HyperionDev

Full-Stack Web Development

HyperionDev

Working with NoSQL Databases, MongoDB, and Mongoose

Full Stack Web Development 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**

Full Stack Web Development 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 Lectures**

Objective
s

- ❖ Understand the key concepts of NoSQL databases.
- ❖ Set up and connect to a MongoDB database.
- ❖ Use Mongoose to perform basic CRUD operations on MongoDB.

Introduction to NoSQL and MongoDB

- **Problem Statement:** Traditional databases can struggle to keep up with the needs of modern web applications. We need a better way to handle data.
- **Overview of NoSQL Databases:**
  - NoSQL databases are flexible and can store different types of data without needing a fixed structure.
  - They are designed to scale easily, meaning they can grow as our data grows.
- **Introduction to MongoDB:**
  - MongoDB is a popular NoSQL database that works well with large amounts of unstructured data (like user-generated content).
- **Real-World Example:**
  - Companies like Uber and eBay use MongoDB to manage their data because it can easily adapt to their changing needs.

Introduction to NoSQL Databases

- **Differences Between NoSQL and SQL Databases:**
  - SQL databases use structured tables and require predefined schemas. NoSQL databases are more flexible and don't require a set structure.
- **Types of NoSQL Databases:**
  - **Document Databases:** Store data in documents (like JSON). Example: MongoDB.
  - **Key-Value Stores:** Use simple key-value pairs for data storage. Example: Redis.
  - **Column Stores:** Organize data into columns instead of rows. Example: Cassandra.
  - **Graph Databases:** Focus on relationships between data points. Example: Neo4j.
- **Focus on Document Databases:**
  - We will mainly look at Document databases, especially MongoDB.

Setting Up and Connecting to MongoDB

- **Guide to Installing MongoDB:**
  - You can either install MongoDB on your computer or use MongoDB Atlas, which is a cloud service that provides MongoDB.
- **How to Start the MongoDB Server:**
  - Once installed, you can start the MongoDB server using the command line.
- **Connecting Using the MongoDB Shell:**
  - After starting the server, you can connect to it through a special program called the MongoDB shell.
- **Basic Commands:**
  - Create a database: `use myDatabase`
  - Insert a document: `db.myCollection.insert({name: "John", age: 30})`
  - Retrieve data: `db.myCollection.find()`

```javascript
const mongoose = require('mongoose');


mongoose.connect('your-atlas-connection-string', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => {
  console.log('Connected to MongoDB Atlas');
}).catch((error) => {
  console.error('Error connecting to MongoDB:', error);
});
```

Using Mongoose with MongoDB

- **Introduction to Mongoose:**
  - Mongoose is a tool that helps us work with MongoDB in a simpler way. It acts like a bridge between our Node.js application and MongoDB.
- **Advantages of Using Mongoose:**
  - Mongoose lets us define a structure for our data (schemas) and makes it easier to interact with MongoDB.
- **Demonstration of Defining Schemas:**
  - A schema is like a blueprint for our data. For example, if we want to store information about users, we can define a User schema.
- **Basic CRUD Operations with Mongoose:**
  - Create: Adding a new user to the database.
  - Read: Finding users in the database.
  - Update: Changing information about an existing user.
  - Delete: Removing a user from the database.

Using Postman to Test RESTful APIs

- **What is Postman?**
  - Postman is a powerful tool for API development and testing.
  - It allows users to send requests to APIs and view responses in a user-friendly interface.
- **Key Features:**
  - **User Interface**: Intuitive interface for constructing requests and viewing responses.
  - **HTTP Methods**: Supports all HTTP methods (GET, POST, PUT, DELETE, etc.).
  - **Environment Variables**: Allows storing variables for reuse in different requests.
  - **Collections**: Organize requests into groups for better management.

Using Postman to Test RESTful APIs

Steps to Use Postman:

1. **Install Postman**: Download and install Postman from the official website.
2. **Create a New Request**:
    - Click on "New" and select "HTTP Request".
3. **Select the HTTP Method**:
    - Choose the method you want to use (GET, POST, etc.) from the dropdown.
4. **Enter the Request URL**:
    - Input the URL of the API endpoint you want to test.
5. **Add Request Body** (for POST/PUT requests):
    - Go to the "Body" tab and select "raw" or "form-data".
    - Enter the required data in JSON format if using "raw".
6. **Send the Request**:
    - Click the "Send" button to send the request to the server.
7. **View the Response**:
    - Check the "Response" section for status code, response time, and data returned from the server.

Using Postman to Test RESTful APIs

- **Tips for Using Postman:**
  - Use **Collections** to group related API requests for easy access.
  - Utilize **Environment Variables** for dynamic data (e.g., tokens, URLs).
  - Explore **Test Scripts** to automate testing of responses.

Hyperiondev

Questions and Answers

HyperionDev

Thank You for attending!