**+7.1 Intro to Express**

|  |  |
| --- | --- |
|  |  |
|  | Welcome to the first of 6 weeks of what we call the ‘Back-End’ aspects of software development.  It’s about servers, databases and how we join up computers so that many people (or just one) can access the data that you allow them to – possibly at the same time – but with a good level of security.   We have seen how React runs on the PC belonging to the user, but we will be looking at how everything else will go via a server. |
|  | READ. |
|  | The following analogy shows how the internet works. How do we interact with websites and web apps.? |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | These following slides will be useful for you as they give the definitions to the keywords that we will be using a lot of from now on.  So, anyone using the website is a **User**. |
|  | The **client** is the computer sending the request.  READ.  (These keywords are important) |
|  | The **server** is the computer that accepts requests from clients and returns responses.  READ.  ‘A server listens’. This is important as it is expecting communications from the internet (a packet of data).  So, a packet of data comes into the machine (IP address), the server looks at it a examines whether it needs to address the security protocols, follows a particular format, and once it understands the format and meets the criteria then the data is sent down a particular **route**.  The server then talks to a **database,** and it pulls the information that is relevant, then sends it back to the **client** as a **response**.    Of course, it doesn’t have to be looking for a particular book – it will have different things to look for (website things) and so different routes do different things. |
|  | READ. We will be looking at Routes in more detail, soon. |
|  | It is basically a file system where data is stored. It is really just meaningless data dumped in a systematic way into text files.  We can access specific data to make the data more meaningful for us.  We might need to **Create** a new record, or **Read** – this is the majority of usages of databases, **Update** and Delete. |
|  | We will look at this in a lot more detail further along this course where we will **CREATE**, **READ**, **UPDATE** AND **DELETE** things from databases.  This is CRUD. |
|  | This brings us onto Express.  Express is a **node-based library** – written in JavaScript.  It allows you to easily create servers using Node.  The same Node that you used in the front-End weeks to be able to use JavaScript outside of a Browser. |
|  | Express handles all your routes easily.  We can set up Express to define routes which will listen for certain requests. When a **User** enters a particular **route** then Express server will be watching out for (listening to) those **URLs** and will respond in some pre-defined way. You will program Express to give different **responses** to different **requests**.  For example, if you wish to get a list of all books in the library – then that will be one request. Or if you want to add a new book to the library – then that will be another request which will need a different route. |
|  | Q: Is this all brand new to you?  Q: Any questions so far? |
|  | READ.  This morning we will have a go at creating a simple **webserver** which will serve-up a couple of simple web pages. |
|  | But first we need to install it…  READ. |
|  | One thing that a Junior Developer must be able to do it to look up and get used to the documentation for specific packages.  There is a website for node packages that you should add to your favourites called npmjs.com  Visit <https://www.npmjs.com/>  And search for Express.  The documentation will guide you through the installation and pretty much everything you need to get it working.  We need to go and take a look at the express website by clicking this link. Or click this:  <https://expressjs.com/> |
|  | We will also find the API reference section very useful and we will be expecting to use the:   - express.json()  - app.get()  - app.use()  - app.post()   … methods extensively. They give more details with examples of usage – so worth bookmarking this website.   Click on the app.get() method:  It gives examples, etc. |
|  | We will be using req.body quite a lot. |
|  | Click on the Response Tab on left.  The res.send() method sends back a custom response to say whether everything went okay, or not |
|  | Router comes into our lives in a couple of weeks’ time. Don’t worry now.  The important thing to remember at the moment is that we have this Express.js documentation that you, as a junior dev, will be able to work through and get familiar – get up to speed.  As part of your independent working ethic – you have read the documentation before asking your Lead developer how to do things. You can have a more precise discussion with your senior developer when you have a problem. |
|  | TIME TO OPEN UP**: VISUAL STUDIO CODE**  Create a new folder in your **week7 folder**.  New Terminal:  - **PowerShell** comes with Windows on PC  - **bash** comes with Windows GiHub and can help you a bit more when using GitHub  In Terminal: **npm init -y** (-y avoids answering Qs)  … and you should see a package.json file appear. |
|  | Create a file called ‘.gitignore’ and add the line:   This prevents a big file being added to our Git repository later on. |
|  | Now let’s install Express. In Terminal type:   **npm install express**  … and you should see a folder called **node\_modules** which contains all of the things to run Express. You can examine the code if you want – but you generally don’t have to worry about it. |
|  | We can see the dependency express has been added. |
|  | We are now going to use Express to create a server.  For the sake of convention, we will create a new folder called ‘**src’** under the main directory. |
|  | Within that src folder, create a new file called ‘**server.js’:** |
|  | We should really import the Express Library into our project… |
|  | We need to set up some routes.  **app.use** takes 2 parameters:  1st = What you declare the route to be. (‘**aboutme’** in this case)  2nd = A method for creating static webpages and we will tell it to look in a folder called ‘**about’** for the HTML. |
|  | We must now tell **Express** to listen on these things called **ports**. Each server has thousands of potential ports and there are standards on what those ports are. A port is like an extra extension to an address. Listeners will ignore traffic that doesn’t go through this port. A server can be doing multiple things at any time, e.g. acting as a web server on port 80 whilst acting as an sql database on port 336. So it allows one type of information to come in on a particular port. |
|  | Visit: [**https://www.hostpapa.com/knowledgebase/commonly-used-ports/**](https://www.hostpapa.com/knowledgebase/commonly-used-ports/)  … and look at the ports to see the standards for the port numbers and what they do.  Some are used for email sending and receiving – not required reading but interesting for developers. You just need to know that there are these things called **ports** and we need to use the correct one when setting up a server and that we are not **conflicting** with other ports used for different things. |
|  | So, let’s add his line to have the server listening on port 5001 and simply console log a line – to make sure that it’s working. |
|  | Save and Run in Terminal using:  **node src/server.js** |
|  | Q: Did you all get this to work? |
|  | We now have a running server – but we don’t have anything to serve because the books directory is empty! It is expecting some **HTML**.  So, let’s give it some HTML to work with. |
|  |  |
|  | This server is now running on our PC locally – to you.  We can stop the server using: **CTRL+C**  We can start the server using: **node src/server.js** |
|  |  |
|  |  |
|  |  |
|  |  |