Hello in this tutorial, we will understand how to manage sessions in a Node.js application. <strong>HTTP</strong> is a <em>stateless protocol</em> and to manage sessions in a Node.js application we will use the <code>express-session</code> dependency.

<h2>1. Introduction</h2>

To set up this application we will put the <code>expression-session</code> and <code>cookie-parser</code> middleware dependencies in place. But before doing any programming stuff we will need to install Node.js on windows and you can download the installer from <a href="https://nodejs.org/en/download/" target="\_blank">this</a> link. Click on the installer (also include the NPM package manager) for your platform and run the installer to start with the Node.js setup wizard. Follow the wizard steps and click on Finish when it is done. If everything goes well you can navigate to the command prompt to verify if the installation was successful as shown in Fig. 1.

// Fig. 1: Verifying node and npm installation

<h2>2. Express.js Sessions Tutorial</h2>

To set up the Express.js server, we will need to navigate a path where our project will reside. For programming stuff, I am using <a href="https://code.visualstudio.com/" target="\_blank">Visual Studio Code</a> as my preferred IDE. You're free to choose the IDE of your choice.

<h3>2.1 Setting up dependencies</h3>

Navigate to a directory and run <code>npm init -y</code> to create a <code>package.json</code> file. This <a href="https://docs.npmjs.com/creating-a-package-json-file" target="\_blank">file</a> holds the metadata relevant to the project and is used for managing the project dependencies, script, version, etc. Add the following code to the file wherein we will specify the dependencies.

<span style="text-decoration: underline;"><em>package.json</em></span>

<pre class="brush:json;">

{

"name": "expressjs-session",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node index.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"express": "^4.17.1",

"express-session": "^1.17.2",

"cookie-parser": "^1.4.5"

},

"devDependencies": {

"nodemon": "^2.0.7"

}

}

</pre>

To download the dependencies navigate to the directory path containing the file and use the <code>npm install</code> command. If everything goes well the dependencies will be loaded inside the <code>node\_modules</code> folder and you are good to go with the further steps.

<h3>2.2 Creating an Express server</h3>

Add the following programming stuff to the <code>index.js</code> file and we will import the <code>expression-session</code> and <code>cookie-parser</code> middleware modules. Whenever a user will make an introductory request a new session will be created for the user. Now when another request from the same user comes, the application will already have their session information stored.

<span style="text-decoration: underline;"><em>index.js</em></span>

<pre class="brush:js;">

const express = require('express');

const cookieParser = require('cookie-parser');

const session = require('express-session');

const app = express();

app.use(cookieParser());

// using secret for cookie handling

app.use(session(

{

secret: 'Your\_Secret\_Key', // used to sign the session id cookie

saveUninitialized: true, // forces the session that is "uninitialized" to be saved to the store

resave: false // forces the session to be saved back to the session store, even if a session was not modified by request

}

));

// endpoint

// http://localhost:4001/session

app.get('/session', function (req, res) {

if (req.session.page\_views) {

// incrementing the page views counter by 1

req.session.page\_views++;

res.status(200).json({info: `Welcome to this tutorial. Visit counter : ${req.session.page\_views}`});

} else {

// introductory request

// setting the page views counter to 1

req.session.page\_views = 1;

res.status(200).json({info: 'Welcome to this tutorial for the first time'});

}

});

const PORT = process.env.port || 4001;

app.listen(PORT, () =&gt; {

console.log(`Server started on port ${PORT}`);

});

</pre>

<h2>3. Run the Application</h2>

To launch the application navigate to the project directory and enter the following command as shown in Fig. 2. If everything goes well the application will be started successfully on port number <code>4001</code>.

// Fig. 2: Launch the application

<h2>4. Project Demo</h2>

When the application is started, open the browser and enter the following url - <code>http://localhost:4001/session</code>. If the request will be the first the following message will be shown to the user - <em>Welcome to this tutorial for the first time</em> and if the user sends the repeated requests the message with display a page visit counter. That is all for this tutorial and I hope the article served you with whatever you were looking for. Happy Learning and do not forget to share!

<h2>5. Summary</h2>

In this tutorial, we learned –

<ul>

<li>Steps to setup Node.js</li>

<li>Sample programming to understand Session management in Express.js server</li>

</ul>

You can download the source code of this tutorial from the <a href="#projectDownload">Downloads</a> section.

<h2><a name="projectDownload"></a>6. Download the Project</h2>

This was a programming tutorial on session management in an Express.js server.

<div class="download"><strong>Download</strong><br />You can download the full source code of this example here: </div>

===================================================================================

Application name - Express.js Sessions Tutorial