Aula 19

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

Integração com Front-End

- JavaScript (fetch)

O fetch é uma versão mais poderosa e mais flexível do XMLHttpRequest, porém, diferente do XMLHttpRequest, ele não funciona como um objeto, e sim como um método.

Esse método executa uma promise, ou seja, uma promessa de retorno.

Integração com Front-End

- JavaScript (fetch)

Esse método recebe dois parâmetros, uma string que irá definir a url que será utilizada para realizar a requisição e um objeto que irá definir todas as configurações da requisição.

Integração com Front-End

- JavaScript (fetch)

```
function HttpRequest() {
    fetch("https://localhost:3000/api/Exemplo/", { method: "GET" });
}
```

Integração com Front-End

- JavaScript (fetch)

Como esse método retorna uma promise, temos dois métodos que utilizamos para executar quando a promise for finalizada:

- then
- catch

Integração com Front-End

- JavaScript (fetch (then))

O método then é executado quando a requisição é concluída com sucesso, e ele recebe uma função como parâmetro.

Integração com Front-End

- JavaScript (fetch (then))

```
function HttpRequest() {
    fetch("https://localhost:3000/api/Exemplo/", { method: "GET" })
        .then(result => {
            // Bloco de código do método then
        });
}
```

Integração com Front-End

- JavaScript (fetch (catch))

O método catch é executado quando algum erro acontece na execução da requisição, e ele recebe uma função como parâmetro.

Integração com Front-End

- JavaScript (fetch (catch))

```
function HttpRequest() {
    fetch("https://localhost:3000/api/Exemplo/", { method: "GET" })
        .then(result => {
            // Bloco de código do método then
        })
        .catch(error => {
            // Bloco de código do método catch
        });
}
```

Integração com Front-End

- JavaScript (fetch)

Quando executado com sucesso, o retorno do método é um objeto do tipo Response, o qual possui dois métodos bastante úteis para acessar as mensagens de retorno, ambos métodos que retornam promises:

- json
- text

Integração com Front-End

- JavaScript (fetch (json))

O método json vai retornar o corpo da resposta formatado como json, e como ele retorna uma promise, ele também possui os métodos then e catch.

Integração com Front-End

- JavaScript (fetch (json))

```
function HttpRequest() {
    fetch("https://localhost:3000/api/Exemplo/", { method: "GET" })
        .then(result => {
            result.json().then(json => {
                console.log(json);
            });
        });
}
```

Integração com Front-End

- JavaScript (fetch (text))

O método text vai retornar o corpo da resposta na forma de uma string, e como ele retorna uma promise, ele também possui os métodos then e catch.

Integração com Front-End

- JavaScript (fetch (text))

```
function HttpRequest() {
    fetch("https://localhost:3000/api/Exemplo/", { method: "GET" })
        .then(result => {
            result.text().then(text => {
                console.log(text);
            });
        });
}
```

Integração com Front-End

- JavaScript (jQuery)

O jQuery é uma biblioteca JavaScript que fornece diversos métodos diferentes para interagir com o HTML e executar processos específicos de forma bastante simplificada.

É esse carinha que eu quero que vocês usem nos primeiros exercícios de integração.

Integração com Front-End

- JavaScript (jQuery)

Para utilizarmos o jQuery, precisamos adicionar os arquivos da biblioteca no HTML da nossa aplicação.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

Integração com Front-End

- JavaScript (jQuery)

```
<!DOCTYPE html>
<html>
    <body>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    </body>
</html>
```

Integração com Front-End

- JavaScript (jQuery)

Uma vez importado os arquivos da biblioteca, para executarmos a nossa requisição basta executarmos um único método chamado ajax que vai receber um único objeto como parâmetro que irá definir todas as informações necessárias para a requisição ser executada.

Esse método, assim como todos os métodos do jQuery possuem um prefixo de cifrão-ponto ($.).

Integração com Front-End

- JavaScript (jQuery)

```
function HttpRequest() {
    $.ajax({ });
}
```

Integração com Front-End

- JavaScript (jQuery)

O objeto que o método recebe como parâmetro possui alguns atributos que irão definir as configurações da requisição:

- headers
- url
- type
- data
- success
- error

Integração com Front-End

- JavaScript (jQuery (headers))

O atributo headers recebe um objeto, que vai definir o cabeçalho da requisição.

Quando a requisição executa um POST, é obrigatório informar esse atributo e passar algumas configurações de cabeçalho de JSON.

Integração com Front-End

- JavaScript (jQuery (headers))

```
function HttpRequest() {
    $.ajax({
        headers: {
            "Accept": "application/json",
            "Content-Type": "application/json"
        }
    });
}
```

Integração com Front-End

- JavaScript (jQuery (url))

O atributo url recebe uma string, que vai definir a url que será utilizada para realizar a requisição.

Integração com Front-End

- JavaScript (jQuery (url))

```
function HttpRequest() {
    $.ajax({
        headers: {
            "Accept": "application/json",
            "Content-Type": "application/json"
        },
        url: "https://localhost:3000/api/Exemplo/"
    });
}
```

Integração com Front-End

- JavaScript (jQuery (type))

O atributo type recebe uma string, que vai definir o método HTTP a ser executado pela requisição.

Integração com Front-End

- JavaScript (jQuery (type))

```
function HttpRequest() {
    $.ajax({
        headers: {
            "Accept": "application/json",
            "Content-Type": "application/json"
        },
        url: "https://localhost:3000/api/Exemplo/",
        type: "POST"
    });
}
```

Integração com Front-End

- JavaScript (jQuery (data))

O atributo data recebe um objeto, que vai definir o corpo da requisição.

Para passarmos um JSON como corpo da requisição, é preciso fazer uma conversão de objeto para um JSON, usando JSON.stringify.

Integração com Front-End

- JavaScript (jQuery (data))

```
function HttpRequest() {
    let objetoExemplo = {
        id: 1,
        nome: "Exemplo"
    };

    $.ajax({
        headers: {
            "Accept": "application/json",
            "Content-Type": "application/json"
        },
        url: "https://localhost:3000/api/Exemplo/",
        type: "POST",
        data: JSON.stringify(objetoExemplo)
    });
}
```

Integração com Front-End

- JavaScript (jQuery (success))

O atributo success recebe um método, que vai definir o que vai ser executado se a requisição for executada com sucesso.

Integração com Front-End

- JavaScript (jQuery (success))

```
function HttpRequest() {
    let objetoExemplo = {
        id: 1,
        nome: "Exemplo"
    };

    $.ajax({
        headers: {
            "Accept": "application/json",
            "Content-Type": "application/json"
        },
        url: "https://localhost:3000/api/Exemplo/",
        type: "POST",
        data: JSON.stringify(objetoExemplo),
        success: function(result) {
            // Bloco de código do método success
        }
    });
}
```

Integração com Front-End

- JavaScript (jQuery (error))

O atributo error recebe um método, que vai definir o que vai ser executado se algum erro ocorrer na execução da requisição.

Integração com Front-End

- JavaScript (jQuery (error))

```
function HttpRequest() {
    let objetoExemplo = {
        id: 1,
        nome: "Exemplo"
    };

    $.ajax({
        headers: {
            "Accept": "application/json",
            "Content-Type": "application/json"
        },
        url: "https://localhost:3000/api/Exemplo/",
        type: "POST",
        data: JSON.stringify(objetoExemplo),
        success: function(result) {
            // Bloco de código do método success
        },
        error: function(error) {
            // Bloco de código do método error
        }
    });
}
```

Exercício

Exercício

No exercício da aula 16, nós fizemos uma API de calculadora.

Então agora vamos fazer o front-end (HTML e JavaScript) para executar a nossa API de calculadora.

Pode tanto ser uma única página para tudo, com uma página para cada tipo de cálculo.