Praktisches Labor - Express-Server (50 Min)

Ziel des Übung:

- Erstellen Sie einen Express-Server und führen Sie ihn aus
- Arbeiten Sie mit Middleware im Express-Server
- Verwenden Sie Middleware und JWT für die Authentifizierung
- Rendern Sie eine statische HTML-Seite über den Express-Server

Einrichtung : Klonen Sie die Labor-Dateien

1. Öffnen Sie ein Terminalfenster über das Menü im Editor: Terminal > Neues Terminal.

2. Wechseln Sie in Ihren Projektordner.

   ```
   cd /home/project
   ```

3. Überprüfen Sie, ob Sie den Ordner **lkpho-Cloud-applications-with-Node.js-and-React** haben.

   ```
   ls /home/project
   ```

Wenn du das tust, kannst du zu Schritt 5 überspringen.

4. Klone das Git-Repository, das die für dieses Labor benötigten Artefakte enthält, falls es noch nicht existiert.

   ```
   git clone https://github.com/ibm-developer-skills-network/lkpho-Cloud-applications-with-Node.js-and-React.git
   ```

5. Wechseln Sie in das Verzeichnis für dieses Labor.

   ```
   cd lkpho-Cloud-applications-with-Node.js-and-React/CD220Labs/expressjs/
   ```

6. Listen Sie den Inhalt dieses Verzeichnisses auf, um die Artefakte für dieses Labor zu sehen.

   ```
   ls
   ```

Sie müssen einige Übungsdateien haben, die Sie in den Übungen verwenden werden.

Code anzeigen, Server ausführen und über curl/Browser mit dem Server verbinden

1. Im Datei-Explorer sehen Sie expressServer.js. Es würde so aussehen.

▶ Sie können auch hier klicken, um den Code anzuzeigen

Dies ist ein einfacher Express-Server, der auf Port 3333 hört, mit 4 Endpunkten.

```
/
/loginDetails
/login/:name - POST
/:name
```

2. Führen Sie im Terminalfenster den folgenden Befehl aus, um sicherzustellen, dass das Express-Paket installiert ist.

```
npm install --save express
```

3. Führen Sie im Terminalfenster den Server mit dem folgenden Befehl aus.

```
node expressServer.js
```

Sie sollten eine ähnliche Ausgabe sehen.

```
Listening at http://localhost:3333
```

4. Klicken Sie auf „Terminal teilen“, um das Terminal zu teilen, wie im Bild unten dargestellt.

5. Verwenden Sie im zweiten Terminalfenster den `curl`-Befehl, um die Anwendung anzupingen.

```
curl localhost:3333
```

Sie sollten eine Ausgabe sehen, die ähnlich aussieht.

```
Welcome to the express server
```

Das zeigt an, dass Ihre App läuft.

6. Probieren Sie die anderen Endpunkte mit den curl-Befehlen im selben Terminal aus.

/login/:name

```
curl -X POST http://localhost:3333/login/Jason
```

Sie sollten eine ähnliche Ausgabe sehen.

```
Jason, You are logged in!
```

/:name

```
curl http://localhost:3333/Jason
```

Sie sollten eine Ausgabe sehen, die ähnlich aussieht wie diese.

```
Hello Jason
```

/Anmeldedaten

```
curl http://localhost:3333/loginDetails
```

Sie sollten eine Ausgabe sehen, die ähnlich aussieht wie diese.

```
[{"name":"Jason","login_time":"2020-11-20T06:06:56.047Z"}]
```

7. Um den Server zu stoppen, gehen Sie zum Hauptbefehlsfenster und drücken Sie Ctrl+c, um den Server zu stoppen.

Aufgabe 1 : Fügen Sie Ihren eigenen Endpunkt hinzu

*Hinweis - Dies wird nicht bewertet

Erstellen Sie eine Liste mit den Namen der Monate. Fügen Sie einen Endpunkt im Code `/fetchMonth/:num` hinzu, der einen bestimmten Monat aus der Liste abruft und an den Benutzer zurückgibt. Wenn die Nummer ungültig ist, sollte eine entsprechende Fehlermeldung zurückgegeben werden.

▼ Klicken Sie hier, wenn Sie Hilfe bei der Aufgabe benötigen

```
# Define an array containing the names of the months
const months = ["January","February","March","April","May","June","July","August","September","October","November","December"];
# Define a route to fetch the month name based on a given number
app.get("/fetchMonth/:num", (req, res) => {
    # Parse the number from the request parameters
    let num = parseInt(req.params.num);
    # Check if the number is a valid month number
    if(num < 1 || num > 12) {
        # Send an error message if the number is not valid
        res.send("Not a valid month number");
    } else {
        # Send the corresponding month name if the number is valid
        res.send(months[num - 1]);
    }
});
```

Middleware verwenden

1. Im Datei-Explorer die Datei expressAppLevelMiddleware.js anzeigen

▶ Hier klicken, um den Code anzuzeigen

Dieser Server verwendet Middleware zur Authentifizierung. Wenn das Passwort nicht pwd123 ist, wird der Benutzer nicht zum Einloggen zugelassen. Dieser Server hat nur einen Endpunkt und nimmt das Passwort als Abfrageparameter entgegen.

2. Starte den Server.

```
node expressAppLevelMiddleware.js
```

Sie sollten eine Ausgabe sehen, die sagt `Listening at http://localhost:3333`

3. Verwenden Sie im zweiten Terminalfenster den `curl`-Befehl, um die Anwendung anzupingen.

   ```
   curl localhost:3333/home
   ```

Sie sollten eine Ausgabe sehen, die sagt `This user cannot login`.

4. Führen Sie den curl-Befehl aus und übergeben Sie den Passwortparameter.

   ```
   curl http://localhost:3333/home?password=pwd123
   ```

Sie sollten eine Ausgabe sehen, die `Hello World!` sagt.

Das liegt daran, dass der Server eine Middleware hat, die jede Anfrage an den Server filtert, um zu überprüfen, was das Passwort ist, und nur dann fortfährt, wenn das Passwort `pwd123` ist.

5. Um den Server zu stoppen, gehen Sie zum Hauptbefehlsfenster und drücken Sie Ctrl+c, um den Server zu stoppen.

Express-Server mit Authentifizierung

In dieser Übung lernen Sie, wie Sie eine Authentifizierungsschicht in Ihren Express-Server integrieren, um den Server sicher zu machen. Sie werden das Tool Postman für dieses Labor verwenden.

1. Im Datei-Explorer den Code expressWithAuthentication.js anzeigen

▶ Hier klicken, um den Code anzuzeigen

2. Um diese Anwendung auszuführen, wie Sie vielleicht bemerkt haben, verwenden wir zwei neue Pakete, die Sie zuvor nicht verwendet haben. Führen Sie den folgenden Befehl aus, um `jsonwebtoken` und `express-session` zu installieren.

   ```
   npm install --save express-session jsonwebtoken
   ```

3. In diesem Code haben Sie einen Endpunkt, **/auth/get_message**, der nur für authentifizierte Benutzer erlaubt ist. Starten Sie den Server und versuchen Sie zunächst, auf den Endpunkt zuzugreifen. Es sollte ein Fehler angezeigt werden.

   ```
   node expressWithAuthentication.js
   ```

Sie sollten eine Ausgabe sehen, die sagt `Listening at http://localhost:5000`.

4. Verwenden Sie im zweiten Terminalfenster den folgenden `curl`-Befehl.

```
curl localhost:5000/auth/get_message
```

Sie sollten eine Ausgabe sehen, die sagt `{"message":"User not logged in"}`.

5. Sie müssen einen Benutzer mit einem Benutzernamen und Passwort registrieren und sich mit diesem Benutzernamen und Passwort anmelden, um auf den Endpunkt zugreifen zu können. Klicken Sie auf das **Skills Network Toolbox**-Symbol, wählen Sie **Others** und klicken Sie auf **Launch Application**. Geben Sie die Portnummer `5000` ein und öffnen Sie die URL. Sie wird in einem neuen Browserfenster geöffnet. Kopieren Sie die URL. Gehen Sie zu https://www.postman.com/. Möglicherweise müssen Sie sich anmelden, wenn Sie Postman zum ersten Mal verwenden.

6. Geben Sie in Postman die kopierte URL ein und fügen Sie **/register** an.

7. Wählen Sie 'Body' >> 'raw' >> 'JSON' und übergeben Sie die Parameter.

```
{"username":"user2", "password":"password2"}
```

Hinweis: "user2" & "password2" dienen nur zur Veranschaulichung. Sie können jeden Benutzernamen & jedes Passwort verwenden.

8. Setzen Sie den Abfragetyp auf **POST** und klicken Sie auf **senden**. Sie werden eine Bestätigung sehen, dass der Benutzer registriert wurde.

9. Verwenden Sie jetzt die gleiche kopierte URL, um sich mit dem Suffix **/login** anzumelden. Die zu übergebenden Parameter bleiben gleich. Dies ist ebenfalls eine **POST**-Anfrage. Klicken Sie auf **senden**. Sie werden eine Nachricht sehen, die bestätigt, dass Sie angemeldet sind, wie unten zu sehen ist.

10. Jetzt können Sie auf den Endpunkt **/auth/get_message** zugreifen, und es wird die Nachricht zurückgegeben.

Routing

Wie der Name schon sagt, können Sie die API-Anfragen an verschiedene Handler weiterleiten. Üblicherweise sind die Handler logisch basierend auf den Objekten, mit denen sie arbeiten, unterteilt.

1. Im Datei-Explorer sehen Sie den Code expressRouting.js

▶ Hier können Sie klicken, um den Code anzuzeigen

Dieser Server verzweigt die Anfragen basierend auf den Endpunkten und verwendet Router, um sie zu verarbeiten. Alle /user-Endpunkte werden vom userRouter und die /item-Endpunkte vom itemRouter verarbeitet.

```
/user/:id
/item/:id
```

```
node expressRouting.js
```

Sie sollten eine Ausgabe sehen, die sagt `Listening at http://localhost:3333`.

2. Verwenden Sie im zweiten Terminalfenster den folgenden `curl`-Befehl.

```
curl localhost:3333/item/1
```

Sie sollten eine Ausgabe sehen, die sagt `Item 1 last enquiry Fri Nov 20 2020 15:17:46 GMT+0530 (India Standard Time)`.

3. Öffnen Sie im zweiten Terminalfenster den folgenden `curl`-Befehl.

```
curl localhost:3333/user/1
```

Sie sollten eine Ausgabe sehen, die sagt `User 1 last successful login Fri Nov 20 2020 15:19:52 GMT+0530 (India Standard Time)`.

4. Um den Server zu stoppen, gehen Sie zum Hauptbefehlsfenster und drücken Sie Ctrl+c, um den Server zu stoppen.

Statische Seiten rendern

1. Im Dateiexplorer die Datei expressStaticPages.js anzeigen

▶ Hier können Sie auf den Code in expressStaticPages.js klicken

Dieser Server hat, wie Sie sehen, keine Endpunkte. Aber er hat ein Middleware, das das Verzeichnis für statische Dateien festlegt. Daher ist jede Datei im Verzeichnis `cad220_staticfiles` zugänglich. Der Ordner enthält die HTML-Seite, die gerendert werden soll.

2. Führen Sie den Server mit dem folgenden Befehl aus.

```
node expressStaticPages.js
```

Sie sollten eine Ausgabe sehen, die sagt `Listening at http://localhost:3333`.

3. Klicken Sie auf die Schaltfläche **Skills Network** auf der linken Seite, um die “Skills Network Toolbox” zu öffnen. Klicken Sie dann auf **Other** und dann auf **Launch Application**. Von dort aus sollten Sie in der Lage sein, den Port 3333 einzugeben und zu starten.

4. Fügen Sie `/ReactCalc.html` zur URL in der Adressleiste hinzu.

Sie werden die Seite wie unten dargestellt sehen.

Aufgabe: Fügen Sie Ihre eigene statische Datei hinzu

*Hinweis - Dies ist nicht benotet

Fügen Sie eine statische Datei, ein Bild oder eine HTML-Datei, zum Verzeichnis `cad220_staticfiles` hinzu und versuchen Sie, darauf über `/<dateiname>` im Browser zuzugreifen, der durch Klicken auf die Schaltfläche **Skills Network** auf der linken Seite geöffnet wurde, um die "Skills Network Toolbox" zu öffnen. Klicken Sie dann auf **Other** und dann auf **Launch Application**. Von dort aus sollten Sie in der Lage sein, den Port einzugeben und zu starten.

Erstellen eines Express-Servers von Grund auf mit Nodemon

1. Gehe zum Verzeichnis `/home/project`.

   ```
   cd /home/project
   ```

2. Erstellen Sie ein Verzeichnis namens `myexpressapp` und wechseln Sie in dieses Verzeichnis.

   ```
   mkdir myexpressapp
   cd myexpressapp
   ```

3. Führen Sie jetzt `npm init` aus.

Dies wird das API-Verzeichnis initialisieren, um als Webanwendung zu dienen. Folgen Sie den Aufforderungen auf dem Bildschirm, um die Initialisierung abzuschließen.

- Der Paketname ist standardmäßig der Name des aktuellen Ordners (in diesem Fall myexpressapp). Sie können einen anderen Namen angeben, wenn Sie möchten.
- Als Nächstes werden Sie nach der Version gefragt, die Sie festlegen möchten. Der Standardwert ist 1.0.0.
- Dann werden Sie nach einer Beschreibung gefragt, in der Sie eine kurze Beschreibung dessen geben können, was die API beabsichtigt zu tun. Sie können es leer lassen.
- Als Nächstes geben wir den Einstiegspunkt in die API an, der standardmäßig index.js ist.
- Wenn Sie nach dem Autor gefragt werden, können Sie Ihren Namen angeben oder es leer lassen.
- Die Lizenz ist standardmäßig ISC (Internet Systems Consortium), was bedeutet, dass es sich um eine permissive Lizenz handelt, die es den Menschen erlaubt, alles mit Ihrem Code zu tun, mit ordnungsgemäßer Attribution und ohne Gewährleistung.
- Es wird die Inhalte für Ihre package.json generieren, eine Datei, die alle Pakete verfolgt, die Ihre Serveranwendung benötigt, und fragt Sie, ob die Details in Ordnung sind. Sobald Sie bestätigen, werden die Details in die package.json geschrieben.

4. Führen Sie jetzt den folgenden Befehl aus, um express zu installieren.

   ```
   npm install express --save
   ```

Die `--save` Option stellt sicher, dass die package.json aktualisiert wird.

5. Führen Sie jetzt den Befehl `touch index.js` aus. Sie werden sehen, dass diese Datei im Datei-Explorer erstellt wird. Sie können die IDE verwenden, um den Code, den Sie aus den vorherigen Übungen gelernt haben, darin zu schreiben.

▶ Beispielcode wurde hier gegeben

6. Ändern Sie `package.json`, um den Server mit npm start zu starten. Fügen Sie `"start"` unter Skripte hinzu.

   ```
   {
     "name": "myexpressapp",
     "version": "1.0.0",
     "description": "",
   ```

```
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^X.X.X"
  }
}
```

7. Vom Befehlszeilenprompt aus kannst du jetzt `npm start` ausführen, um den Server zu starten.

8. Jetzt kannst du weitere Endpunkte hinzufügen oder Änderungen am Server vornehmen, wie es nötig ist. Es kann jedoch sehr frustrierend sein, den Server jedes Mal zu stoppen und neu zu starten, wenn du Änderungen vornimmst. In diesem Fall gibt es ein nützliches Paket. Das Paket heißt nodemon. Jedes Mal, wenn du Änderungen an der Server-API vornimmst, wird der Server automatisch neu gestartet. Lass uns das im selben Verzeichnis installieren, in dem wir unsere index.js erstellt haben. Wir werden es als Entwicklungsabhängigkeit mit der –save-dev-Option installieren und speichern, da wir dies nur verwenden möchten, wenn wir den Server lokal in unserer Entwicklungsumgebung ausführen.

   ```
   npm install --save-dev nodemon
   ```

9. Sobald nodemon installiert ist, werden wir Änderungen an der package.json vornehmen, um dies zu nutzen und das Skript neu zu starten, wenn es Änderungen gibt. Wir werden `"start" : "nodemon index.js"` im Abschnitt “scripts” unserer package.json einfügen. Mit den Änderungen wird die package.json folgendermaßen aussehen.

   ```
   {
     "name": "myexpressapp",
     "version": "1.0.0",
     "description": "",
     "main": "index.js",
     "scripts": {
       "start" : "nodemon index.js",
       "test": "echo \"Error: no test specified\" && exit 1"
     },
     "author": "",
     "license": "ISC",
     "dependencies": {
       "express": "^X.X.X"
     },
     "devDependencies": {
       "nodemon": "^X.X.X"
     }
   }
   ```

Führen Sie jetzt im Befehlszeilenprompt `npm start` aus, um den Webserver zu starten.
Nehmen Sie nun eine Änderung vor oder fügen Sie einen weiteren Endpunkt hinzu und sehen Sie, ob der Server neu gestartet wird und die Änderungen ohne ausdrückliches Neustarten angezeigt werden. Magie!

Herzlichen Glückwunsch! Sie haben das Labor für Express JS abgeschlossen.

Zusammenfassung

Jetzt, da Sie gelernt haben, wie man einen Express-Server erstellt und ausführt und wie man Middleware, Vorlagen und Routing verwendet, werden wir weiter lernen, wie man Clients erstellt, um sich mit den Servern zu verbinden.

Author(s)

Lavanya