Exercises: Session Tracking

1. Use session tracking to make a servlet that says "Welcome Aboard" to first-time visitors (within a browsing session) and "Welcome Back" to repeat visitors. If you did this same task earlier with the Cookie API, was this servlet harder or easier than the equivalent version using cookies explicitly?

2. Write a servlet that displays the values of the firstName, lastName, and emailAddress request parameters. But, remember what users told you in the past, and use the old values if the current values are missing. So, if a parameter is missing and the client is a first-time visitor, have the servlet list "Unknown" for the missing values. If a parameter is missing and the client is a repeat visitor, have the servlet use previously-entered values for the missing values. This should definitely be easier than a version that uses cookies explicitly.

3. Make a servlet that prints a list of the URLs of the pages that the current user has used to link to it (within the current browsing session). That is, if one user has followed hypertext links from three different pages to the servlet, the servlet should show the user the URLs of those three pages. Test out your servlet by making a couple of different static Web pages that link to it. If you feel inspired, modify the basic approach from the notes so that you do not store repeated entries; if the same user follows a link from page1 to the servlet twice, the servlet should list the URL of page1 only once in the list. Start with my ShowItems class, and make a few small changes such as getting the data from a request header (which one?) and changing doPost to doGet. If you are unfamiliar with the list-related data structures in Java, see the notes on the next page regarding the ArrayList class.

4. **[Medium hard, for the moderately inspired.]** The previous problem tracked the referring page and ignored repeated entries. In this problem, you should track a request parameter and count the repeats. Make a servlet that keeps track of the number of each item being ordered. For example, if the user orders yacht, car, book, yacht, the servlet should show something like this:
   - yacht (2)
   - car (1)
   - book (1)

   To simplify your code, you can just use the item name as sent from the HTML form; there is no need to check each name against a table of legal names as in the much-more-complicated Shopping Cart example in the book.

Note: Using the ArrayList Class

The java.util.ArrayList class is useful for keeping lists of items when you don't know how many items you will have. It has several key methods. Here is a summary that assumes that you are storing entries of type String. Change <String> to <YourType> if you store something other than Strings. ArrayList and LinkedList are two data structures that *all* Java programmers should know, so if you are not familiar with them, learn them more thoroughly later. Still, the following is probably enough for the purposes of this exercise.

- Constructor: no required arguments. E.g.

```
List<String> items = new ArrayList<String>();
```

(In Java 7, the above can be List<String> items = new ArrayList<>();)

- Adding items to end of list: call "add". E.g.

```
items.add("Item One");
```

- Looping down the list

```
for(String item: items) {
  doSomethingWith(item);
}
```

- Number of items in list: call "size". E.g.

```
int numItems = items.size();
```

- Get an item out of a specific location: "get". E.g.

```
String item = items.get(0);
```

- Testing if an item is already in the list: "contains". E.g.

```
if (items.contains("Item One")) ...
```