Liam’s Guide to Tools for Web Development Projects

The document is intended to give you some basic guidance on the various tools available for web development. Only a few tools are covered and like always, there are many many ways to build a website/system. The intention is to point out some tools and their features, so that you know they exist and can look into them yourself if you think they would be useful.

1 Version Control

Version control allows you to track files and the changes between files. You can go back in time to set points. This allows you to try our new features (i.e., change the code) and if it doesn’t workout then you can revert back to an earlier point in time when the code did work.

I would **strongly recommend** that all the code you write (even for non-web projects) is placed under version control using a cloud git host.

If you don’t want to read any more in this section, then just know that **Liam recommends you use git with all your code, placing it on Bitbucket as a git host. In case you don’t believe me, here is a quote by one of my past project students.**

A quote by Andreea Capatan (One of Liam’s project students 2018/2019).

> For my third year project, I decided to build a web application and after a short meeting with Liam where he gave me some suggestions, I decided on using Laravel for it. I had never used it before and one of the first things he had suggested was to use git throughout the whole process of the implementation. For some reason, I thought it was ok to just start typing away on my code and simply save the files locally. After a few weeks I reached a point in my project where things started to go wrong, and when I tried to fix the errors, I ended up messing up the parts that were working beforehand. One of the main reasons why I didn’t use git at the beginning was because I heard people say that it was difficult to use and I didn’t want to have to deal with learning another new tool. When I told Liam that I was stuck, he suggested to just revert to the previous commit on git and at that point I was a bit too scared to tell him I didn't use git. He was shocked when I told him, but he helped me set it up, showed me how it works and suggested Bitbucket for version control. I can honestly say, it was the best decision and it made life so so much easier. It is so easy to use and with bitbucket you can look back at all the commits, the changes you’ve made and view every single file’s history. If you are using it for a solo project, it’s even easier as you don’t have to deal with merge conflicts and tricky stuff that comes up when two people try to edit the same file. My advice is to use git from the very beginning as it will make life so much easier, saves a lot of time and it’s always good to know how to work with git for any future work you do. It is so easy to use, please don’t avoid it like I did.

1.1 Git

Git is the current standard tool for version control. It is a distributed system which can make it a little hard to get used to. It is well worth learning git. Git is an essential tool across the entire software development industry. If a company is not using git (I would guess 80% to 90% of software developers are) then they will be using some other version control system.

Git stores files in entities called **commits**. Each commit stores the **changes** of files from the last commit. They also store meta information such as date/time of when commit was made, author, and the id of the previous (parent) commit. Git stores all off this information in a hidden .git folder within your local project directory on your computer. If you delete the main project folder (that contains the .git folder) then you have deleted everything! See Section 1.3 for using git with cloud services to keep everything safe.

At a minimum you need to know how to:

- Create commits - create a new “save point” that you can get back to later.
- Checkout commits - move around the history of the commits.

1.2 The Git Tools

There are various tools for git. The basic one is a command line tool simply named git. Where you run basic commands via a terminal. There is also a command line gitk tool for visualising the history of your commits.

There are various GUI git clients as well - some are listed in Section 1.3.4.

1.2.1 Which Files to Place Under Version Control?

Normally, you place only text files under version control. But, you can also add any resource files that your project uses such as images, sounds, etc.
You should never place automatically generated files under version control. These include compile output files (e.g., .class files from javac, .pyc files from Python, etc), temporary files, build directories (e.g., bin folders), etc. You only want to track your code and possibly the project configuration files for your IDE.

1.3 Cloud Git Hosts

Git stores all of its information in a .git folder on your computer (within the top-level project folder). It is dangerous if this is the only copy of your git repository. As git is distributed, it contains mechanisms to copy the repository onto other computers. I would suggest using a free git host for this.

1.3.1 Github

GitHub is a very popular git host. It offers free and paid packages. Previously, the free packages requires you to make your git repository open source and publically available unless you sign up for a time limited student account. Recently, GitHub allows for free private repositories and allows you to access it with a team of up to 3 people. However, I have already have many repositories in BitBucket, thus, I do not use GitHub.

1.3.2 Bitbucket

Bitbucket is similar to GitHub. I really like Bitbucket and use it for any new code project. There are a few limitations on the free account, the main one being that you can only work with a team of up to 5 people (2 more than GitHub). **Liam recommends you use this git host (it is good and allows unlimited private repositories, so it can grow with you after you finish being a student).**

1.3.3 GitLab

Another popular git host is GitLab, but I do not know anything about this. It seems to be free with unlimited private repositories and team sizes, and full access to most features. This might be the best choice - but I don't have experience with them.

1.3.4 Git Resources:

Atlassian, the company behind Bitbucket (see Section 1.3.2) provide some good resources:

- https://www.atlassian.com/git - Various guides and interactive tutorial to learn git (focused of course on Bitbucket).
- https://www.sourcetreeapp.com - A reasonably simple, cross platform, GUI git client (that also works with all major cloud git hosts). **Liam recommends either using this tool or the command line git tool (depending on how comfortable you are at using a terminal/command-line).**

2 Web Development

Here we point out a few general concepts that you should know a little about before going further and looking at frameworks for developing websites.

2.1 General Concepts

Please read and roughly understand the idea of these general concepts. Understanding the idea will help you learn the web frameworks.

2.1.1 Model-View-Controller

Model-View-Controller (MVC) is used extensively in modern web applications. Basically:

- The model contains the main business data for your application. For a connect-4 game this would be the grid (i.e., what cells are filled by which token), whos turn it is, the player names, etc.
- The view would be the visual representation of the state of the game (i.e., the model). The view displays the model.

- The controller contains the behaviour (functions/methods) that allows the user to interact with the model. The view might call methods in the controller. In connect-4 one method might be to place a token in a certain column of the grid (the method would apply logic to work out where the token falls).

A short article on this is https://www.codecademy.com/articles/mvc.

2.1.2 Active Record Design Pattern

The Active Record design pattern is used to interact with databases without using SQL. A class is used to represent a table. Each object of the class represents one row in that particular database table. Change the model and the changes will be written back to the database. This way, you do not write SQL.

A short article on this is https://en.wikipedia.org/wiki/Active_record_pattern

2.1.3 REST and CRUD

REST is a popular way of building web applications these days. It basically says that the server does not track any state of the users session. All the information needed to perform a task is sent to the server as part of each request.

CRUD is a way to organise URLs and operations. If we are creating URLs to allow the managing of a list of tasks then we will want something like:

- To get the current list of tasks:
    - GET http://www.myrestaurant.com/dishes/
- To get the details of a specific task (with id 42 in this case):
    - GET http://www.myrestaurant.com/dishes/42/
- To get the form create a new URL:
    - https://www.supertasklist.com/tasks/create
- To actual create a new task send the data to:
    - POST https://www.supertasklist.com/tasks
- Many others for editing and deleting.

Read more at https://www.codecademy.com/articles/what-is-crud

2.2 Full Web Development Frameworks

Full web development frameworks allow you to design the whole site as you want. They do not provide a working site - you build everything. However, you have full control.

2.2.1 C# and ASP.NET Core MVC 5

Using Visual Studio you can create a Website in C# using ASP.Net Core MVC. Sean Walton used to teach this as part of the Web Apps module but has now switched to Laravel. There is not a huge amount of good help available online. Please note that you want to use ASP.Net Core and not ASP.Net (without the word core). The new cross-platform version is the one with the word “core”.

2.2.2 Laravel

Laravel is a good system. This is Liam’s favourite web framework right now. It is not easy and you will need to program the entire system. That said **you will gain a vast amount of skills needed for a career as a Software/Web Developer**. Sean Walton has recently decided to switch the Web Apps module from ASP.Net Core MVC 5 to Laravel.

Laracasts has **very good** video series for learning laravel related topics (most are not free). Watch this **free** video series, pause and duplicate what you see and you will learn everything you need to learn to build a website in Laravel.

I have also created a video that guides you through the installation of Laravel using Homestead, VirtualBox and Vagrant. This video can be found at https://www.youtube.com/watch?v=MBWXfaX9Gus

Some main concepts:

- Setup and deployment: Laravel requires quite a few packages to be installed. A normal way to develop is to use VirtualBox to create a virtual machine that you will use to do development on. Vagrant is a tool you use to setup the virtual machine (so you don’t need to do it manually). Homestead is a particular ‘Box’ that Vagrant uses to setup your Virtual Machine. All this is explained in the Laravel Documentation.
- Composer: PHP Package manager to install various tools (Laravel is installed using composer).
- Artisan: Main tool of Laravel (comes with it) for creating boilerplate code.
- Migrations: You do not build a database by writing manual SQL. You specify the database structure in a migration. You run migrations to execute the code to create the database.
- Seeds: A file that creates dummy data in your database to allow you to test the system.
- Factory: Usually used with a seed file. You might have a User factory that creates randomly generated users (with real looking data like email addresses and postal addresses).
- Models: A model represents a row in the database - see Section 2.1.1 and Section 2.1.2.
- Controllers: These service different URLs and makes your app do things when these URLs are visited.
- Routes: Routes allow you to point URLs at Controllers.
- Views / Blade: You build Blade templates to display your data in HTML pages. The controllers execute your blade templates.

2.3 Content Management Systems

Content Management Systems will provide a whole website where you can edit and create content via a web interface. You can implement your own themes and plugins to customise and provide advanced features. But for basic pages you just edit using the web interface.

Some examples:

- WordPress: This is a Blog based CMS. It can be hard to 'turn off' the Blog look and feel. You would need to implement a plugin(s) and theme to implement a system in this. WordPress is a free Open Source Project (https://wordpress.org), but there is also a paid hosting service too (https://wordpress.com).
- October: This is a very low level CMS. You have to program and setup much more than when using WordPress, but you get much more freedom. This particular CMS is based on Laravel.

3 Single Page Applications in Laravel

Single Page Applications (SPAs) are somewhat popular. These are where the entire web application is a **single page**. This page uses client side Javascript to issue asynchronous requests (Ajax to the back-end web-server for data and then changes the displayed page to incorporate the new data.

Writing a SPA is **much harder** and **a lot more effort** that writing a traditional style website. The list of tools and the software stack that you will end up using is also a **lot longer**. **You will need to build two systems:** A back-end server and a front-end-client that work together.

3.1 Back-end

You would create a Laravel system which has models and API routes. These API routes would send and receive JSON data to the front-end client. The back-end might also provide a single normal URL (i.e., non-api) in order to deliver the SPA client front end to the web browser.

Laravel can be used for this purpose: you create models and controllers, but do not use Blade templates (views). Instead you return JSON directly from your controllers. In the case that you definitely will not use Blade at all then you can consider using Laravel's Lumen framework (a cut-down version of Laravel for creating pure API back-ends).

3.2 Front-end

You would develop the front-end client in another framework such as AngularJS (or its successor Angular), VueJS, etc.

3.3 Technology Stack

New frameworks like VueJS and Angular use languages that not all common web-browsers support yet. So how does a web-browser run the code? You setup your web-server to "transpile" the front-end code (compile it from something like TypeScript back to common Javascript, which common web-browsers understand and can run).

Some tools to know about:

- NodeJS: A system for running Javascript on a server. This is mainly used to run various development tools such as build utilities.
- NPM: A package manager for NodeJS that allows you to install various Node modules.
- WebPack: Tool for running various build processes during your development. E.g., running Bable, “compiling” your CSS fif you use SASS or LESS.
- LaravelMix: Built on top of WebPack. You would use LaravelMix instead of WebPack. LaravelMix is a simpler and easier version of WebPack.
- Babel: Will transpile your front-end code (normally run via LaravelMix via a npm command).