The Internet
Chris Piech
CS106A, Stanford University
I came here to make friends and program the internet...
... And I already made friends.
For the second time ever in CS106A:
Learning Goals

1. Write a program that can make internet requests
2. Write a program that can respond to internet requests
How does your phone communicate with Facebook?
The Java program on your phone talks to the Java program at Facebook
* Android phones run Java. So do Facebook servers
Is this legit?

piech@cs.stanford.edu

piech@cs.stanford.edu is now logged in

Face Book Server

Java

Piech, CS106A, Stanford University
Send me the **full name** for piech@cs.stanford.edu

"Chris Piech"
Send me the **cover photo** for piech@cs.stanford.edu

Face Book Server

Chris Piech

Java

Piech, CS106A, Stanford University
Send the **profile photo** for piech@cs.stanford.edu
Send the status for piech@cs.stanford.edu

Face Book Server

“chillin”
Set the status for piech@cs.stanford.edu to be “lecturing”

“success”
Send me the status for piech@cs.stanford.edu

Face Book Server

"lecturing"

Set status:
Background: The Internet

The internet is just many programs sending messages (as *Strings*)

Thanks Nick for the teaching YEAH
Background: The Internet

The internet is just many programs sending messages (as Strings)

Facebook datacenter

Your computer (facebook.com)

Thanks Nick for the teaching YEAH
The internet is just many programs sending messages (as Strings)

Facebook datacenter

"Server"

Your computer (facebook.com)

"Client"

Thanks Nick for the teaching YEAH
Background: The Internet

The internet is just many programs sending messages (as *Strings*)

Thanks Nick for the teaching YEAH
The internet is just many programs sending messages (as Strings)
There are two types of internet programs. Servers and Clients
Internet 101
Servers are computers (running code)
Facebook’s closest datacenter is here
The Internet

Face Book Server

Java

Chris Piech
Status: Chris is chillin
Status: Chris is lecturing
Get status for piech@cs.stanford.edu
Face Book Server

The Internet

Chris Piech

Chris is chillin

Chris is lecturing
The Internet

Face Book Server

teaching
The Internet

Face Book Server

Java

Status: Chris is chillin'

Status: Chris is lecturing
teaching

The Internet

Face Book Server

Chris Piech
Status: Chris is chillin
Status: Chris is teaching
Many computers can connect to the same server
The Internet

Facebook datacenter

“Server”

REQUEST

RESPONSE

Tate Ole Keko’s computer (facebook.com)

Chris’ phone (facebook app)

RESPONSE

REQUEST

“Client”

Your mom’s computer (linux shell)

“Client”

“Client”

REQUEST

RESPONSE

“Client”

“Client”
Most of the Internet

Server / Clients

Aka “the backend”
Aka “the cloud”
Aka “the brains”

Aka “the frontend”
Aka “the GUI”
There are two types of internet programs. Servers and Clients
First, the server
A Server’s Simple Purpose

Request
someRequest

Server

String
serverResponse

Java
A Server’s Simple Purpose

Request
someRequest

String
serverResponse

ChatServer

Starting server on port 8080...
getMsgs
newMsg
Added new message
getMsgs
Returned 1 messages
getMsgs
Returned 1 messages
newMsg
Added new message
getMsgs
Returned 1 messages
getMsgs

Piech, CS106A, Stanford University
public String requestMade(Request request) {
    // server code goes here
}

// make a Server object
private SimpleServer server = new SimpleServer(this, 8000);

public void run(){
    // start the server
    server.start();
}
public String requestMade(Request request) {
    // server code goes here
}

// make a Server object
private SimpleServer server
    = new SimpleServer(this, 8000);

public void run()
{
    // start the server
    server.start();
}
What is a Request?

Request request

/* Request has a command */
String command;

/* Request has parameters */
HashMap<String,String> params;

// methods that the server calls on requests
request.getCommand();
request.getParam(key);  //returns associated value
Requests are like Remote Method Calls

Server has a bunch of discrete things it can do

makeToast

getChocolate
Requests are like Remote Method Calls

Server

makeToast

getchocolate
Requests are like Remote Method Calls

```java
request.getCommand();
=> "makeToast"
```

- **makeToast**
- **getChocolate**
Requests are like Remote Method Calls

To make toast, I need a parameter which is the kind of bread

makeToast
Requests are like Remote Method Calls

I was given a parameter!

makeToast
Requests are like Remote Method Calls

```java
request.getParam("bread")
makeToast
```
Requests are like Remote Method Calls

makeToast
Requests are like Remote Method Calls
Requests are like Remote Method Calls
public String requestMade(Request request) {
    String cmd = request.getCommand();
    if (cmd.equals("makeToast")) {
        Bread input = request.getParam("bread");
        Bread output = runToaster(input);
        return output.toString();
    }
    ...
}

.toString()
Requests are like Remote Method Calls
Requests are like Remote Method Calls
Requests are like Remote Method Calls
Requests are like Remote Method Calls
Requests are like Remote Method Calls
Requests are like Remote Method Calls
What is a Request?

/* Request has a command */
String command;

/* Request has parameters */
HashMap<String,String> params;

Request request

// methods that the server calls on requests
request.getCommand();
request.getParam(key); //returns associated value
public String requestMade(Request request) {
    // server code goes here
}

// make a Server object
private SimpleServer server = new SimpleServer(this, 8000);

public void run() {
    // start the server
    server.start();
}
public String requestMade(Request request) {
    // server code goes here
}

// make a Server object
private SimpleServer server = new SimpleServer(this, 8000);

public void run(){
    // start the server
    server.start();
}
What is a Port?
A Server’s Simple Purpose

1. `public String requestMade(Request request) { // server code goes here }

2. `// make a Server object
private SimpleServer server
    = new SimpleServer(this, 8000);

3. `public void run(){ // start the server
    server.start();
}
public String requestMade(Request request) {
    // server code goes here
}

// make a Server object
private SimpleServer server = new SimpleServer(this, 8000);

public void run(){
    // start the server
    server.start();
}
Echo Server

Your command was 11 chars long.
Echo Server

Request
Any Request

String
Length of the cmd

Starting server...
Request received hello
Request received this+is+a+test
Request received what'sGood
Request received ping
Request received ping
Request received ping
Request received ping
Request received pong
Request received ping
There are two types of internet programs. Servers and Clients
Then, the client
A Clients’s Purpose

1. Interact with the user
2. Get data from its server
3. Save data to its server
try {

    // 1. construct a new request
    Request example = new Request("getStatus");

    // 2. add parameters to the request
    example.addParam("name", "chris");

    // 3. send the request to a computer on the internet
    String result = SimpleClient.makeRequest(HOST, example);

} catch (IOException e) {

    // The internet is a fast and wild world my friend

}
try {

  // 1. construct a new request
  Request example = new Request("getStatus");

  // 2. add parameters to the request
  example.addParam("name", "chris");

  // 3. send the request to a computer on the internet
  String result = SimpleClient.makeRequest(HOST, example);

} catch(IOException e) {

  // The internet is a fast and wild world my friend

}
try {

// 1. construct a new request
Request example = new Request("getStatus");

// 2. add parameters to the request
example.addParam("name", "chris");

// 3. send the request to a computer on the internet
String result = SimpleClient.makeRequest(HOST, example);

} catch (IOException e) {

// The internet is a fast and wild world my friend

}
try {

    // 1. construct a new request
    Request example = new Request("getStatus");

    // 2. add parameters to the request
    example.addParam("name", "chris");

    // 3. send the request to a computer on the internet
    String result = SimpleClient.makeRequest(HOST, example);

} catch(IOException e) {

    // The internet is a fast and wild world my friend

}
try {

    // 1. construct a new request
    Request example = new Request("getStatus");

    // 2. add parameters to the request
    example.addParam("name", "chris");

    // 3. send the request to a computer on the internet
    String result = SimpleClient.makeRequest(HOST, example);

} catch(IOException e) {

    // The internet is a fast and wild world my friend

}
try {

    // 1. construct a new request
    Request example = new Request("getStatus");

    // 2. add parameters to the request
    example.addParam("name", "chris");

    // 3. send the request to a computer on the internet
    String result = SimpleClient.makeRequest(HOST, example);

} catch(IOException e) {

    // The internet is a fast and wild world my friend

}
try {

    // 1. construct a new request
    Request example = new Request("getStatus");

    // 2. add parameters to the request
    example.addParam("name", "chris");

    // 3. send the request to a computer on the internet
    String result = SimpleClient.makeRequest(HOST, example);

} catch(IOException e) {

    // The internet is a fast and wild world my friend

}
try {

    // 1. construct a new request
    Request example = new Request("getStatus");

    // 2. add parameters to the request
    example.addParam("name", "chris");

    // 3. send the request to a computer on the internet
    String result = SimpleClient.makeRequest(HOST, example);

} catch(InterruptedException e) {

    // The internet is a fast and wild world my friend

}
Time for a little chat
Chat Server and Client

> CJP: Testing
> LMS: Hellooooo
> LMS: I'm online!
> LMS: This is great
> LMS: And this is going to make it into lecture
addMsg

msg = Hello world

history = []
history = 
  [Hello world]

getMsgs
index = 0
history = [
    Hello world
]

[Hello world]
history = [
    Hello world
]

addMsg
msg = Im here too

Chat Client
> Hello world

Chat Client
> Hello world
Im here too
Send
history = [Hello world, Im here too]

getMsgs

index = 1

Chat Client

> Hello world

Chat Client

> Hello world
Chat Client

> Hello world
> Im here too

Chat Client

> Hello world
> Im here too

history = [Hello world, Im here too]
Chat Server

```
addMsg
msg = text
```

```
getMsgs
index = startIndex
```