How the Catan Server Uses HTTP Cookies

The Catan web service APIs accept JSON inputs and produce JSON outputs. These are passed in the HTTP request and response bodies. The one exception to this is that the Catan server uses HTTP cookies to track the user's identity and the game they are playing. This means that the JSON going back-and-forth between client and server will not contain the user's identity or game ID, because that information is passed through HTTP cookie headers instead of in the JSON request/response body.

If you are unfamiliar with HTTP Cookies, you can refer to the Wikipedia article titled "HTTP cookie" in order to learn more, or other online resources that you can find.

When the client calls the `/user/login` web service API, if the login succeeds, the server returns an HTTP response with a 200 success status code, and the string `Success` in the response body. The HTTP response also includes a `Set-cookie` header of the form:

```
Set-cookie:
catan.user=%7B%22name%22%3A%22Sam%22%2C%22password%22%3A%22sam%22%2C%
22playerID%22%3A0%7D;Path=/;
```

This header contains a name/value pair where the name is `catan.user` and the value is a URL-encoded JSON object containing information about the user. To process this cookie, do the following:

1. Read the `Set-cookie` header from the HTTP response using the `HttpURLConnection.getHeaderFields` method. The value of this header should look something like this:

```
catan.user=%7B%22name%22%3A%22Sam%22%2C%22password%22%3A%22sam%22%2C%
22playerID%22%3A0%7D;Path=/;
```

2. To get the value of the header, strip off the "catan.user=" on the front and the ";Path=/;" at the end. The remaining characters contain the URL-encoded JSON object that contains information about the user. The cookie's value should look something like this:

```
%7B%22name%22%3A%22Sam%22%2C%22password%22%3A%22sam%22%2C%22playerID%
22%3A0%7D
```

3. Decode the cookie's value by passing it to the `URLDecoder.decode` method. The decoded value should look something like this:

```
{"name":"Sam","password":"sam","playerID":0}
```

4. The resulting JSON object contains the user's name, password (I have no idea why), and their ID. This ID uniquely identifies the user, and is different than their "player index" (or playing order) in the game. It can be any integer.

5. Hang on to the original encoded cookie value, because later you will need to send it back to the server when making further web service requests in order to identify the user. In other words, hold on to the thing that looks like this:

```
%7B%22name%22%3A%22Sam%22%2C%22password%22%3A%22sam%22%2C%22playerID%
22%3A0%7D
```

6. When calling the `/games/join` web service API, include the user's identity in the HTTP request by adding a header of the form:

```
Cookie:
catan.user=%7B%22name%22%3A%22Sam%22%2C%22password%22%3A%22sam%22%2C%
22playerID%22%3A0%7D
```

Headers may be added to an HTTP request by calling the `HttpURLConnection.setRequestProperty` method.

7. If the `/games/join` request succeeds, the server returns an HTTP response with a 200 success status code, and the string `Success` in the response body. The HTTP response also includes a `Set-cookie` header of the form:

```
Set-cookie: catan.game=NN;Path=/;
```

where NN is an integer representing the joined game's ID. Extract the game ID, and hold onto it, because all subsequent web service calls require the game ID to be passed to the server so it knows which game the user is playing.

8. After successfully calling `/user/login` and `/games/join` all subsequent web service calls should include an HTTP `Cookie` header that includes both the `catan.user` and `catan.game` cookies. This is how the server knows which user is making the call, and what game they are playing. More specifically, to each request you should add a `Cookie` header of the form:

```
Cookie:
catan.user=%7B%22name%22%3A%22Sam%22%2C%22password%22%3A%22sam%22%2C%
22playerID%22%3A0%7D; catan.game=NN
```

where NN is the game ID.