MySQL

Lecture Question

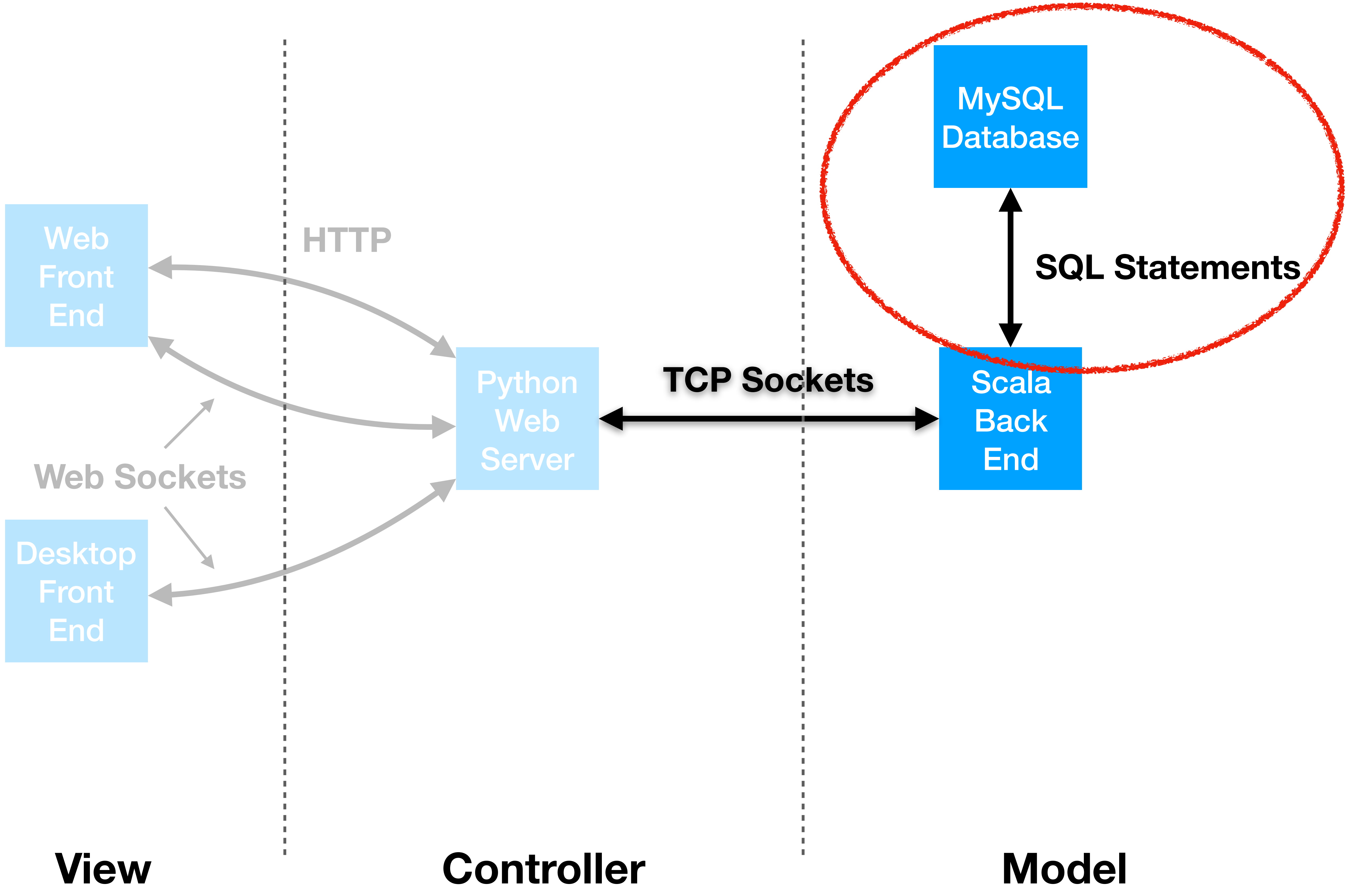
Task: Create a custom class and store objects of your type in a database

- Install and run MySQL
- Connect to MySQL in Scala using the JDBC Driver
- Write methods to store and update objects of your type in the database
- Write methods to retrieve your objects from the database

- Good News: I haven't figured out how to grade MySQL in AutoLab yet
    - Free points. Please practice so you're ready for Clicker 2 (assuming I get good at MySQL grading for the HW)

* This question will be open until midnight

CSE116 - End Game



MySQL v. SQLite

- MySQL
  - Database server
  - Runs as a separate process that could be running on a different machine
  - Connect to the server and send it SQL statements to execute
- SQLite
  - Removes networking
  - Must run on the same machine as the app
  - Can be used for small apps
    - Common in embedded system - Including Android/iOS apps

MySQL

- A program that must be downloaded, installed, and ran
- Is a server
  - By default, listens on port 3306
- Connect using JDBC (Java DataBase Connectivity)
  - Must download the MySQL Driver for JDBC (Use Maven. Artifact in repo)
  - JDBC abstracts out the networking so we can focus on the SQL statements

MySQL

- After MySQL is running and the JDBC Driver is downloaded..
- Connect to MySQL Server by providing
  - url of database
  - username/password for the database
    - Whatever you chose when setting up the database

```
val url = "jdbc:mysql://localhost/mysql?serverTimezone=UTC"
val username = "root"
val password = "12345678"

var connection: Connection = DriverManager.getConnection(url, username, password)
```

MySQL - Security

- For real apps that you deploy
    - Do not check your password into version control!
        - A plain text password in public GitHub repo is bad
        - Attacker can replace localhost with the IP for your app and can access all your data
    - Common to save the password in a environment variable to prevent accidentally pushing it to git
    - Do not use the default password for any servers you're running
        - This is what caused the Equifax leak (Not with MySQL)
- Attacker have bots that scan random IPs for such vulnerabilities

```
val url = "jdbc:mysql://localhost/mysql?serverTimezone=UTC"
val username = "root"
val password = "12345678"

var connection: Connection = DriverManager.getConnection(url, username, password)
```

MySQL

- Once connected we can send SQL statements to the server

```
val statement = connection.createStatement()
statement.execute("CREATE TABLE IF NOT EXISTS players (username TEXT, points INT)")
```

- If using inputs from the user always use prepared statements
  - Indices start at 1 😢

```
val statement = connection.prepareStatement("INSERT INTO players VALUE (?, ?)")

statement.setString(1, "mario")
statement.setInt(2, 10)

statement.execute()
```

MySQL - Security

- Not using prepared statements?
  - Vulnerable to SQL injection attacks
- If you concatenate user inputs directly into your SQL statements
  - Attacker chooses a username of "';DROP TABLE players;"
  - You lose all your data
  - Even worse, they find a way to access the entire database and steal other users' data
  - SQL Injection is the most common successful attack

MySQL

- Use executeQuery when pulling data from the database
- Returns a ResultSet
  - The next() methods queue the next result of the query
  - next returns false if there are no more results to read
- Can read values by index of by column name
  - Use get methods to convert SQL types to Scala types

```scala
val statement = connection.createStatement()
val result: ResultSet = statement.executeQuery("SELECT * FROM players")

var allScores: Map[String, Int] = Map()

while (result.next()) {
  val username = result.getString("username")
  val score = result.getInt("points")
  allScores = allScores + (username -> score)
}
```

SQL

- SQL is based on tables with rows and column
  - Similar in structure to CSV except the values have types other than string
- How do we store an array or key-value store?
  - With CSV our answer was to move on to JSON
  - SQL answer is to create a separate table and use JOINs (Or move to MongoDB)
  - This is beyond CSE116 so we'll stick to data that fits the row/column structure

Lecture Question

Task: Create a custom class and store objects of your type in a database

- Install and run MySQL
- Connect to MySQL in Scala using the JDBC Driver
- Write methods to store and update objects of your type in the database
- Write methods to retrieve your objects from the database

- Good News: I haven't figured out how to grade MySQL in AutoLab yet
  - Free points. Please practice so you're ready for Clicker 2 (assuming I get good at MySQL grading for the HW)

* This question will be open until midnight