Variations

In this lesson, we'll look at variations on the server-side integration with ESI approaches we've already discussed.

WE'LL COVER THE FOLLOWING

Different ESI implementations

Of course, instead of Varnish, a different ESI implementation could be used by Squid or by a CDN like Akamai.

SSI

Another option for server-side frontend integration is **SSI (Server-side includes)**. This is a feature that most web servers offer. https://scs-commerce.github.io/ is an example of a system that uses SSI with the nginx webserver to integrate the frontends.

SSI and ESI have different benefits and disadvantages.

- Web servers are often already available in the infrastructure for SSL/TLS termination or for other reasons. Because web servers can implement SSI, no additional infrastructure is necessary.
- Caches not only speed up applications, but also compensate for web server failures for some time, thus improving resilience. This speaks for ESI and a cache like Varnish. ESI also has more features for further

optimizing caching.

- Correct caching can also be difficult to implement. For example, changing the data of a single data record can trigger a cascade of invalidations. Finally, every page and every HTML fragment containing information about the goods must be regenerated.

Tailor

Tailor is a system for server-side frontend integration that Zalando implemented as part of Mosaic. It is optimized for showing the user the first parts of the HTML page as quickly as possible. For e-commerce, the rapid display of a web page is very important to keep users and can increase sales.

To achieve this, Tailor implements a BigPipe. First, very simple HTML code is transferred to the user in order to be able to display a simple page very quickly. JavaScript is used to load more details step by step. Tailor implements this with asynchronous I/O using Node.js streams.

Client-side integration

Client-side integration does not use any additional infrastructure and can be the simpler option for frontend integration. Therefore, it makes sense to find out how far you can go with client-side integration before using server-side frontend integration.

For the integration of a header and footer, like in the example for this chapter, **server-side integration is the better choice** as a page cannot be displayed without these elements. The pages should be delivered to the user in such a way that they can be displayed to the user **without loading any additional conten**t.

Therefore, **client-side integration** for **optional** elements makes sense. Dealing with failed services is then a task for the client code. That might simplify the server implementation and the server setup.

Shared library

The example from chapter 4 uses a library to deliver assets.

Theoretically, the HTML fragments that are integrated with ESI in this example could also be delivered as a library. But then all systems would have to be rebuilt and deployed for an additional link in the navigation. With the ESI solution, all you have to do is change the HTML fragment on the server.

Additional integration

Pure frontend integration is rarely enough. Therefore, a system will combine backend integration with synchronous (see chapter 9) or asynchronous (see chapter 6) communication mechanisms with frontend integration. An exception is a scenario like in chapter 4, where a complex portal application is implemented. The parts of the portal can communicate through frontend integration. Backend integration is not necessary because the services do not implement a lot of logic and have no database; they only create a web interface.

QUIZ

1 Which of the following is an advantage of SSI?

In the next lesson, we'll look at some experiments to further deepen your SSI understanding with ESI concepts.