04

SystemJs

Transcrição

Faltou adicionarmos uma importação ao arquivo `NegociacaoService.js` :

```
import {HttpService} from './HttpService';
import {ConnectionFactory} from './ConnectionFactory';
import {NegociacaoDao} from '../dao/NegociacaoDao';
import {Negociacao} from '../models/Negociacao';

export class NegociacaoService {

      constructor(){

          this._http = new HttpService()
      }
 //...
```

Agora, fizemos todas as importações necessárias. Mas por que o código não funciona como está? Porque a especificação ES2015 define o `import` e `export` , além de que cada script é um módulo independente. No entanto, não definimos como estes módulos devem ser carregados no navegador. Não existe um consenso... Precisamos que os scripts sejam carregados em uma determinada ordem no seu sistema, definindo apenas o primeiro. A partir deste, serão carregados os demais. O responsável pelo processo é *loader*, porém, não existe um padrão nos navegadores. Para resolver a questão, teremos que escolher uma biblioteca de terceiro que atue como um loader de script. Uma biblioteca muito famosa é System JS. Nós iremos baixá-lo pelo NPM do Node.JS, e iremos colocá-lo na pasta `node_modules` .

Em seguida, vamos parar o Terminal. Dentro da pasta `client` , instalaremos o System JS.

```
npm install systemjs@0.19.31 --save
```

O SystemJS é um script que precisa ser carregado com a aplicação. O processo de instalação é bem rápido. Então, dentro da pasta `node_modules` encontraremos o `systemjs` . Depois, importaremos o script no `index.html` , o arquivo ficará logo no início:

```
<div id="negociacoesView"></div>
<script src="node_modules/systemjs/dist/system.js"></script>
<script src="js/app/polyfill/fetch.js"></script>
<script src="js/app/models/Negociacao.js"></script>
<script src="js/app/models/ListaNegociacoes.js"></script>
<script src="js/app/models/Mensagem.js"></script>
<script src="js/app/controllers/NegociacaoController.js"></script>
<script src="js/app/helpers/DateHelper.js"></script>
<script src="js/app/views/View.js"></script>
<script src="js/app/views/NegociacoesView.js"></script>
<script src="js/app/views/MensagemView.js"></script>
<script src="js/app/services/ProxyFactory.js"></script>
<script src="js/app/helpers/Bind.js"></script>
<script src="js/app/services/NegociacaoService.js"></script>
```

```
<script src="js/app/services/HttpService.js"></script>
<script src="js/app/services/ConnectionFactory.js"></script>
<script src="js/app/dao/NegociacaoDao.js"></script>
<script src="js/app/dao/datex.js"></script>
<script>
    var negociacaoController = new NegociacaoController();
</script>
```

Importamos o loader. Porém, ao adicionarmos o responsável pela importação dos scripts, não precisaremos mais importar todos os demais:

```
<div id="negociacoesView"></div>

<script src="node_modules/systemjs/dist/system.js"></script>
<script>
    var negociacaoController = new NegociacaoController();
</script>
```

Nós vamos especificar para o `systemjs` qual será o primeiro módulo carregado, e automaticamente, ele baixará os demais scripts na ordem de dependência. Ou seja, o desenvolvedor não precisará mais se preocupar com a ordem. Porém, falta indicar o primeiro módulo. Faremos a seguinte configuração:

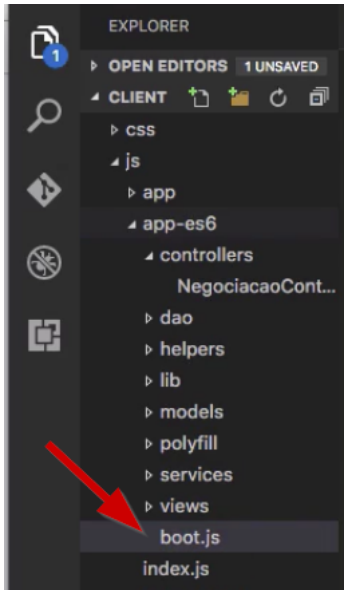
```
<div id="negociacoesView"></div>

<script src="node_modules/systemjs/dist/system.js"></script>
<script>
    System.defaultJSExtensions = true;
    System.import('js/app/boot').catch(function(err){
        console.error(err);
    });
</script>
<script>
    var negociacaoController = new NegociacaoController();
</script>
```

O `systemjs` tem a variável global `System`. Quando definimos `true` para `defaultJSExtensions`, isto nos permite omitir as extensões JS dos imports. Como em `NegociacaoController.js`, que fizemos as importações sem adicionar a extensão `js`:

```
import {ListaNegociacoes} from '../modelo/ListaNegociacoes';
import {Mensagem} from '../modelo/Mensagem';
import {NegociacaoView} from '../view/NegociacaoView';
import {MensagemView} from '../view/MensagemView';
import {NegociacaoService} from '../services/NegociacaoService';
import {DateHelper} from '../helpers/DateHelper';
import {Bind} from '../helpers/Bind';
import {Negociacao} from '../models/Negociacao';
```

De volta ao `index.html`, especificamos com `System` qual será o primeiro módulo carregado: no caso, será o `boot.js` que criaremos a seguir. Dentro da pasta `app-es6`:



Este será o primeiro arquivo a ser carregado. Também não fará mais sentido termos o `NegociacaoController` no escopo global, por isso, iremos removê-lo.

```
<div id="negociacoesView"></div>

<script src="node_modules/systemjs/dist/system.js"></script>
<script>
    System.defaultJSExtensions = true;
    System.import('js/app/boot').catch(function(err){
        console.error(err);
    });
</script>
```

Conseguimos apagar diversos scripts que precisam ser carregados manualmente.