Databases, Networks And The Web
Course Notes

Felipe Balbi

May 7, 2020

Contents

Week 1 **4**
1.201 What is a web application? 4
1.203 Further reading 4
1.205 The life and times of a web request 5
1.207 Accessing remote resources, HTTP 6

Week 2 **7**
1.301 Three-tier web application architecture 7
1.304 Information retrieval activity 7

Week 3 **8**
2.001 Introduction to Node.js and Express 8
2.101 Web servers 8
2.103 Essential reading 8
2.104 Web server architecture 8
2.106 Essential reading 9
2.107 Web hosting 10
2.201 Introduction to Node.js and Express 10
2.204 Node.js 10

Week 4 **11**
2.503 Express 11
2.601 Summary and Further reading 11

Week 5 **12**
3.201 Separation of concerns (SoC) 12

Week 6 **13**
3.401 Templating engines 13
3.407 Further reading, Pug 13
3.501 Summary and Further reading 14

Week 7 **15**
4.001 Introduction 15
4.103 Form-data validation 15

Week 8 **16**
4.201 GET and POST request methods 16

4.203 HTTP methods 16
4.301 Summary and Further reading 16

Week 1

Key Concepts

- recognise the tools available for this module to edit a node.js file and run it
- describe what static and dynamic web applications are.

1.201 What is a web application?

Data is everywhere around us. For example our account balance is a form of data which gets checked and updated during a commercial transaction.

Web application is a client-server software application in which the user interface runs in a browser[1]. It could be a computer program which allows the user to submit and retrieve data to and from a database.

Static Web Applications

Web applications with little or no interaction with the user.

Dynamic Web Applications

Web applications which allow the user to input, change, and manipulate data.

Desktop vs Web Applications

Desktop	Web
Accessed through OS	Accessed through Web Browser
Different appearance in each OS	Consistent appearenc accross platforms
Fast access to system resources	Slow access to system resources
Lower risk of data loss	Higher risk of data loss
Different version for each OS	Same version accross all platforms
Multiple updates required	Single update for all users

1.203 Further reading

Some useful Further reading is:
`https://techterms.com/definition/web_application`

[1] `https://en.wikipedia.org/wiki/Web_application`

1.205 The life and times of a web request

What happens when we call a web application? We know everything starts with typing the URL in the web browser, but what happens then?

Calling a Web Application

This better shown with the flowchart 1 below.

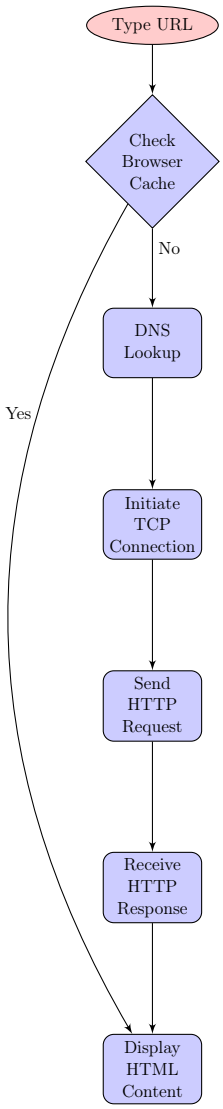

Figure 1: Calling a Web Application

HTTP Protocol

The HTTP Protocol is based on a Request/Response architecture. The client, i.e. the web browser, makes a request for a particular resource. The server returns a response to the browser. This response may or may include the requested resource. This is because errors may occur.

After the browser receives the response, it will evaluate it and decide what and how to display it on the screen.

What if we're dealing with a dynamic web application? Where will the content for the web page come from?

Dynamic web sites are based on programs which run on the web server when an HTTP Request comes in. These programs will generate the content on the fly for the user. One major source of such content is a database.

1.207 Accessing remote resources, HTTP

Ceri, S. et al. Designing data-intensive web applications. (San Francisco, CA: Morgan Kauffman Publishers, 2003). [ISBN 9780080503936].

Chapter 1, p.5-8, Accessing remote resources: the hypertext transfer protocol.

Accessible from here.

Week 2

Key Concepts

- describe what 3-tier web application architecture is.

1.301 Three-tier web application architecture

A three-tier web application architecture is a specialization of the more generic n-tier architecture.

In three-tier architecture, the three modules are as follows:

Presentation Tier Commonly referred to as the *Front-end*, it is responsible for receiving input and displaying output. In general, this part is written in HTML, CSS, JavaScript.

Application Tier Commonly referred to as the *Middleware*, it is responsible for the business logic and calculations. In general, this part is written in JavaScript, PHP, Ruby, Python, and many others.

Data Tier Commonly referred to as the *Back-end*, it is responsible for storing and managing the data the application requires. In general, this part is written in SQL.

The main idea is to keep presentation, application logic and data store separate from each other. This allows each part to be developed and maintained as separate modules.

A web application can be split into two main blocks, depending on where they run:

Client Side The part of the application that runs on the client's computer. Commonly, only the Front-end runs on the client.

Server Side Composed of both the Middleware and the Back-end, it runs on servers (or cloud instances) owned by whoever made the application.

1.304 Information retrieval activity

Ceri, S. et al. Designing data-intensive web applications. (San Francisco, CA: Morgan Kauffman Publishers, 2003). [ISBN 9780080503936].

Chapter 1, section 1.5.7 p.54-55, Three-tier architectures.

Accessible from here.

Week 3

Key Concepts

- explain what a web server is
- use the tools available for this module to edit a simple Node.js web server and run it.

2.001 Introduction to Node.js and Express

During this topic, we create our first web server. The end goal being creating a full web application.

We learn about `Node.js` and `Express.js` which we will use to write our web server.

2.101 Web servers

A web server is a program that uses HTTP to serve web pages in the form of files to their users.

The users send requests to the web servers and its response is a web page.

HTTP, or HyperText Transfer Protocol, is a method for encoding requests and responses. A method of communication between the server and client which defines the rules of interaction.

2.103 Essential reading

Ceri, S. et al. Designing data-intensive web applications. (San Francisco, CA: Morgan Kauffman Publishers, 2003). [ISBN 9780080503936].

Chapter 1, p.5-8, Accessing remote resources: the hypertext transfer protocol

Accessible from here.

2.104 Web server architecture

Figure 2 below presents a common web server architecture. It contains the basic elements of a web server which is composed of the underlying HW, the Operating System, an HTTP Server, a Database, and a Scripting Language Runtime.

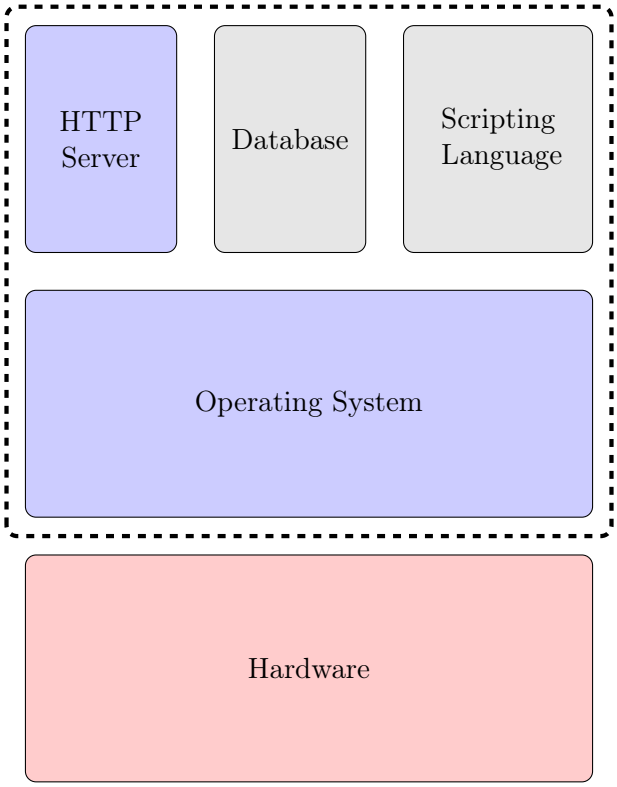

Figure 2: Web Server Architecture

The hardware for a web server could be anything from an embedded, low power device, such as Raspberry Pi or a Beagle Bone, all the way to a set of standard rack-mounted servers running Intel Xeon server-grade CPUs.

The Sofware side of the Web Server includes at least an Operating System, commonly Linux, and an HTTP Server such as Apache, Nginx, etc.

Usually, these also include a Database (MySQL, PostgreSQL, MongoDB, etc) and a Scripting Language runtime (NodeJS, Ruby, Python, Perl, and so on).

2.106 Essential reading

Mendez, M. The missing link: an introduction to web development and programming. (Geneseo, NY: Open SUNY textbooks, 2014). [ISBN 9781502447968].

Refer to Chapter 3 to learn more about web server architecture.

Accessible from here.

2.107 Web hosting

Read about web hosting at the site given below and then answer the questions in the discussion forum.

website.com 'Web hosting: what is web hosting?'

Accessible from here.

2.201 Introduction to Node.js and Express

NodeJS is a server-side JavaScript runtime. It's open-source and cross-platform.

It's implemented as an asynchronous, event-driven runtime environment. Event-driven means that the flow of the program is determined by events received by the program, rather than the sequence of the code. Asynchronous means that NodeJS doesn't wait for responses from external sources.

Because of these characteristics, NodeJS is very good for I/O-intensive applications. It's not, however, a good pick for CPU-intensive applications.

2.204 Node.js

Krause, J. Programming web applications with Node, Express and Pug. (New York, NY: Apress Media LLC, 2017). [ISBN 9781484225110]

Chapter 3 Introduction to Node, pp.15–46. You can refer to this chapter throughout Topics 2–4.

Accessible from here.

Open.js foundation 'AboutNode.js'.

Accessible from here.

Week 4

Key Concepts

- use the tools available for this module to edit an Express.js web server and run it.

2.503 Express

Yaapa, H. Express web application development: learn how to develop web applications with the Express framework from scratch. (Birmingham: Packt Publishing Ltd., 2013). [ISBN 9781849696555]

- Chapter 1, pp.24–26, What is Express?
- Chapter 1, pp.30–36, The stuff that makes up Express
- Chapter 2, pp.51–55, Your first Express app

Accessible from here.

2.601 Summary and Further reading

Krause, J. Programming web applications with Node, Express and Pug. (New York, NY: Apress Media LLC, 2017). [ISBN 9781484225110]
Accessible from here.

Week 5

Key Concepts

- describe and apply routing in web server development
- describe and apply the 'separation of concerns' programming principle in your web application development.

3.201 Separation of concerns (SoC)

SoC is the idea that each module in a web application should be responsible for one thing.

It's a design principle where we aim at isolating modules from each other and have each module address a separate concern.

A concern can be "presenting data to the user" or "connecting to a database".

In order to implement a web application using this design principle, we must first define the concerns. Then we can design a separate module or layer for dealing with that concern. One thing to remember is that we also aim at minimizing coupling of the different concerns by making a minimal interface through which the modules communicate.

SoC is about planning and designing logical layers in a way that it is loosely coupled by interfaces with other layers. What we mean by that is that modules should indepent from each other, thus allowing any module to be replaced by another module that implements the same interface.

SoC can be expressed at different levels: functions, modules, controls, widgets, tiers, services, etc. It's not limited to large layers within an application.

SoC helps us reduces complexity by means of encapsulation, it aids us with keeping up with the DRY (*Don't Repeat Yourself*) principles, improves portability of the application and improves maintainability and testability of the application.

Week 6

Key Concepts

- describe and apply templating in web application development.

3.401 Templating engines

Templating engines allow us to produce content dynamically while also allowing us to keep HTML separate from JavaScript.

With that, we can produce Web Applications that are more interactive and respond to users' input.

In summary, our web application will contain static template files containing static content and variables. During runtime, the template engine will replace the variables with actual values and compile HTML files to be served to the user.

Usually, template engines also allow us to render e.g. a list of elements from a single template by using a loop.

Some popular template engines are:

- EJS
- Pug
- Mustache

During this course we focus on EJS.

An EJS file is just a regular HTML file renamed to have the extension `.ejs`. It contains two special tag elements, they are:

<%= %> Used to output the return value of a javascript expression into the document

<% %> Used to evaluate an expression but it won't add the result of evaluation to the document.

3.407 Further reading, Pug

For more information on the Pug templating engine refer to the following:

Krause, J. Programming web applications with Node, Express and Pug. (New York, NY: Apress Media LLC, 2017). [ISBN 9781484225110].

- Chapter 6, Introduction to Pug
- Example pug templates pp.83, 84, 85
- Chapter 7, Language components of Pug

Accessible from here.

3.501 Summary and Further reading

In this topic you learned about:

- the separation of concerns (SoC) principle of programming
- organising your web applications into separate folders
- templating engines.

Some Further reading you may be interested in is:
Krause, J. Programming web applications with Node, Express and Pug. (New York, NY: Apress Media LLC, 2017). [ISBN 9781484225110].

- Chapters 3, 4 and 5

  Accessible from here.

Yaapa, H. Express web application development. (Birmingham: Packt Publishing Ltd., 2013). [ISBN 9781849696555].

- Chapters 1, 2 and 3

  Accessible from here.

Week 7

Key Concepts

- describe and apply form handling in your web application.

4.001 Introduction

Input forms are one way to provide user interaction in a dynamic web application. We learn how to create a form using HTML tags and how to display it to the users in the front end of a dynamic web application.

4.103 Form-data validation

Read about validation in JavaScript:

- `https://www.tutorialspoint.com/javascript/javascript_form_validations.htm`

Read about express-validator module:

- `https://www.npmjs.com/package/express-validator`

Try to find answers to these questions:

- What is form-data validation?
- What are the different types of form data validation?
- How can it be achived?
- How can express-validator module facilitate form-data validation?

Week 8

Key Concepts

- describe and apply GET and POST request methods in your web application
- describe and run the code to retrieve form data in middleware.

4.201 GET and POST request methods

GET and *POST* are two of the main HTTP methods. *GET* is used to request data from a resource on a web server while *POST* is used to send data to create or update a resource on a web server.

More information about HTTP methods can be found here.

4.203 HTTP methods

Yaapa, H. Express web application development. (Birmingham: Packt Publishing Ltd., 2013). [ISBN 9781849696555].

pp.86–87, A quick introduction to HTTP verbs

Accessible from here.

4.301 Summary and Further reading

In this topic you learned about:

- creating forms using HTML
- accessing forms from middleware (res.send(), res.render())
- collecting form data in middleware (main.js) in Node.js
- collecting form data when the request method is GET
- collecting form data when the request method is GET.

Some Further reading you may be interested in is:

Krause, J. Programming web applications with Node, Express and Pug. (New York, NY: Apress Media LLC, 2017). [ISBN 9781484225110].

Accessible from here.