Forms and cookies in JSP

Martin Pola

HTML frontend

A form can be created in pure HTML, with a range of pre-defined types of input fields.

- Line input
- Multiple line input (`textarea`)
- Radio buttons
- Checkbox
- Dropdown (`select`)

HTML frontend

A form can be created in pure HTML, with a range of pre-defined types of input fields.

- Line input
- Multiple line input (`textarea`)
- Radio buttons
- Checkbox
- Dropdown (`select`)

In order to handle the data once submitted, one would usually have some server-side logic. That cannot be pure HTML.

A sample form

```
1 <form action="form-handler.jsp">
2     Enter your full name:
3     <input type="text" name="fullname" />
4
5     Enter your age:
6     <input type="number" name="age" min="0" />
7
8     <input type="submit" value="Continue" />
9 </form>
```

A sample form

```
1 <form action="form-handler.jsp">
2     Enter your full name:
3     <input type="text" name="fullname" />
4
5     Enter your age:
6     <input type="number" name="age" min="0" />
7
8     <input type="submit" value="Continue" />
9 </form>
```

- The `action` attribute in the `form` tag specifies where the browser should go upon submission
- The `name` attribute in the `input` tag can be used to obtain the value at the server (with JSP)

Handle form submission

Look at the URL in the browser when the form has been submitted:
`...:8080/form-handler.jsp?fullname=Martin+Pola&age=24`

The part after the question mark (?) is called the *query string*.

```
1  <%
2  String fullName = request.getParameter("fullname");
3  String age = request.getParameter("age");
4
5  int ageInt = Integer.parseInt(age);
6
7  out.print("Hello " + fullName + "<br />");
8
9  if (ageInt < 13 || ageInt > 19) {
10     out.print("You're not a teenager.");
11 } else {
12     out.print("You are a teenager.");
13 }
14
15 %>
```

From stateless to stateful

HTTP is *stateless*. Each request is handled individually and multiple requests from the same browser are not linked together.

Sometimes, a stateful design is desirable – e.g. to keep track of a user who is logged in.

Cookies let us save data in the browser, and have the browser return the data with every sub-sequent request.

- First, the server sends a `Set-Cookie` header to save cookie data
- On every subsequent request, the web browser includes a `Cookie` header containing previously saved cookies

Saving a cookie

```
1  <%
2
3  // [...]
4
5  if (ageInt < 13 || ageInt > 19) {
6      out.print("You're not a teenager.");
7  } else {
8      out.print("You are a teenager.");
9  }
10
11 // create a cookie
12 Cookie c = new Cookie("fullname", fullName);
13
14 // ask the browser to save it for seven days
15 c.setMaxAge(60 * 60 * 24 * 7);
16
17 // make the cookie part of the web server's response
18 response.addCookie(c);
19
20 %>
```

Reading a cookie

```
1  <%
2
3  // we might have a cookie with a full name
4  String savedFullName = null;
5
6  Cookie[] cookies = request.getCookies();
7
8  if (cookies != null) {
9      for (Cookie cookie : cookies) {
10         if (cookie.getName().equals("name")) {
11             savedFullName = cookie.getValue();
12             break;
13         }
14     }
15 }
16
17 if (savedFullName != null) {
18     out.write("Hello " + savedFullName + "!");
19 }
20
21 %>
```