04

Quero gravar em uma Object Store, mas onde está a transação?

Transcrição

Como queremos gravar `negociacoes`, vamos importar o `script` na nossa página de teste `aprendendo_indexeddb.html`:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
</head>
<body>

    <script src="js/app/models/Negociacao.js"></script>
//...
```

Importamos o modelo para criar a instância de `negociacoes`. Depois, abaixo do `onerror`, criaremos a `function` que receberá o nome `adiciona()`.

```
function adiciona() {

}

adiciona();
```

Quando a função for chamada, ela deverá ser capaz de gravar uma instância de negociação no `IndexedDB`.

Em seguida, vamos colocar a variável `minhaConnection` dentro do `let`, porque ela está dentro do bloco de `openRequest`.

```
openRequest.onupgradeneeded = e =>  {

        console.log('Criando ou atualizando o banco');

        let minhaConnection = e.target.result;
        minhaConnection.createObjectStore('negociacoes');
};
```

Para conseguirmos gravar, precisaremos de uma transação (`transaction`) dentro da função `adiciona()`.

```
 function adiciona() {

     let transaction = connection.transaction(['negociacoes'],'readwrite');

      let store = transaction.objectStore('negociacoes');

 }
```

Especificamos qual será a Object Store colocando o `negociacoes` no array. Com o nosso segundo parâmetro, podemos "ler e escrever" (`readwrite`) - se quiséssemos apenas "ler", utilizaríamos o `readonly` . Nesta transação, teremos acesso a uma Object Store transacional. O código parece redundante, mas é assim que o IndexedDB funciona...

Agora por meio da `store` , conseguiremos fazer transações de persistência(gravar, incluir, alterar e listar). Depois, iremos gerar a variável `negociacao` :

```
function adiciona() {

        let transaction = connection.transaction(['negociacoes'],'readwrite');

        let store = transaction.objectStore('negociacoes');

        let negociacao = new Negociacao(new Date(), 200, 1);

        let request = store.add(negociacao);
}
```

Após termos uma negociação, iremos gravar na Store, usando o `store.add()` . Com isto, faremos uma requisição pedindo que a `store` realize a gravação, para sabermos se a ação foi executada, ela foi colocada em um `request` .

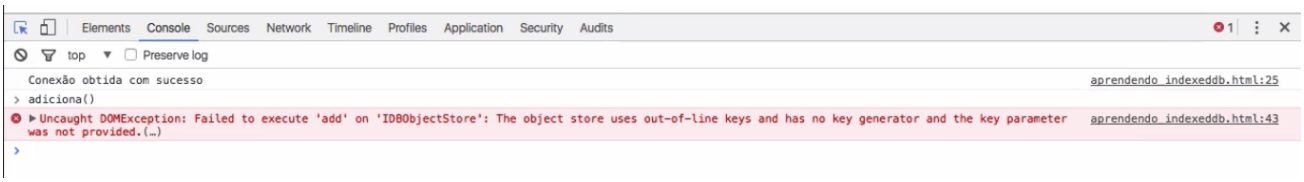
Em seguida, adicionaremos ao `request` o `onsuccess` e `onerror` , para os casos de que a requisição tenha sucesso ou não.

```
let request = store.add(negociacao);

request.onsuccess = e => {

  console.log('Negociação incluída com sucesso');
};

request.onerror = e => {

  console.log('Não foi possível incluir a negociação');
};
```

E não chamaremos mais o `adiciona()` depois do `request.onerror` . Iremos chamá-lo manualmente no navegador.

```
> adiciona()
```

E será exibida uma mensagem de erro que a Object Store usa chaves fora da linha.



Esta mensagem significa que quando gravamos numa Object Store, precisamos definir um `id` - que deve ser único ou autonumerado. Para isto, no `onupgradeneeded` , vamos alterar o banco. Também teremos que destruir a Object Store criada anteriormente. Começaremos alterando o valor da versão.

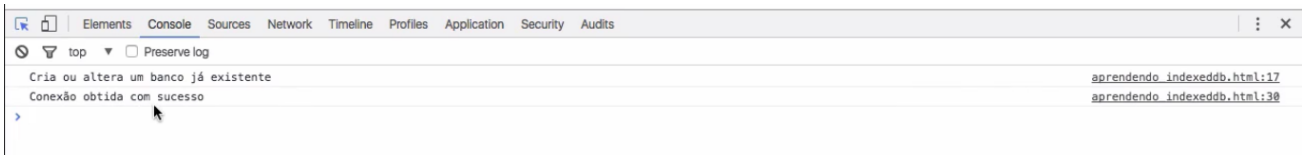
```
var openRequest = window.indexedDB.open('aluraframe', 3);

openRequest.onupgradeneeded = e => {

    console.log('Cria ou altera um banco já existente');

    let minhaConnection = e.target.result;

    if(minhaConnection.objectStoreNames.contains('negociacoes')) {
      minhaConnection.deleteObjectStore('negociacoes');
    }
    minhaConnection.createObjectStore('negociacoes');
};
```

Sempre que chamarmos o `onupgradeneeded`, ele destruirá a Object Store caso já exista uma e outra será criada logo abaixo. Dentro do `createObjectStore()`, adicionaremos o `autoIncrement`.

```
openRequest.onupgradeneeded = e => {
    // CÓDIGO OMITIDO
    minhaConnection.createObjectStore('negociacoes', { autoIncrement: true });
```

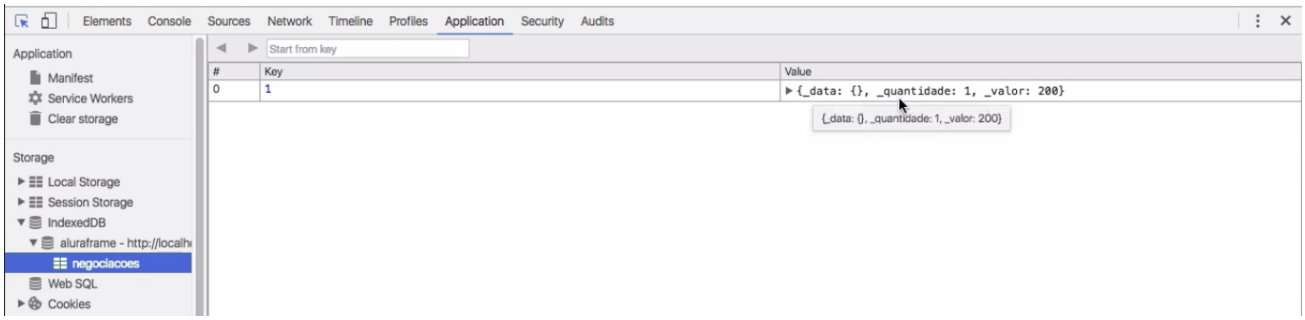
Como esta é versão `3` do banco, ele executará o `onupgradeneeded` e o nosso `if` testará se a Object Store existe - se já existir uma, ela será detonada e uma nova será criada com o `autoIncrement`. Lembrando que esta não conseguirá alterar uma Object Store já criada.

Vamos executar o código e ver o que acontece:



As mensagens são exibidas corretamente, e se chamarmos o método `adiciona()` no navegador, receberemos a confirmação de que a negociação foi adicionada com sucesso.

Na aba "Application", veremos que o `negociacoes` foi incluído na Object Store e os dados estão sendo salvos corretamente:



Conseguimos gravar a nossa primeira `negociacao`, mais adiante, seremos capazes de listar os dados.

Update: Atualmente, com as versões mais recentes do Chrome, já é possível visualizar o campo `_date` no console do IndexedDB.