# WebSockets

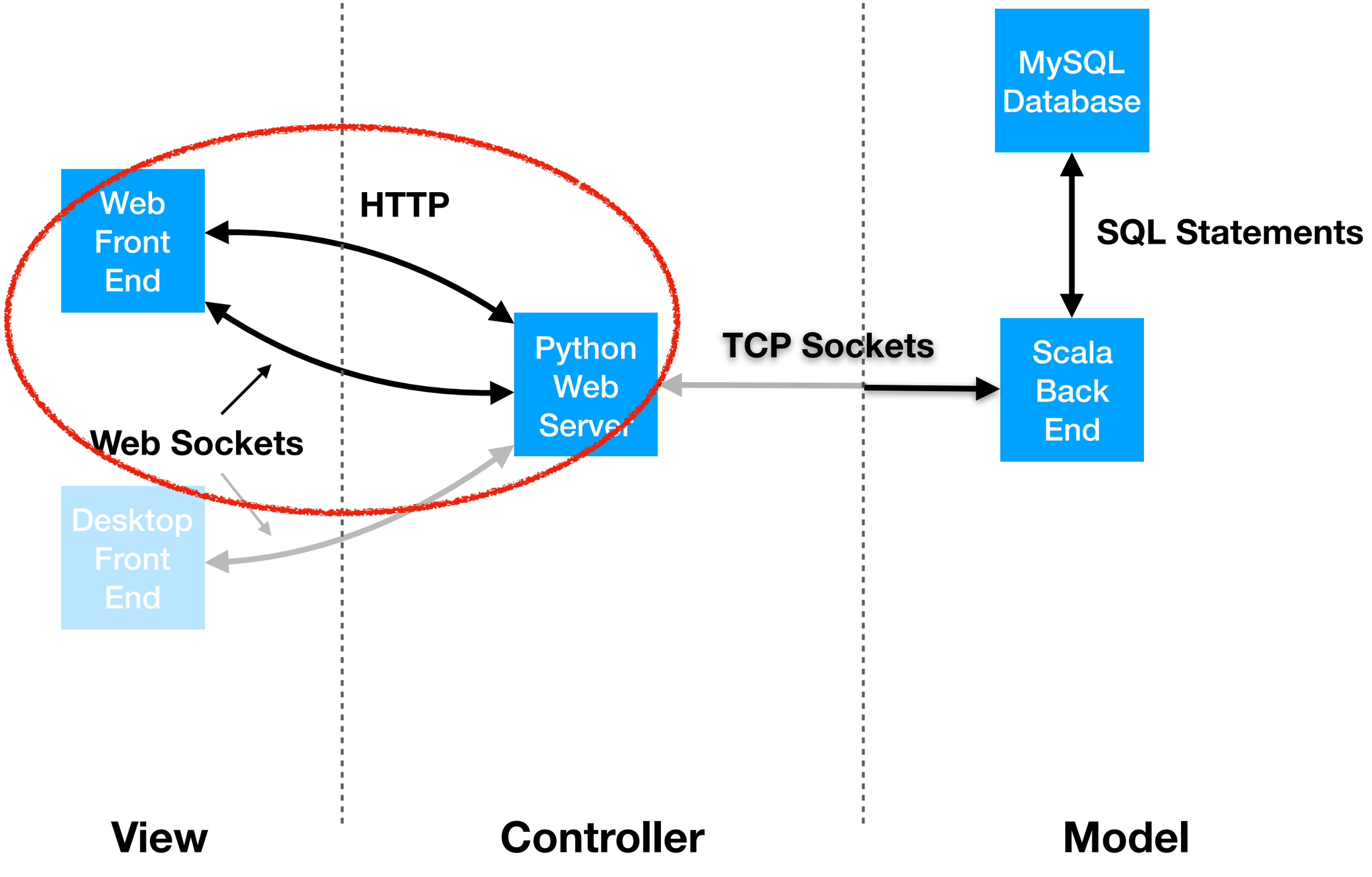
# Lecture Question

## Task: Get started on Clicker 2

- Submit anything for the points

* This question will be open until midnight

# CSE116 - End Game



# Flask

- A python web framework
  - Similar syntax to bottle
  - Offers simpler web socket support
  - Example assumes there is a file "index.html" in directory "static"

```
from flask import Flask, send_from_directory

app = Flask(__name__)

@app.route('/')
def index():
    return send_from_directory('static', 'index.html')

app.run(port=8080)
```

# Flask

- Handle POST requests using methods=["POST"]
- Get form POST data with request.form

```
<form action="/game" method="post">
    <label>
        Enter your username:
        <input type="text" name="username"/>
    </label>
    <input type="submit"/>
</form>
```

```
@app.route('/game', methods=["POST"])
def game():
    username = request.form.get('username')
    # do something with username
    return send_from_directory('static', 'index.html')
```

# Flask

- Many more features
- But let's focus on web sockets
- Check out the documentation if you need more features for your project

# Flask - Web Sockets

- Use flask_socketio library
- Install with pip, or..
- Create a requirements.txt file
  - Add all dependancies to this file
  - IntelliJ will see the file and ask to install dependancies
  - Similar to pom.xml
- Once installed, start a socket server using the library

**requirements.txt**

```
flask
flask-socketio
eventlet
```

```python
from flask_socketio import SocketIO

app = Flask(__name__)
socket_server = SocketIO(app)

socket_server.run(app, port=8080)
```

# Flask - Web Sockets

- Send/receive messages with types over web sockets
  - Similar to Scala actor messages
- Annotate methods to run them when certain message types are received

```
app = Flask(__name__)
socket_server = SocketIO(app)

@socket_server.on('clickGold')
def click_gold():
    pass
    # received a "clickGold" message over the socket


@socket_server.on('buy')
def buy_equipment(equipmentID):
    pass
    # received a "buy" message over the socket with a parameter equipmentID

socket_server.run(app, port=8080)
```

# Flask - Web Sockets

- Send message using emit
- Specify the message type and any parameters needed

```
app = Flask(__name__)
socket_server = SocketIO(app)

@socket_server.on('clickGold')
def click_gold():
    # received a "clickGold" message over the socket
    emit('goldClicked', 'I see you clicking that gold')


socket_server.run(app, port=8080)
```

# Flask - Web Sockets

- To send a message to a client later
  - Remember their socket id (sid)
- Access request.sid when a user connects
  - Associate this sid with their username (or other identifier)
- When you want to send this user a message
  - Get their sid and send a message to their "room"

```
usernameToSid = {}

@socket_server.on('register')
def got_message(username):
    usernameToSid[username] = request.sid


# Later

user_socket = usernameToSid.get(username)
socket_server.emit('message', data, room=user_socket)
```

# Browser - Web Sockets

- Connect to the web server via sockets
- Import the library in the head of your HTML

```
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.2.0/socket.io.js"></script>
```

- Connect, send, receive in JavaScript
  - Use callback methods when a message is received
  - Use emit to send a message with a type

```
var socket = io.connect({transports: ['websocket']});

socket.on('connect', function (event) {
    // connected to server
});

socket.on('message', function (event) {
    // received a message from the server
    console.log(event);
    socket.emit("ack", "thanks for the message!");
});

socket.emit("clickGold");
socket.emit("buy", "shovel");
```

# Lecture Question

## Task: Get started on Clicker 2

- Submit anything for the points

* This question will be open until midnight