ESI in Order & Common

In this lesson, we'll look at how ESI is used in the common and order applications.

We'll cover the following

Order microservice

The *order* microservice offers a normal web interface, which has been supplemented with ESI tags in some places.

A typical HTML page of the *order* microservice looks like this:

```
<html>
<head>
  ...
  <esi:include src="/common/header"></esi:include>
</head>

<body>
  <div class="container">
    <esi:include src="/common/navbar"></esi:include>
    ...
  </div>
  <esi:include src="/common/footer"></esi:include>
</body>
</html>
```

HTML with ESI tags in the example

The *order* microservice is available at port 8090 of the Docker host. The output goes past the Varnish and still contains the ESI tags. At http://localhost:8090/ (http://localhost:8090/) the HTML with the ESI tags can be viewed.

The ESI tags look like normal HTML tags. They only have an `esi` prefix. Of course, a web browser cannot interpret them.

ESI tags in the HTML head

- **line 4**: In the `head` the ESI tags are used to integrate common assets like Bootstrap into all pages. Changing the header under `"/common/header"` causes all pages to get new versions of Bootstrap or other libraries. If the pages with a new version are no longer displayed correctly, such a change will cause problems. Therefore, the pages themselves should be responsible for using new versions.

For example, a version number can be encoded in the URL in the ESI tag.

ESI Tags in the remaining HTML

- **line 9**: The ESI Include for `"/common/navbar"` ensures that each web page has the same navigation bar.
- **line 12:** Finally, `"/common/footer"` can contain scripts or a footer for the web page.

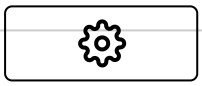
Result: HTML at the browser

Varnish collects these HTML snippets from the common service so that the browser receives the following HTML:

```
<html>
<head>
...
  <link rel="stylesheet"
   href="/common/css/bootstrap-3.3.7-dist/css/bootstrap.min.css" />
  <link rel="stylesheet"
   href="/common/css/bootstrap-3.3.7-dist/css/bootstrap-theme.min.css" />
</head>

<body>
  <div class="container">
    <a class="brand"
     href="https://github.com/ultraq/thymeleaf-layout-dialect">
     Thymeleaf - Layout </a>
    Mon Sep 18 2017 17:52:01 </div></div>
    ...
  </div>
  <script src="/common/css/bootstrap-3.3.7-dist/js/bootstrap.min.js" />
</body>
```

```
</html>
```



The ESI tags have thus been **replaced** by suitable HTML snippets.

ESI offers many other features for securing a system against the failure of a web server or for integrating HTML fragments only under certain conditions.

No tests without ESI infrastructure

A problem with the ESI approach is that individual services cannot be tested without an ESI infrastructure. At the very least, they do not display any pages with meaningful content, because the ESI tags would have to be interpreted for that. This works only if the HTTP requests are routed through Varnish. Therefore, suitable environments containing a Varnish must be provided for the development.

Effects on the application

The application is a normal Spring Boot web application without any dependencies on Spring Cloud or ESI. This shows that pure frontend integration leads to a very loose coupling and has little impact on the applications.

Common microservice

In the example provided, the *common* service is a very simple Go (https://golang.org/) application. It handles the three URLs `"/common/header"`, `"/common/navbar"`, and `"/common/footer"`. For these URLs, the Go code generates suitable HTML fragments.

Asset server

The Go code also contains a web server that provides static resources under `"/common/css/"` – the Bootstrap framework. In this way, the common microservice assumes the function of an asset server. Such a server offers CSS, images, or JavaScript code to applications. The ESI example shows an alternative for the integration of shared assets. In chapter 4 (https://www.educative.io/collection/page/10370001/5441945024331776/4860469534785536), a common asset project has ensured that all applications can use the same assets. In the example in this chapter, an asset server is used for this purpose.

The application displays the current time in the navigation bar. This shows that dynamic content can also be displayed with ESI includes.

QUIZ

1 What is the purpose of the common microservice?

A) To improve efficiency.

B) To increase load time by caching static parts of the web-page.

C) To ensure that all applications use the same assets and HTML snippets for the frontend.

Submit Answer

Question 1 of 2
0 attempted

Reset Quiz

In the next lesson, we'll look at some variations to this.

← Back

Varnish

Next →

Variations

Mark as Completed

Report an Issue