



## Ah se meu HttpService fizesse mais coisa...

Durante o treinamento criamos a classe `HttpService` para centralizar toda infraestrutura necessária para realizarmos requisições Ajax, inclusive utilizamos o padrão `Promise` para tornar nosso código mais fácil de manter e legível. Contudo, nosso serviço possui apenas o método `get` responsável em ler os dados do servidor. Que tal encapsularmos nesse serviço toda complexidade para realizarmos requisições do tipo `POST`?

Vamos alterar `aluraframe/client/js/app/services/HttpService.js` e adicionar o método `POST` :

```
class HttpService {

  get(url) {

    return new Promise((resolve, reject) => {

      let xhr = new XMLHttpRequest();

      xhr.open('GET', url);

      xhr.onreadystatechange = () => {

        if (xhr.readyState == 4) {

          if (xhr.status == 200) {

            resolve(JSON.parse(xhr.responseText));
          } else {

            reject(xhr.responseText);
          }
        }
      };

      xhr.send();

    });
  }

  post(url, dado) {

    return new Promise((resolve, reject) => {

      let xhr = new XMLHttpRequest();
      xhr.open("POST", url, true);
      xhr.setRequestHeader("Content-Type", "application/json");
```

```
    });  
  
  }  
}
```

Nosso novo método recebe uma URL e o dado que desejamos enviar. Criamos uma instância de `XMLHttpRequest` e usamos o verbo `POST` no já conhecido método `open` de `XMLHttpRequest`. Contudo, quando estamos enviando um dado para o servidor, precisamos dizer **qual tipo de conteúdo** estamos enviando. A ideia é enviarmos um dado no tipo (formato) `JSON`. É por isso que precisamos adicionar ao cabeçalho da requisição que será realizada a informação `Content-Type` com o valor `application/json`.

Os próximos passos vocês já conhecem. Precisamos saber quando a requisição foi realizada e para isso usamos o `onreadystatechange`:

```
class HttpService {  
  
  get(url) {  
  
    // código anterior omitido  
  
  });  
}  
  
  post(url, dado) {  
  
    return new Promise((resolve, reject) => {  
  
      let xhr = new XMLHttpRequest();  
      xhr.open("POST", url, true);  
      xhr.setRequestHeader("Content-type", "application/json");  
      xhr.onreadystatechange = () => {  
  
        if (xhr.readyState == 4) {  
  
          if (xhr.status == 200) {  
  
            resolve(JSON.parse(xhr.responseText));  
          } else {  
  
            reject(xhr.responseText);  
          }  
        }  
      };  
      // falta enviar!!!!!!  
    });  
  }  
}
```

Veja que usamos `JSON.parse` para converter o JSON (String) retornado pelo servidor em um objeto JavaScript. Mas perceba que ainda falta a chamada do método `xhr.send` que recebe com parâmetro os dados que queremos enviar. Mas atenção: como HTTP é um protocolo textual, não podemos enviar um objeto JavaScript diretamente, precisamos convertê-lo para uma string no formato JSON. Para isso, usamos `JSON.stringify`:

```
class HttpService {

  get(url) {

    // código omitido
  }

  post(url, dado) {

    return new Promise((resolve, reject) => {

      let xhr = new XMLHttpRequest();
      xhr.open("POST", url, true);
      xhr.setRequestHeader("Content-type", "application/json");
      xhr.onreadystatechange = () => {

        if (xhr.readyState == 4) {

          if (xhr.status == 200) {

            resolve(JSON.parse(xhr.responseText));
          } else {

            reject(xhr.responseText);
          }
        }
      };
      xhr.send(JSON.stringify(dado)); // usando JSON.stringify para converter objeto em uma string
    });
  }
}
```

Excelente, o método POST está pronto. Agora, se você fez os exercícios do capítulo anterior, deve lembrar que já fizemos um exercício para enviar dados com o método POST para nosso servidor e criamos a página `post.html`. Se você não lembra, é porque pulou exercícios e deixou de aprender várias coisas extras durante o treinamento. Enfim, partindo do pressuposto que você fez o exercício do `post.html`, vamos alterá-lo para fazer uso do nosso serviço e seu método `post`.

Primeiro, vamos ver o código antes da nossa modificação:

```
<!-- aluraframe/client/post.html -->
<!-- aluraframe/client/post.html -->
<html>
<head>
```

```
<meta charset="UTF-8">
<title>Negociações</title>
<link rel="stylesheet" href="css/bootstrap.css">
<link rel="stylesheet" href="css/bootstrap-theme.css">
</head>

<body class="container">
  <form class="form">

    <div class="form-group">
      <label for="data">Data</label>
      <input type="date" id="data" class="form-control" required autofocus/>
    </div>

    <div class="form-group">
      <label for="quantidade">Quantidade</label>
      <input type="number" min="1" step="1" id="quantidade" class="form-control" value="1" requir
    </div>

    <div class="form-group">
      <label for="valor">Valor</label>
      <input id="valor" type="number" class="form-control" min="0.01" step="0.01" value="0.0" re
    </div>

    <button class="btn btn-primary" type="submit" onclick="sendPost(event)">Enviar dados para servi
  </form>

<script>
  function sendPost(event) {

    event.preventDefault();

    console.log("Enviando post");

    let $ = document.querySelector.bind(document);
    inputData = $('#data');
    inputQuantidade = $('#quantidade');
    inputValor = $('#valor');

    let negociacao = {
      data: inputData.value,
      quantidade: inputQuantidade.value,
      valor: inputValor.value
    };

    let xhr = new XMLHttpRequest();
    xhr.open("POST", "/negociacoes", true);
    xhr.setRequestHeader("Content-type", "application/json");

    xhr.onreadystatechange = () => {

      if (xhr.readyState == 4) {

        if (xhr.status == 200) {
```

```

        inputData.value = '';
        inputQuantidade.value = 1;
        inputValor.value = 0.0;
        inputData.focus();
        alert('Negociação enviada com sucesso');
    } else {
        alert(`Não foi possível enviar a negociação: ${xhr.responseText}`);
    }
}
}
xhr.send(JSON.stringify(negociacao));

```

```

}
</script>

```

```

</body>

```

```

</html>

```

Agora, importando o script `aluraframe/client/js/app/services/HttpService.js` :

```

<!-- aluraframe/client/post.html -->
<html>
<head>
  <meta charset="UTF-8">
  <title>Negociações</title>
  <link rel="stylesheet" href="css/bootstrap.css">
  <link rel="stylesheet" href="css/bootstrap-theme.css">
</head>

<body class="container">
  <form class="form">

    <div class="form-group">
      <label for="data">Data</label>
      <input type="date" id="data" class="form-control" required autofocus/>
    </div>

    <div class="form-group">
      <label for="quantidade">Quantidade</label>
      <input type="number" min="1" step="1" id="quantidade" class="form-control" value="1" requir
    </div>

    <div class="form-group">
      <label for="valor">Valor</label>
      <input id="valor" type="number" class="form-control" min="0.01" step="0.01" value="0.0" re
    </div>

    <button class="btn btn-primary" type="submit" onclick="sendPost(event)">Enviar dados para servi
  </form>

  <script src="js/app/services/HttpService.js"></script>
<script>
  function sendPost(event) {

```

```
event.preventDefault();

console.log("Enviando post");

let $ = document.querySelector.bind(document);
inputData = $('#data');
inputQuantidade = $('#quantidade');
inputValor = $('#valor');

let negociacao = {
  data: inputData.value,
  quantidade: inputQuantidade.value,
  valor: inputValor.value
};

// usando nosso serviço. Veja que nem guardei em uma variável
new HttpService()
  .post('/negociacoes', negociacao)
  .then(() => {
    inputData.value = '';
    inputQuantidade.value = 1;
    inputValor.value = 0.0;
    inputData.focus();
    alert('Negociação enviada com sucesso');
  })
  .catch(erro => alert(`Não foi possível enviar a negociação: ${erro}`));
}
</script>
</body>
</html>
```

Veja que agora não precisamos lidar com detalhes de `XMLHttpRequest` quando formos realizar requisições do tipo POST!