MySQL,
User Authentication

http://goo.gl/Vu1RVa

code in 5H folder

This mini-project

- **Remember:**
  - We only have about one week left together
  - For most of you (if not all of you) this is your first web application
  - Web development, like most coding, can be messy
    - lots of components
    - lots of technology options available

The purposes of this mini-project

- Learn how to approach designing and building a web application
- Give enough breadth so that you can pursue web dev in depth in the future
  - Get exposure to all parts of web application implementation, and available technology options
- Practice debugging, reading documentation and code

- *Gain intuition*

Web App Design

- **Problem**: People don’t have a good way to keep track of (and discover) favorite quotes.
- **Solution**: Web app allows users to post their favorite quotes. Web app also allows users to save other people’s favorite quotes.
- **Purpose**: Give people a place to store (and discover) favorite quotes.

Data Models

- User
  - id, username, password
- Quotes
  - id, content, source, created_by

- SavedBy
  - id, quote_id, savedby_id

Today we will

- Create the users table in MySQL
- Query table from web application
- Create signup, login forms
- signup route/controller
- login route/controller
- logout route/controller

User Data Model

- In MySQL
  - Create new database ‘myapp’
  - Create ‘users’ table in database
- ‘users’ table
  - id
  - username
  - password
- "CREATE TABLE users (id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY, username VARCHAR(30) NOT NULL, password VARCHAR(1000) NOT NULL)"
- "INSERT INTO users (username, password) VALUES ('testusername', 'testpassword')"

Template Code

- npm install
- views/index.ejs
- node index.js

Node-MySQL

- Node module that allows us to connect to a MySQL database https://github.com/felixge/node-mysql
- Remember to start your MySQL server before you start your Node server

User Creation, Authentication

- Simple:
  - password → hash, salt
- Password hashing and salting
  - http://security.stackexchange.com/questions/51959/why-are-salted-hashes-more-secure

- https://www.npmjs.com/package/bcrypt

myapp/controllers/UsersController.js

- UsersController.signup(req, res) **TODO**
  - you need 2 SQL queries, one nested in the other's callback function
    - SQL queries with node: https://github.com/felixge/node-mysql
  - use bcrypt to hash password: https://www.npmjs.com/package/bcrypt

Usage

async (recommended)

To hash a password:

```
var bcrypt = require('bcrypt');
bcrypt.genSalt(10, function(err, salt) {
    bcrypt.hash('B4c0/\/', salt, function(err, hash) {
        // Store hash in your password DB.
    });
});
```

User Sessions

- Recall that HTTP is stateless
- Store on server-side something about the user/web state
- http://stackoverflow.com/questions/3804209/what-are-sessions-how-do-they-work
- http://machinesaredigging.com/2013/10/29/how-does-a-web-session-work/

- http://passportjs.org/guide/username-password/

myapp/index.js

- configure passport auth strategy **TODO**
  - I have copied+pasted example configuration code from http://passportjs.org/guide/username-password/
  - We need to make the code work for us.
    - you need one SQL query.
    - use bcrypt to compare passwords: https://www.npmjs.com/package/bcrypt

To check a password:

```
// Load hash from your password DB.
bcrypt.compare('B4c0/\/', hash, function(err, res) {
    // res == true
});
bcrypt.compare('not_bacon', hash, function(err, res) {
    // res == false
});
```

login, logout

- login done
  - passport: http://passportjs.org/guide/authenticate/
- logout done
  - passport: http://passportjs.org/guide/logout/

Intro to Template rendering

1) Server-side
    - Controller/route retrieves data, renders view with template
2) Client-side
    - Ajax
    - Controller/route retrieves and returns data as JSON
    - Client-side JavaScript updates HTML