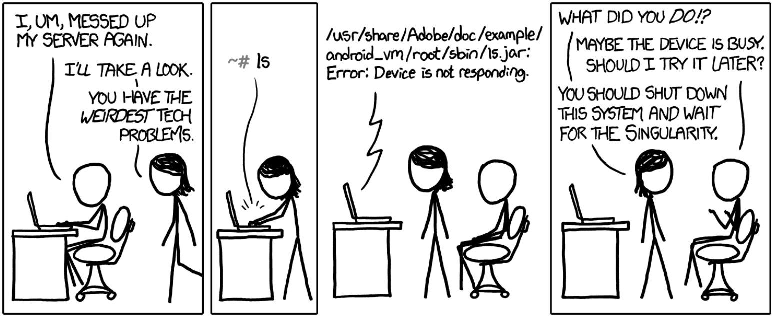
I, UM, MESSED UP MY SERVER AGAIN.
I'LL TAKE A LOOK.
YOU HAVE THE WEIRDEST TECH PROBLEMS.
~# ls
/usr/share/Adobe/doc/example/android_vm/root/sbin/ls.jar: Error: Device is not responding.
WHAT DID YOU DO!?
MAYBE THE DEVICE IS BUSY. SHOULD I TRY IT LATER?
YOU SHOULD SHUT DOWN THIS SYSTEM AND WAIT FOR THE SINGULARITY.

Credit: Randall Munroe xkcd.com

Lab 11: Servlets

CS 2112 Fall 2020

December 7 / 9, 2020

What is HTTP?

- HyperText Transfer Protocol (HTTP) is a protocol designed for client-server communication.
- It follows a request-response model, where the client can make requests of the server, and the server must respond to that request.
- The two most commonly used request methods are GET and POST.
- GET requests some information from the server, while a POST request submits some data.

GET Requests

- Query strings sent in URL
  i.e. `demo_form.php?name1=value1&name2=value2`
- Query strings represented as `key=value` pairs, separated by `&`
- Can be cached & bookmarked
- In the browser history
- Length restricted
- Should never be used when dealing with sensitive data

POST Requests

- Additional data sent in message body
- Never cached
- Not in the browser history
- Cannot be bookmarked
- No restrictions on data length

Response Codes

When the client sends a request, the server sends a three-digit response code to inform the client of the status of the request. The first digit of the response code defines its category.

- 1xx Information
- 2xx Success
- 3xx Redirect
- 4xx Client Error
- 5xx Server Error

Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

For example, 404 is the client error for requesting a URL that was Not Found.

What is a Servlet?

A Servlet is simply a Java class which has the capability of responding to a request over any client-server protocol, like HTTP.

You would have implemented one on your own in A7.

JSON

- JavaScript Object Notation
- A simple data-interchange format for messages between client and server
- Works by associating Attribute and Value pairs

```
{
    "key": "value",
    "other_key": "value2"
}
```

JSON Types

JSON objects can have the following data types:

- Strings
- Numbers
- Arrays
- Booleans
- Null
- JSON Objects

```
{
    "string_key": "hello",
    "int_key": 2112,
    "array_key": [3,3,3],
    "boolean_key": true,
    "null_key": null,
    "nested_key": {
        "key": "value"
    }
}
```

Assignment 7

For Assignment 7 you would have needed to choose a library to use for communicating between a server and a client.
In this lab, we will show you how to use Spark, although other libraries were also typically allowed.

Demo Code

The demo code for this lab is available on the course website.

You can see it at
https://courses.cs.cornell.edu/cs2112/2020fa/labs/lab11/lab11.zip.

Starting the Server

All the main Spark functions are static functions in the class `spark.Spark`.

The first function you will want to call is `port(int port)`. This specifies the port the server will run on. For example, this code would make your server run on port 8080:

```
Spark.port(8080);
```

Route Handlers

Another important function is `get(String path, Route route)`. This function binds a handler to a specific route on your path on your server. For example, the following code creates a handler which responds to `GET` requests at `/hello` with the string `Hello World`:

```java
get("/hello", new Route() {
    @Override
    public Object handle
      (Request request, Response response)
      throws Exception {
        return "Hello World";
    }
});
```

Lambda Expressions

This is not very elegant. To fix this, Java 8 introduced lambda expressions. The following code is equivalent and much easier to read and write:

```
get("/hello", (request, response) -> "Hello World");
```

Additional Examples

More examples can be found in the repo at `src/main/java/Server.java`.

Postman

- Postman is a tool that allows you to send and receive HTTP requests through a graphical interface
- This will be invaluable for testing servlets
- You can download Postman at https://www.getpostman.com/

Starting the Demo Server

Before connecting to the server, you have to start it.

From Eclipse, run the file `Main.java` as a Java Application.

Endpoints

These are endpoints the server is set up to accept. Try testing them out in Postman!

```
GET http://localhost:8080/
GET http://localhost:8080/message
GET http://localhost:8080/json
GET http://localhost:8080/params
POST http://localhost:8080/post
POST http://localhost:8080/message
```

Demo Client

You can test out the demo client by running `Client.java`.

Inside are examples of how to send GET and POST requests in Java.

Lab Exercise

We're running our own copy of the demo server at `http://cs2112.azurewebsites.net/`.

You'll notice that a GET request to `http://cs2112.azurewebsites.net/message` displays a message. See if you can figure out how to change it.