Server-side Web Application Seminar

Yasmin Mushahdi
Kurs 1dv612 Webb Architecture
2020 feb-march

What does serverless architecture refer to?
In short: serverless essentially refers to a software architecture that uses a third- party service for hosting the software. This means that no server-code is ever written, and the application code consists of client-side code and function calls to the third-part-service server. Examples of these services are such as heroku, Azure, AWS.

What's the trend over the last years where a web application's code is placed?
The current trend seems to be: moving code to the client side, by letting the client-code handle all logic and using API's or third-party-services to handle server side actions. Cloud-based solutions for this kind are also popular.

What's a server-side web framework (a.k.a. "web application framework")?
A server-side web framework offers a complete and standardized arrangement for building applications with the back-end as the main focus. It comes with prearranged generic functionality that makes it easy to extend and use it for building apps. A framework's main functionality is to handle the application code-flow.

The difference between a framework and a library is; a framework acts upon your code, which is different from a library, where your code is acting upon the library code.

What can a server-side web framework do for you?
The server-side frameworks has useful tools that makes it easy to work with a variety of complex technologies, e.g. communication to database, user-authentication/authorization, security, routing, changing data format, about every task that includes in a standard web application.
And in a way, it promotes, (forces) the developer to follow a specific code structure throughout the project, e.g ExpressJS uses the MVC structure.

What's the difference between an ORM and an ODM?
Databases using the ORM organize the data in a table structure, the data is organized by rows and the properties-names are the columns. The ODM databases organize the data in a class-like or object-like, referred to document. These documents belong to a "collection".

ORM (Object Relational Mapping) SQL, NoSQL, MySQL all use ORM, ODM(Object document mapper) MongoDB, ArangoDB, Apache CouchDB, CreateDB.

What factors may affect your selection of a server-side web framework?
The general criteria that could be thought of is,

- **What language** you prefer to use, or learn if needed in order to develop and maintain the product.
- **Producticity**, means how quickly you will be able to add new features and maintain your product. Things to consider here can be:
    1. What type of problem are you trying to solve or what type of application will be built. Some frameworks are better suited to specific applications and some are more flexible and lightweight.
    2. How much do you want to rely on the framework to solve a particular problem? As mentioned earlier, it depends on how flexible you would like to be.
    3. If the framework encourages a specific code structure, for example, Express.js does encourage the Model-View-Controller code structure.
- **Performance** is another thing to consider. What are the framework's strongest performance points and weaker points and do they matter?
- Other things to evaluate are **Caching**, **Scalability** and **Security**. How much do you need, how much does the framework help with and so on.

Whats server-side web frameworks are there for programming languages such as Python, Java, JavaScript, C#, PHP, Ruby, etc.?
Python: Flask and Django
Java….who knows…. Spring Boot maybe?
JavaScript: Express.js and a million more..
C#: ASP.NET
PHP: Laravel
Ruby: Ruby on Rails, Hanami, Sinatra
Perl: Mojolicious

Frameworks for persistent data?
Lido? KodoJDO? Apache Cayenne? MyBatis?

What frameworks are there for Node.js, except for Express.js?
Meteor.js, Nest.js, Sails.js, Koa.js, Hapi.js, Derby.js, Total.js, Adonis.js….as said…millions.

Client-side Web Application Seminar

What are the pros and cons of using a client-side framework in your course assignment stack?

Pros:
No calls to the server.
Can work offline, as the entire web app is already downloaded to the browser.

It is faster after the initial load. (Explained beneath).

Cons:
SPA/client-based will have a longer initial page load, because it sends the entire application (all the files included) the first time. After this, it is faster because the browser already has all the content needed associated with that application.

It is harder for a search bot/search engine to index an SPA. Search Engine Optimization (SEO) does not know how to index an SPA because of the functionality of an SPA, it first has to be downloaded for anyone to have access to the content.

“Never trust the client”. Security is a bugger issue here, for example, extra careful with client inputs, doing extra validation.

What can Service Workers do for your application in the course assignment?
Manage push notifications. Offline experience.