

Desenvolvimento de Software para Web



UFC - Universidade Federal do Ceará

André Meireles
andre@crateus.ufc.br

MPA - Template Engines

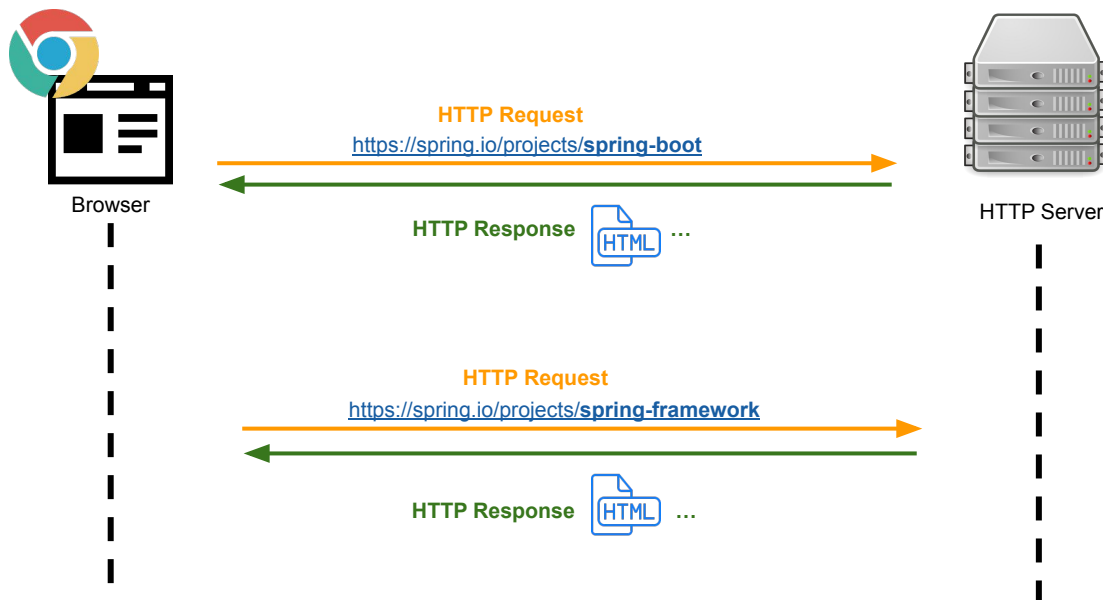


UFC - Universidade Federal do Ceará

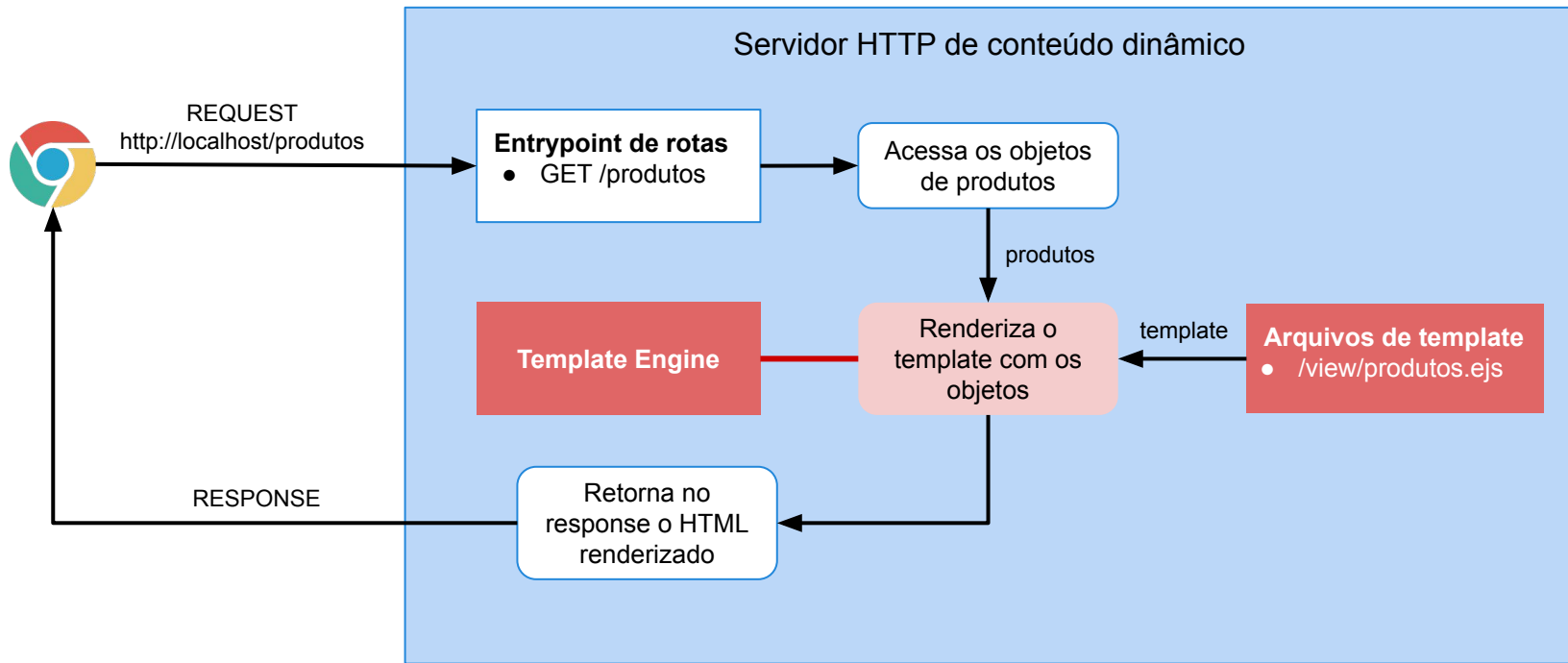
André Meireles
andre@crateus.ufc.br

Arquitetura Web Multi-page App (MPA)

- A navegação é baseada na troca de documentos HTML (site exemplo: <https://spring.io/>)
- O navegador reprocessa o DOM carregando automaticamente o documento recebido do servidor



MPA - Template Engine



Template Engines

- Facilita a construção de conteúdo HTML com conteúdo dinâmico
- Separa a camada de serviço e acesso aos dados da camada de visão
- Facilita o reuso de conteúdo HTML entre páginas
- Acrescenta recursos de programação à construção do HTML
- Existem vários no mercado e para várias linguagens:
 - Para Java Spring: JSP, JSF, Thymeleaf, Free Maker, Groove
 - Para Python Flask: Jinja
 - Para Node Express: Pug, Haml.js, EJS

EJS - Principais recursos

Acesse a documentação oficial: <https://ejs.co/>

- `<% %>`

'Scriptlet' tag para controle de fluxo sem saída

- `<%= %>`

Saída valores para dentro do template (HTML escaped)

- `<%- %>`

Saída valores UNSCAPED para dentro do template

- `<%- include ('arquivo.ejs') ; %>`

Injetar conteúdo de outro arquivo EJS

Node + EJS - Funções básicas

Para passar objetos para serem renderizados com o template:

No index.js:

```
app.get('/', (req, res) => {  
  res.render('home', {saudacao: 'Olá, seja bem vindo à LOJA'});  
});
```

No home.ejs:

```
<h1><%= saudacao %> </h1>
```

Para redirecionar o usuário para outra página:


```
app.post('/produto/produto-novo', (req, res) => {  
  let produto = req.body;  
  bdProduto.addProduto(produto);  
  res.redirect('/produto/produto-listar');  
});
```

EJS - Include

header.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Loja MPA</title>
  </head>
  <body>
    <h1 style="background-color: aquamarine;">Loja MPA -
    Produtos</h1>
```

produtos.ejs



```
<%- include('header') %>

<h2>Listar Produtos</h2>
<div style="background-color: green;">
```


Node - Modularização

Para criar um módulo em uma arquivos JS separado e utilizar os seus recursos (funções e objetos), você pode utilizar [exports e require](#) Veja:

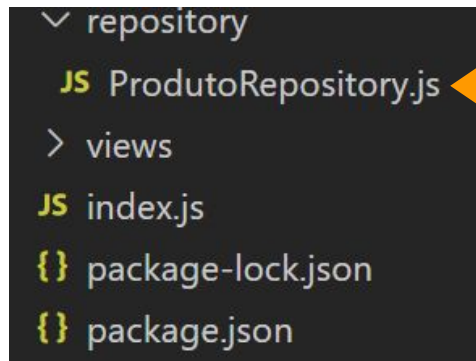
```
▼ repository
  JS ProdutoRepository.js
  > views
  JS index.js
  {} package-lock.json
  {} package.json
```

Suponha que você deseja armazenar os produtos em um array que controlá-lo separadamente no módulo ProdutoRepository

...e depois, você deseja utilizar (importar) o ProdutoRepository no index.js

Node - Modularização

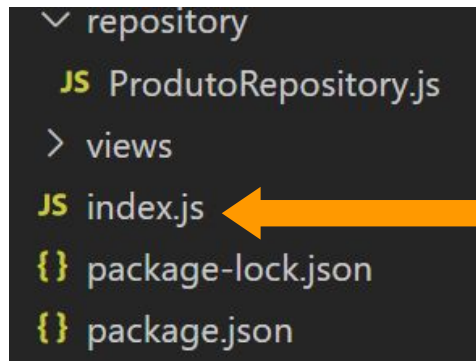
No módulo produto, você deve declarar normalmente os seus objetos e funções utilizar o `exports` para definir aqueles que você deseja expor através do `require`. Nesse caso, estamos expondo as funções `addProduto` e `getProdutos`.



```
function addProduto(produto) {  
  produto.id = produto_id+=1;  
  produtos.push(produto);  
}  
  
function getProdutos() {  
  return produtos;  
}  
  
exports.addProduto = addProduto;  
exports.getProdutos = getProdutos;
```

Node - Modularização

Agora, no arquivo que irá importar o módulo, nesse caso, o `index.js`, utilizamos o `require` e utilizamos as funções que foram expostas através do `exports`.



```
const bdProdutos =  
require('./repository/ProdutoRepository');  
...  
bdProdutos.addProduto(p);  
...  
console.log( bdProdutos.getProdutos() );
```

Links importantes

- <https://www.treinaweb.com.br/blog/utilizando-template-engine-ejs-com-node-js>
- EJS: <https://ejs.co/>
- Node JS - Modules: <https://nodejs.org/api/modules.html#exports>

Dúvidas??