4-3-8.Webスコープについて

はじめに

JavaでつくるWebページについて学んできていますが、
ページが1ページだけで終わるという事はなかなかないですよね。
たくさんの情報を載せたいのでWebページは複数にわたります。
複数あるページを扱う時、ページが切り替わるたびにログインIDや
入力情報を求められたら大変ですね。
今回はページをまたいでも入力情報を保持しておける仕組みを学んでいきます。

Step1: 概念を知る

Webスコープ
サーブレットやJSPをまたいでデータを共有したい場合に用いる保存領域のこと。

共有範囲に応じて以下の3種類がある。

スコープ名	クラス名	内容
application	ServletContext	Webアプリケーション自体を指し、最も広い有効範囲を示します。 アプリケーション・サーバーが終了しない限り保持されるスコープ。 ブラウザを閉じても保持されます。 【使用例】： マスターデータのキャッシュやアクセスカウント数の保管。
session	HttpSession	ある特定のWebクライアントとWebサーバの通信を1つの単位として扱い、 その通信が閉ざされるまでの間を有効範囲とします。 ブラウザを閉じない、あるいはタイムアウトしない限り保持されます。 【使用例】： ショッピングカード内容の保存や、 各ユーザーのログイン／ログアウト状態の記録。
request	HttpServletRequest	クライアント（ブラウザ）から1回のリクエストの間だけを有効範囲とします。 ブラウザからのリクエスト毎に生成されます。 次のリクエストが来た場合やブラウザを閉じたり、 アプリケーション・サーバーが再起動した場合は原則保持されません。 【使用例】： ログインIDやパスワード情報の保管。

>

セッションは、どこかの場所に入るための入館証のようなイメージです。

セッションの開始（= **入館証を渡された** タイミング）
セッションの終了（= **期限切れで無効の** タイミング）
となります。

・セッション切れ
セッション（= 入館証の有効期限）としています。
有効期限が過ぎた入館証で会場へのアクセス（入館証を読み取り機器にあてる）する行為は、
無効 で、おそらく、その入館証は無効になります。
(このエラーがいわゆる「 **セッションタイムアウト** 」ということです)

?

Step2: 使い方を知る

下記にWebスコープの **セッション** を使用した属性を設定する例を示します。

各Webスコープに属性を設定します。
属性はWebスコープの有効範囲内で維持され、必要に応じて設定した属性の値を取得することが可能です。
各Webスコープを表すクラスを設定します。
クラスそれぞれに用意されている、
`setAttribute()メソッド` と `getAttribute()メソッド` を用いて属性の設定や取得を行います。

```
HttpSession session = request.getSession();
session.setAttribute("NAME", "JobSupport");
```

セッションは、HttpServletRequestクラスのオブジェクトの `request` から `getSession()メソッド` で生成します。
そのセッションの `setAttribute()メソッド` で第1引数に属性の名前を文字列で与え、第2引数で属性値を与えます。
これで属性の設定となります。
上記の例では、属性値に `JobSupport` という文字列を与えていますが、
属性値はJavaのオブジェクトであれば何でも与えられます。
例えば、自作のクラスをオブジェクトにして属性値して与えることも可能です。

別のサーブレットにてセッションに設定した属性値を取得したい場合は、下記のようになります。

```
HttpSession session = request.getSession();
String value = (String)session.getAttribute("NAME");
```

属性を設定する時と同様に、セッションを生成します。
そのセッションから属性値を取得するには `getAttribute()メソッド` で引数に属性の名前を文字列で与えることで、
対応する属性値が取得できます。

> **ポイント**

`getAttribute()メソッド` 使用時の注意点ですが、
`getAttribute()メソッド` の戻り値の型は **Object型** となります。
そのため、特定のクラスの型に代入するにはキャストする必要があります。

・キャスト
元の変数の型はそのままで、一時的に宣言時とは別の型に置き換えること。

`session.getAttribute("NAME");` の際に必要になり、
ここでは **String** にキャストしています。

取得する値の型と、キャストする値の型が不一致の場合は、`ClassCastException` という例外が発生しますので、
`setAttribute()` で格納した値の型と必ず一致するようにキャストしましょう！

例外処理を施したり、
`instanceof` という **変数の型をチェックする演算子** を用いてキャスト時の対応を行ったりしてもよいのですが、
プログラム中の値の取り扱いや、運用後の保守性を鑑みると、
基本的には **同一のキー名で、複数の型を格納する** なんてことは避けるべきですので、
「 `setAttribute()` を用いて格納する **値・型・キー名は一意** となるよう開発以前の仕様として定めておく ようにしましょう。

課題

提出課題はありませんので、一通り学習が終わったら次の章に進んで下さい。

最終更新日時: 2022年 09月 10日(土曜日) 06:26