Aula 18

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

Integração com Front-End

Integração com Front-End

Uma parte muito importante do desenvolvimento de aplicações web, é integrar o back-end (C#, Java, etc) com o front-end (HTML, CSS, JavaScript, etc).

Ou seja, fazer a parte front-end acessar os processos do back-end.

Integração com Front-End

Para isso, primeiramente vamos precisar fazer duas coisas:

- ➔ Criar algumas páginas HTML para representar o front-end de um sistema.
- ➔ Criar uma API simples para representar o back-end de um sistema.

Uma vez feito isso, vamos poder nos aprofundar nas formas que existem de realizarmos essa integração.

Integração com Front-End

- JavaScript

Vamos começar do básico, e fazer a nossa integração através do JavaScript, que possui ótimos módulos e métodos para fazer requisições HTTP, tando módulos e métodos nativos, como importados de pacotes.

Integração com Front-End

- JavaScript

Nessas nossas primeiras aulas, vamos aprender três maneira diferentes de fazer o nosso front-end se comunicar com o back-end:

- XMLHttpRequest
- fetch
- jQuery

Integração com Front-End

- JavaScript (XMLHttpRequest)

O XMLHttpRequest é uma das maneiras mais tradicionais de realizar requisições assíncronas no JavaScript.

Ele é um objeto que fornece de maneira simples funcionalidades ao front-end para transferir dados com um back-end.

Integração com Front-End

- JavaScript (XMLHttpRequest)

Foi originalmente projetado pela Microsoft e adotado pela Mozilla, Apple e Google.

Apesar do nome, o XMLHttpRequest pode ser usado para recuperar qualquer tipo de dados, e não apenas XML, suportando também, protocolos diferentes de HTTP, com o FTP.

Integração com Front-End

- JavaScript (XMLHttpRequest)

Como se trata de um objeto, para podermos acessar todos os métodos e atributos necessários para realizar uma requisição, precisamos instanciar esse objeto em uma variável ou constante.

Integração com Front-End

- JavaScript (XMLHttpRequest)

```
function HttpRequest() {
    const request = new XMLHttpRequest();
}
```

Integração com Front-End

- JavaScript (XMLHttpRequest)

Uma vez instanciado o nosso objeto, temos vários métodos que utilizamos para construir a nossa requisição e para executá-la. Vamos no momento nos preocupar com 2 deles:

- open
- send

Integração com Front-End

- JavaScript (XMLHttpRequest (open))

O método open vai inicializar uma nova requisição.

Ele recebe dois parâmetros obrigatórios, ambos do tipo string, que irão definir o tipo do método HTTP que está sendo executado e a url que será utilizada para realizar a requisição.

Integração com Front-End

- JavaScript (XMLHttpRequest (open))

```
function HttpRequest() {
    const request = new XMLHttpRequest();
    request.open("GET", "https://localhost:3000/api/Exemplo/");
}
```

Integração com Front-End

- JavaScript (XMLHttpRequest (open))

O método open também pode receber um terceiro parâmetro do tipo booleano, o qual é opcional, para indicar se a requisição será realizada de forma assíncrona ou não.

Por padrão esse valor é true, ou seja, a requisição é executada de forma assíncrona por padrão.

Integração com Front-End

- JavaScript (XMLHttpRequest (open))

```
function HttpRequest() {
    const request = new XMLHttpRequest();
    request.open("GET", "https://localhost:3000/api/Exemplo/", false);
}
```

Integração com Front-End

- JavaScript (XMLHttpRequest (send))

O método send vai enviar requisição para o servidor, para ser executada.

Ele recebe um parâmetro opcional, que permite especificar qual é o corpo da requisição, e ele deve ser a última coisa a ser executada.

Integração com Front-End

- JavaScript (XMLHttpRequest (send))

```
function HttpRequest() {
    const request = new XMLHttpRequest();
    request.open("GET", "https://localhost:3000/api/Exemplo/");
    request.send();
}
```

Integração com Front-End

- JavaScript (XMLHttpRequest)

Além de métodos, o XMLHttpRequest possui uma estrutura de eventos, sendo que cada um deles é executado quando um determinado processo é realizado. Vamos no momento nos preocupar com 2 deles:

- load
- error

Integração com Front-End

- JavaScript (XMLHttpRequest (load))

O evento load é executado quando a requisição é concluída com sucesso.

Para definirmos o evento load, fazemos uso de um handler chamado onload que recebe uma função.

Integração com Front-End

- JavaScript (XMLHttpRequest (load))

```
function HttpRequest() {
    const request = new XMLHttpRequest();
    request.open("GET", "https://localhost:3000/api/Exemplo/");
    request.onload = function() {
        // Bloco de código do evento load
    };
    request.send();
}
```

Integração com Front-End

- JavaScript (XMLHttpRequest (error))

O evento error é executado sempre que algum erro acontece na execução da requisição.

Para definirmos o evento error, fazemos uso de um handler chamado onerror que recebe uma função.

Integração com Front-End

- JavaScript (XMLHttpRequest (error))

```
function HttpRequest() {
    const request = new XMLHttpRequest();
    request.open("GET", "https://localhost:3000/api/Exemplo/");
    request.onload = function() {
        // Bloco de código do evento load
    };
    request.onerror = function() {
        // Bloco de código do evento error
    };
    request.send();
}
```

Integração com Front-End

- JavaScript (XMLHttpRequest (responseText))

Além de métodos e eventos, o XMLHttpRequest possui atributos. Os quais entre eles o único que nos interessa no momento é o responseText, que nos permite acessar a mensagem de resposta da requisição.

Integração com Front-End

- JavaScript (XMLHttpRequest (responseText))

```
function HttpRequest() {
    const request = new XMLHttpRequest();
    request.open("GET", "https://localhost:3000/api/Exemplo/");
    request.onload = function() {
        console.log(this.responseText);
    };
    request.send();
}
```