Monday, 14 Nov 2016, 19:35 GMT+8

Hunnan, Shenyang, Liaoning, China
0°C Clear

Small Presentation (2 minutes)

So today I'm here to introduce the basic architecture of general web back-end development.

Back-end is the core of a website, while it carries most of the process & logic of the whole web application. Each request happening in web pages or web APIs would firstly meet the load balancers of the website, containing software balancers & hardware balancers, to dispatch the task to nodes of a distributed system. A mature website won't run on a single server, while the count of servers would be hundreds or thousands in the whole back-end of a big website.

And then comes the main part, view generation (HTML, JSON or XML) in response. These kinds of views carries the necessary information the user or front-end needs, accompanied with the HTTP headers. Currently, most back-end systems generates HTML directly to front-end, which mixes some front-end logic into the back-end logic, enforcing the back-end developers to have some simple knowledge of front-end, like the basic syntax of JavaScript or CSS.

Perhaps the main part contains some operations of databases, which is formally called Data Persistence. The web application needs to keep the necessary information in disk or memory, while memory approaches the data faster and is always used in keeping session data, and disk-based database systems keeping the user information, for example.

So this is the simple logic of web back-end.

---

1. Why single server can't carry the whole big website?
2. It could be noticed that you're introducing the load balancer in such long words. Is that a main part? Would it deserve to cost so much time?
3. How does JSON or XML works? Same as HTML, with rendering the view?
4. I've heard that there are some special kinds of database systems now, which are called NoSQL. What about their applications in web back-end development?
5. How much front-end knowledge does a back-end developer need to know?