CoGrammar

Workshop 13 - Session Management

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: **Feedback on Workshops**

Skills Bootcamp Certification Overview

✅ **Criteria 4: Original Deadline for Demonstrating Employability without Co-certification**

Record a job outcome by 23 September 2024

✅ **Criterion 4: Updated Deadline for Imperial College London & University of Manchester Co-certification**

Record a job offer by 15 July 2024

CoGrammar

Lecture Objectives

- Understand what session management is
- Understand the different techniques for securing data over the internet
- Clearly differentiate between authentication and authorisation
- Understand how session management can be incorporated into a web application

Which of the following options would you pick to handle a user session

A. Keep track of the active users on the server side and send the user ID with every request to determine what they can do

B. Using a token that stores user details, the token will verified with every request

C. Send the username/email and password with every request and authenticate the user with every request

Client Side Rendering:
Recap

CoGrammar

Client-Side Rendering

What is it

- When JavaScript on the users end generates the HTML content
- Makes navigating the page application faster
- Reduces the load on the server

Client Side Rendering

Problems

- Can be slow on initial loads since all of the scripts are copied
- We can't have any sensitive information since all of the code will be sent to the client
  - Database connection strings
  - Private API keys
  - Private HTTP endpoints

Client Side Rendering

Solution

- Connecting to a backend system
  - We can perform “secret” operations without end-users having access to the code
  - We can store application secrets securely
  - We can connect to private services without the client knowing about these connections

Client-Server Communication

CoGrammar

Client-Server Communication

What is it

- Typically performed using the stateless HTTP communication
- When the user needs data the client application makes a request to the server for the data
- The server will respond to the clients request
- The server can only ever respond to a request

Client-Server Communication

What is statelessness

- The server doesn't "remember" what the user did in the past
  - All of the information required to complete a task should be included in the request
- There should be no sequence of requests
  - The user shouldn't have to make requests in a specific order to get a final output
- The same request should always return the same response

Client-Server Communication

What is statelessness

- The server doesn't "remember" what the user did in the past
  - All of the information required to complete a task should be included in the request
- There should be no sequence of requests
  - The user shouldn't have to make requests in a specific order to get a final output
- The same request should always return the same response

Access Control

CoGrammar

Access Control

If the server doesn't "remember" past communication?

- How do we know who's making requests
- How do we control access to specific resources

Access Control

Authentication VS Authorization

- **Authentication (Who are you) -** Verifying that someone should be able to use the system
  - Username + Password
  - Access Token / API Key
  - Biometrics
  - etc
- **Authorization (What can you do)** - Determines what an authenticated user can do in the system
  - Update certain records
  - View specific pages
  - Delete content
  - etc

Access Control

Importance of Authentication and Authorization?

- You need to know who is using your application
  - If you application has sensitive information, you only want verifiable people to use the application
  - If the application has custom features per user, you need to be able to identify these users
- Not all users are created equally
  - Access should be limited to certain parts of the application
  - Users should only be able to access the things they need

Session Management

CoGrammar

Session Management

We can set rules on the user, but how does the server know who the user is?

- Since HTTP is stateless
  - The client needs to send their details with each request
  - Details sent should provide the server with verifiable information that state that a user should be able to access the feature they requested

Session Management

Session Tokens?

- After the user logs in, a token can be created, it will:
  - Store key information about a user (user id, role, username)
  - Be valid for a finite amount of time
  - Store the users data as a single encrypted value
- The token is sent to the client
  - The client stores the token
  - The client will add the token to each request that they make

Session Management

Session Token: Client Server Communication

- Client
  - Gets the token and stores it
    - Local Storage
    - Session Storage
    - Cookies
  - Sends the token with every request that requires it
- Server
  - Looks for the token in authentication operations
  - Verifies the token
    - Valid encryption
    - Hasn’t expired

Securing the Token

CoGrammar

Securing Session Tokens

Problems

- Tokens need to be sent from on machine to another machine
- Anyone with a valid token can send requests to the server
- How do we ensure that tokens are safe on the client side

Securing Session Tokens

Sending the Token Securely

- HTTPS (TLS)
  - Encrypts the data that is being sent, this protects data against eavesdropping
  - Never switch between HTTP and HTTPS
  - Implement HSTS to enforce HTTPs connections
- Use Cookies
  - **Secure** - cookies only send information where theres an HTTPS connection
  - **HttpOnly** - Protects cookies from being used on different sessions
  - OWASP

Implementing Session Management

CoGrammar

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Thank you for joining!