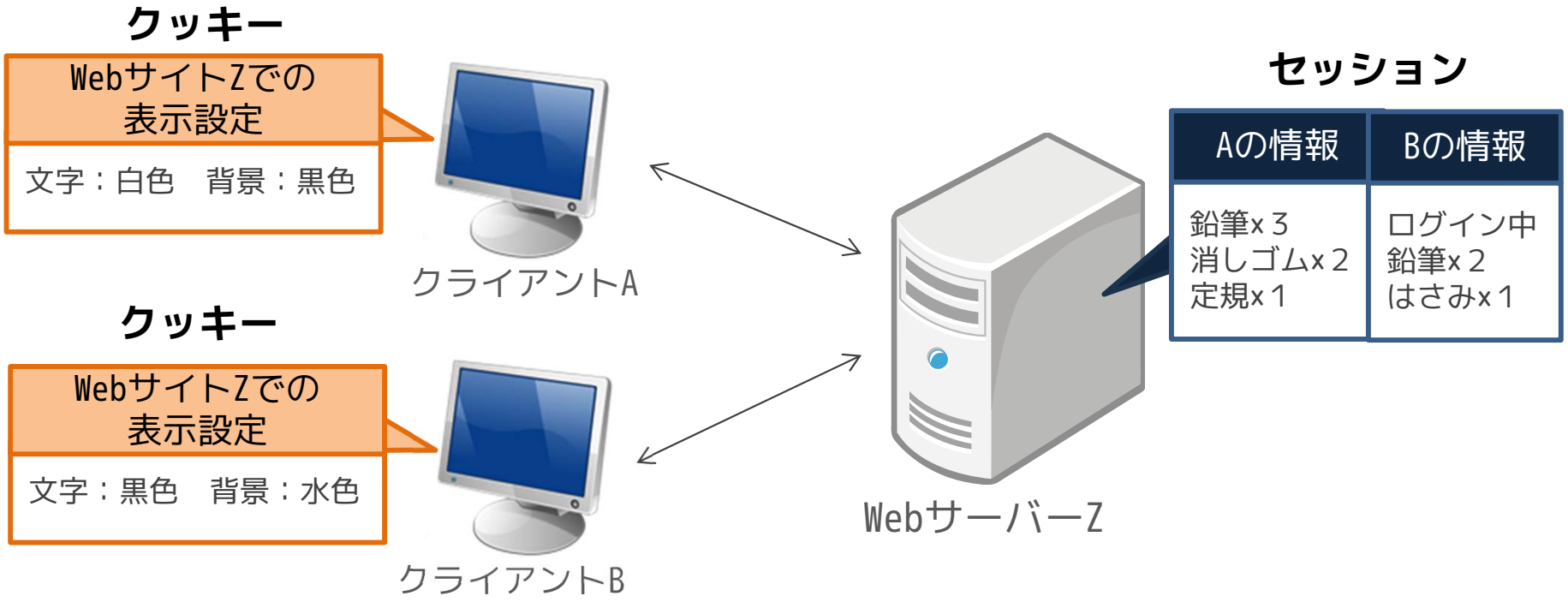
データを保持する方法

- 複数のページ(複数のリクエスト)をまたいで、共通のデータを利用するには、データを保持しておく仕組みが必要になる
- データを保持するには、以下のような方法がある
  - ファイル
  - データベース

  → 長期的に利用するデータ
  - クッキー → 中期的に利用するデータ (言語や文字サイズの設定等)
  - セッション → 一時的に利用するデータ (ショッピングカート等)

データを保持する方法

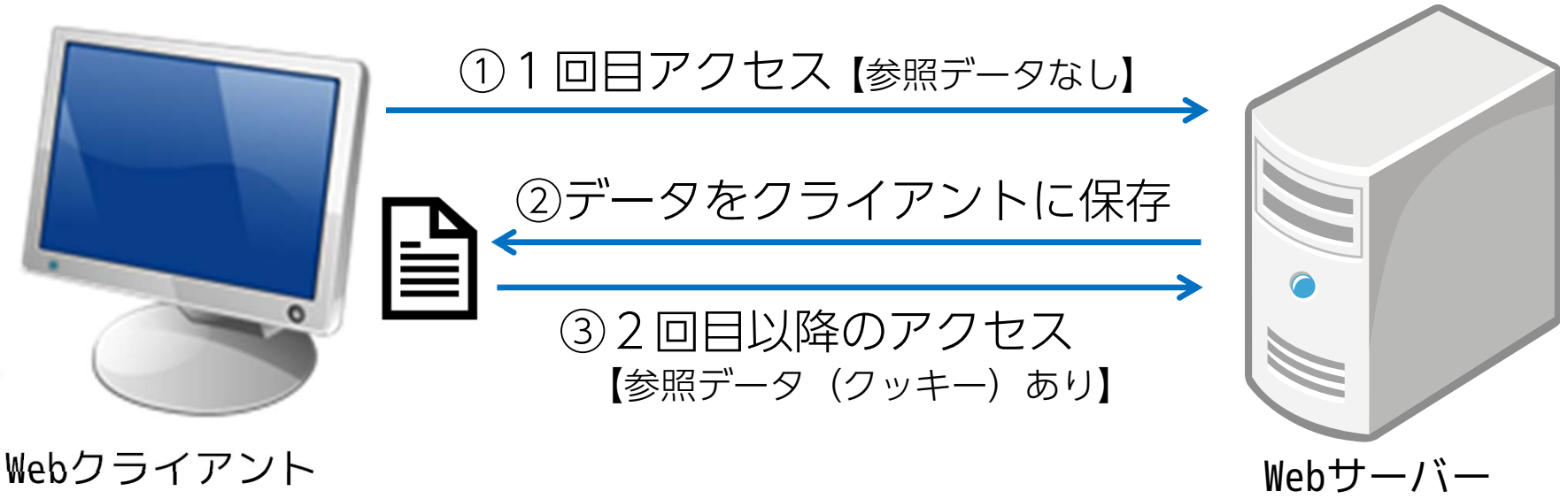
- クッキー
  - 個々のWebクライアント(ブラウザ)にデータを保存する仕組み
- セッション
  - 個々のクライアントに関する情報をサーバーに保存する仕組み



クッキーの利用

クッキーとは

- Webサーバーからクライアントにデータを送って保存できる仕組み
  - Webサーバーはこの保存データを参照することが可能



クッキーの操作

①クッキーをセットする(クライアントにクッキーを送り保存させる)

1. Cookieオブジェクトを作成し、設定をする
2. HttpServletResponse#addCookie()メソッドを使い、Cookieオブジェクトを登録する

②クッキーを参照する (クライアントのもつクッキーの内容を参照する)

- コントローラーメソッドの引数に@CookieValueアノテーションを付与する

①クッキーのセット(1)

- クッキー名と値を指定してjakarta.servlet.http.Cookieオブジェクトを生成し、有効期限などの設定をする

記述例

```
@GetMapping("/setting")
public String userSetting(HttpServletResponse res) {
  // colorという名前のクッキーに青色(#00f)をセットする
  Cookie cookie = new Cookie("color", "#00f");

  // 有効期限を7日後に設定する(秒数で指定)
  cookie.setMaxAge(60 * 60 * 24 * 7);

  // 「/user/」以下のディレクトリで有効とする
  cookie.setPath("/user/");
  ...
```

①クッキーのセット(1)

- Cookieクラスのおもなメソッド

メソッド	説明	デフォルト値
`void setMaxAge(int expiry)`	クッキーの存続期間を秒単位でセットする。**0だとクッキーを削除する。**負の値だとブラウザ終了時まで有効。	ブラウザ終了時まで有効
`void setPath(String uri)`	クッキーが有効になるパスを指定する。指定されたパスとそのサブディレクトリが有効範囲。	現在のパス
`void setDomain(String pattern)`	クッキーが有効になるドメインを指定する。	クッキー送信元のドメイン
`void setSecure(boolean flag)`	HTTPSなどSSLが有効な場合のみクッキーを有効にするかどうかを指定する。	`false`

①クッキーのセット(2)

- HttpServletResponse#addCookie()を使い、生成したCookieオブジェクトを登録する
  - HttpServletResponseオブジェクトはコントローラーメソッドの引数として指定する

記述例

```
@GetMapping("/setting")
public String userSetting(HttpServletResponse res) {
  Cookie cookie = new Cookie("color", "#00f");
  cookie.setMaxAge(60 * 60 * 24 * 7);
  cookie.setPath("/user");

  res.addCookie(cookie);

  return "redirect:/user";
}
```

②クッキーの参照

- コントローラーメソッドの引数に@CookieValueアノテーションを付与することでCookieの値を利用できる

```
@GetMapping("/user")
public String userHome(
        @CookieValue(defaultValue="#000") String color) {
            System.out.println(color)
            return "userHome";
}
```

- @CookieValueには、defaultValue(初期値)以外にも以下のような設定ができる

nameの代わりにvalueとしてもよい

```
@CookieValue(name="color" required=false) String color
```

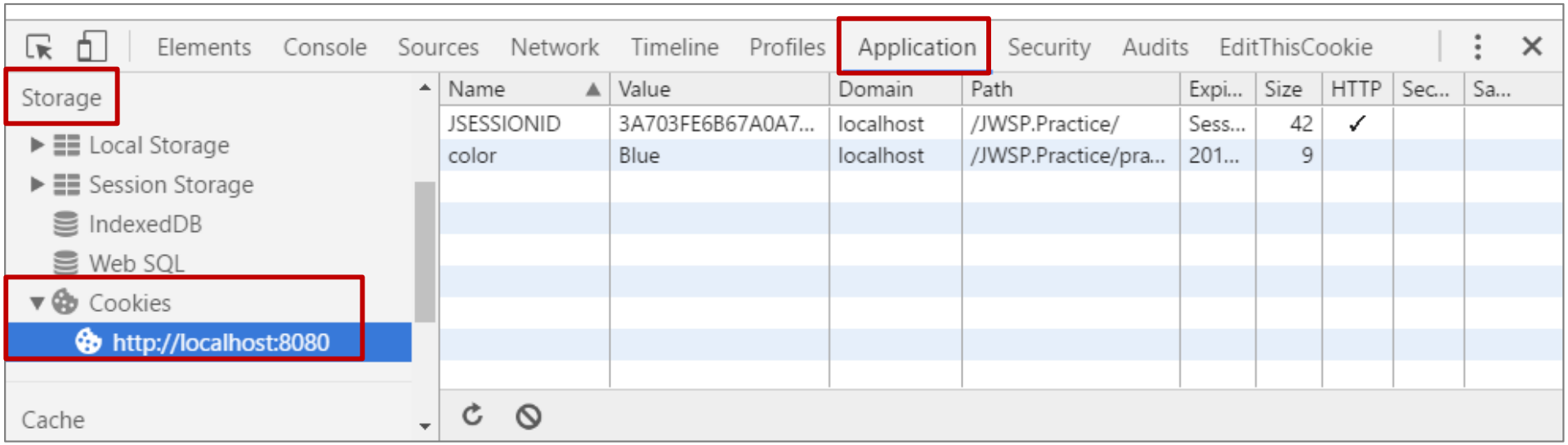
name="color" は省略可能。
クッキー名(青字)と引数名(緑字)が同じ場合は記述を省略できる

Chromeでのクッキー操作

クッキー情報の表示：

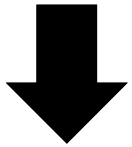
① F12キーでDeveloper Toolを開く

②「Applications」⇒「Storage」 ⇒「Cookies」



クッキー利用の注意点

- クッキーの内容は、HTTPSの通信でない場合、テキストデータがそのまま送られる
- Webブラウザによってはクッキーの内容を閲覧したり編集することが可能
- クッキーの内容はクライアントのコンピュータに保存される



クッキーにパスワードなどの重要な情報を
保存しないようにする

クッキーの制限

- 1つのクッキーにつき4096バイトまで保存可能
  - それ以上のデータの保存の可否はブラウザ依存
- 1つのホスト名につき50個のクッキーまで保存可能
  - それ以上のデータの保存の可否はブラウザ依存
- 1つのクライアント (例えばWebブラウザ) につき、3000個のクッキーまで保存可能
  - それ以上のデータの保存の可否はブラウザ依存

出典 RFC6265：https://tex2e.github.io/rfc-translater/html/rfc6265.html

練習問題

- 練習07-1

セッションの利用

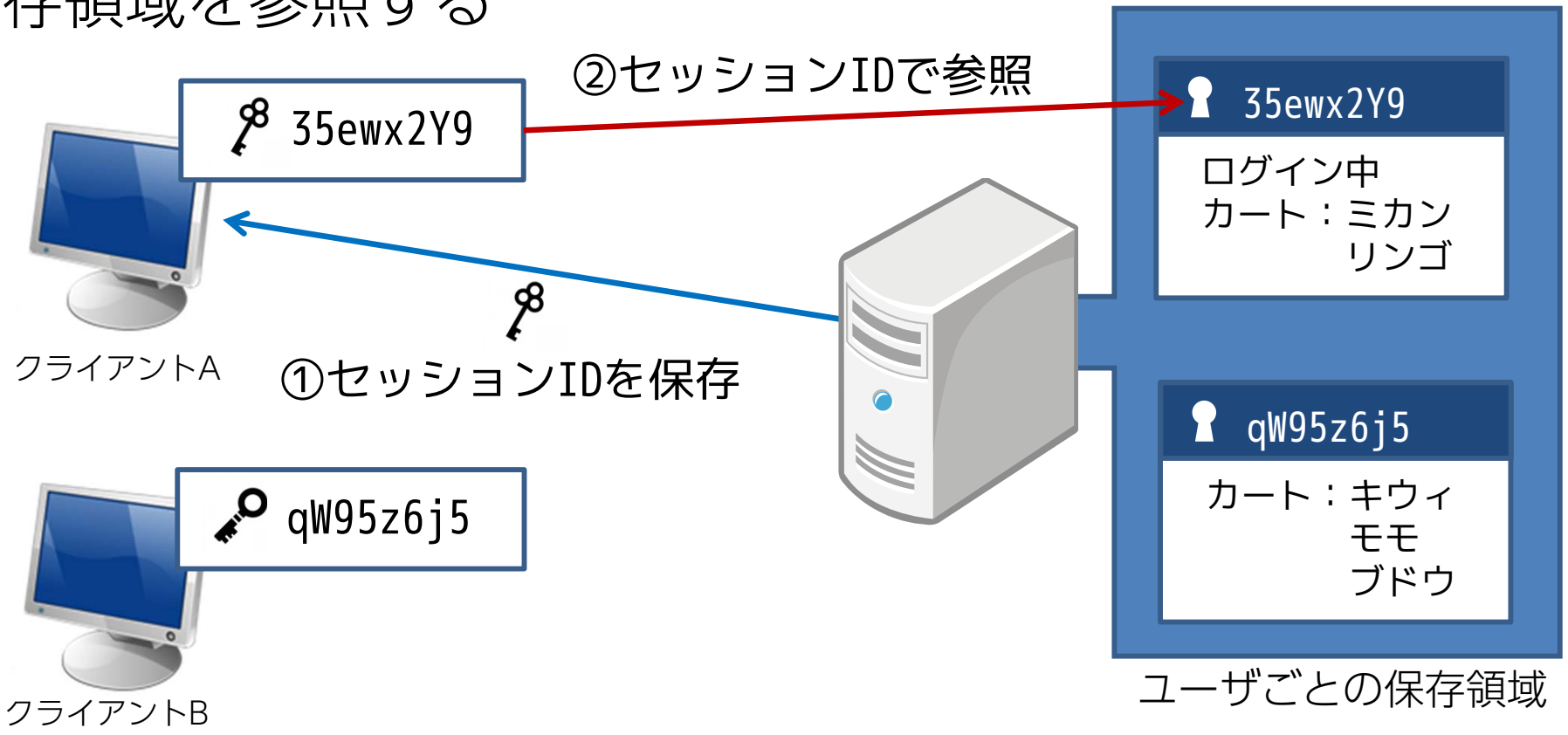
セッションとは

- Webサイトにアクセスしたユーザ（正確にはWebクライアント）についての情報を一時的に保持する仕組み
  - 情報はサーバに保存される
  - セッションIDを利用してクライアントを識別する
- ログイン認証の情報やショッピングカート内のデータなどの保持に利用される
  - 商品の決済処理など、ある目的を果たしたら破棄するデータを一時的に保持する場面で利用

セッションとは

①クライアントがサーバーにアクセスするとセッションIDが発行され、クッキーとして保存される

②セッションIDを使い、サーバー側にあるユーザごとの保存領域を参照する



セッションの操作

- セッションの操作には、HttpSessionオブジェクトを利用する

セッションの操作

① HttpSessionオブジェクトの取得
② セッションへのデータの保存
③ セッションからのデータの取得
④ セッションのデータ削除
⑤ セッションデータの表示

コントローラーでの処理

Thymeleafでの処理

①HttpSessionオブジェクトの取得

方法１：コントローラーのフィールドとして定義

```
@Controller
@RequiredArgsConstructor
public class RegisterController {
    private final HttpSession session;
    ...
```

方法２：コントローラーメソッドの引数として指定

```
@PostMapping("/register")
public String registerUser(HttpSession session,
   @RequestParam String name,
   @RequestParam int age) {
     ...
```

②セッションへのデータ保存

- データの保存はHttpSession#setAttribute()で行う

```
session.setAttribute(キー, 保存するデータ);
```

例：フォーム入力されたユーザー情報をセッションに保存し、確認画面へリダイレクト

```
@postMapping("/register")
public String registerUser(HttpSession session,
    @RequestParam String name,
    @RequestParam int age) {
      session.setAttribute("userName", name);
      session.setAttribute("userAge", age);
      return "redirect:/register/conf";
}
```

③セッションからのデータ取得

- データの保存はHttpSession#getAttribute()で行う

```
Object obj = session.getAttribute(キー);
```

例：セッションに保存されているユーザー情報の取得

```
@GetMapping("/register/conf")
public String confirmInfo(HttpSession session) {
   //getAttributeの戻り値はObject型なので、必要に応じてキャストする
    String name = (String) session.getAttribute("userName");
    int age = (int) session.getAttribute("userAge");
    System.out.println(name + " " + age + "才");
    return "confirm";
}
```

④セッションのデータ削除

方法１：個別のデータ削除

```
session.removeAttribute(キー);
```

方法２：全データの削除

```
session.invalidate();
```

例：セッションに保存されているユーザー情報の破棄

```
@GetMapping("/register/done")
public String removeInfo(HttpSession session) {
    session.removeAttribute("userName");
    session.removeAttribute("userAge");
    return "redirect:/home";
}
```

⑤セッションデータの表示

- セッションデータをThymeleaf内で参照する場合は、暗黙オブジェクトsessionを利用する
  - 暗黙オブジェクトは変数宣言せずに利用できるオブジェクト

例：登録内容の確認ページでセッション内のデータを表示させる

```
<h1>登録内容確認</h1>
<table border="1">
    <tr>
        <th>名前</th>
        <td>[[${session.userName}]]</td>
    </tr>
    <tr>
        <th>年齢</th>
        <td>[[${session.userAge}]]</td>
    </tr>
</table>
```

RedirectAttributesオブジェクト

- リダイレクト先で**一度だけ**使用するようなデータは、HttpSession#setAttribute()ではなく、RedirectAttributes#addFlashAttribute()を利用して保存すると便利
  - リダイレクト先では、Model#getAttribute()メソッドでデータを取り出すことができる。また、EL式での利用も可能

```
@PostMapping("/add")
public String add(RedirectAttributes ra) {
    …商品の追加処理…
    ra.addFlashAttribute("tempMsg", "商品を追加しました");
    return "redirect:/list";
}
```

同様の処理を、HttpSessionオブジェクトで実装する場合、removeAttributeの記述が必要になる

セッションの有効期間

- セッションデータの有効期間は30分に設定されている
  - クライアントからのアクセスがない状態で有効期間を経過するとサーバー側でセッションが自動的に破棄される
- 有効期限を変更するには、application.propertiesに追記を行う

例：セッションの有効期限を60分に変更する

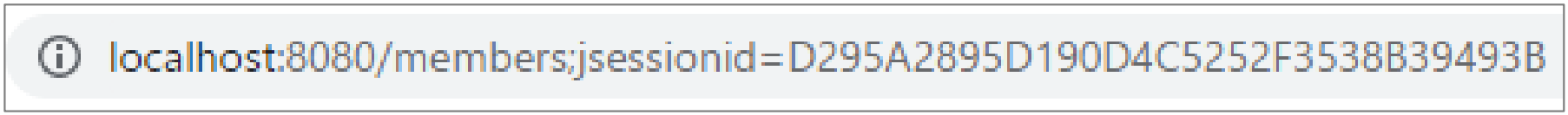
```
# セッションの有効期限設定: 3600秒(1時間)
server.servlet.session.timeout=3600

# セッションに紐づくクッキーの有効期限の設定
server.session.cookie.max-age=3600
```

jsessionid

- JSESSIONIDは、セッションIDをクッキーで管理するときの名称(サーブレットの仕様として決められている)
- クッキーが利用できない場面では、URL末尾のパラメーターとしてセッションIDを管理することができる
  - このときのパラメーター名はjsessionidと決められている
  - この方法は推奨されていないため、基本的にクッキーで管理する

- セッションを利用する場面(ログイン等)で、初回アクセス時にURL末尾にjsessionidが表示されてしまう場合があり、これによって例外が発生してしまうことがある



jsessionid

- セッションのクッキー利用を設定することで、問題を解消できる

方法１： application.propertiesで対応する方法

```
server.servlet.session.tracking-modes=cookie
```

方法２： 設定ファイル(@Configurationが付与されたクラス)で対応する方法

```
@Bean
ServletContextInitializer servletContextInitializer() {
  return servletContext -> servletContext
            .setSessionTrackingModes(
                  Collections
                  .singleton(SessionTrackingMode.COOKIE));
}
```

練習問題

- 練習07-2
- 練習07-3