Documentação do arquivo Vehicles.cls

Introdução

O arquivo `Vehicles.cls` é uma classe Apex que gerencia operações relacionadas a veículos em um sistema Salesforce. Ele fornece métodos para recuperar uma lista de veículos disponíveis para venda e para atualizar o status de um veículo como vendido.

Descrição

A classe `Vehicles` contém dois métodos principais:

1. `getVehicles`: Recupera uma lista de veículos que ainda não foram vendidos.
2. `persistPurchase`: Atualiza o status de um veículo específico para indicar que ele foi vendido.

Esses métodos são expostos para uso em componentes Lightning (Aura ou LWC) por meio da anotação `@auraEnabled`.

Estrutura

A classe está estruturada da seguinte forma:

- Declaração da classe `Vehicles` com o modificador `public with sharing`.
- Dois métodos estáticos:
  - `getVehicles`: Método cacheável que retorna uma lista de veículos disponíveis.
  - `persistPurchase`: Método que atualiza o status de um veículo para vendido.

Dependências

A classe depende do seguinte:

- Objeto personalizado `Vehicles__c`, que contém os campos:
  - `Id`
  - `Name`
  - `Year__c`
  - `Preview__c`
  - `Brand__c`
  - `Sold__c`

Imports

Não há imports explícitos, pois a classe utiliza recursos nativos do Salesforce Apex.

Variáveis

Variáveis Locais

- `lVehicles`: Lista de veículos do tipo `Vehicles__c` usada no método `getVehicles`.
- `lVehichles`: Lista de veículos do tipo `Vehicles__c` usada no método `persistPurchase`.

Métodos

`getVehicles`

Descrição:
Este método retorna uma lista de veículos que ainda não foram vendidos. Ele utiliza uma consulta SOQL para buscar os registros no objeto `Vehicles__c` onde o campo `Sold__c` é `false`.

Anotação:

- `@auraEnabled(cacheable = true)`: Permite que o método seja chamado a partir de componentes Lightning e habilita o cache para melhorar o desempenho.

Retorno:

- Uma lista de registros do tipo `Vehicles__c`.

Código:

```
@auraEnabled(cacheable = true)
public static List<Vehicles__c> getVehicles(){
    List<Vehicles__c> lVehicles = new List<Vehicles__c>();
    lVehicles = [SELECT Id, Name, Year__c, Preview__c, Brand__c, Sold__c FROM Vehicles__c WHERE Sold__c = false];
    return lVehicles;
}
```

`persistPurchase`

Descrição:
Este método atualiza o status de um veículo específico para indicar que ele foi vendido. Ele recebe o ID do veículo como parâmetro, busca o registro correspondente e altera o campo `Sold__c` para `true`.

Anotação:

- `@auraEnabled`: Permite que o método seja chamado a partir de componentes Lightning.

Parâmetros:

- `carId` (Id): O ID do veículo que será atualizado.

Tratamento de Exceções:

- Caso ocorra algum erro durante a execução, uma exceção do tipo `AuraHandledException` é lançada com a mensagem de erro.

Código:

```
@auraEnabled
public static void persistPurchase(Id carId){
    try {
        List<Vehicles__c> lVehichles = [SELECT id, Name, Year__c, Preview__c, Brand__c, Sold__c FROM Vehicles__c WHERE Id =: carId];
        lVehichles[0].Sold__c = true;
        update lVehichles;
    } catch (Exception e) {
        throw new AuraHandledException(e.getMessage());
    }
}
```

Exemplo

Exemplo de uso do método `getVehicles`:

```
import { LightningElement, wire } from 'lwc';
import getVehicles from '@salesforce/apex/Vehicles.getVehicles';

export default class VehicleList extends LightningElement {
    @wire(getVehicles)
    vehicles;
}
```

Exemplo de uso do método `persistPurchase`:

```
import { LightningElement } from 'lwc';
import persistPurchase from '@salesforce/apex/Vehicles.persistPurchase';

export default class PurchaseVehicle extends LightningElement {
    handlePurchase(vehicleId) {
        persistPurchase({ carId: vehicleId })
            .then(() => {
                console.log('Veículo comprado com sucesso!');
            })
            .catch(error => {
                console.error('Erro ao comprar o veículo:', error);
            });
    }
}
```

Diagrama de Dependência

O diagrama abaixo ilustra as dependências entre os métodos e o objeto `Vehicles__c`.

```
classDiagram
    class Vehicles {
        +List~Vehicles__c~ getVehicles()
        +void persistPurchase(Id carId)
    }
    class Vehicles__c {
        +Id
        +Name
        +Year__c
        +Preview__c
        +Brand__c
        +Sold__c
    }
    Vehicles --> Vehicles__c
```

Notas

- A anotação `@auraEnabled(cacheable = true)` no método `getVehicles` permite que os dados sejam armazenados em cache, melhorando o desempenho em componentes Lightning.
- O método `persistPurchase` utiliza tratamento de exceções para garantir que erros sejam gerenciados de forma controlada.

Vulnerabilidades

- **Consulta SOQL em `persistPurchase`**: O método assume que o ID fornecido sempre retornará um registro. Caso contrário, pode ocorrer uma exceção de índice ao acessar `lVehichles[0]`. Uma verificação adicional deve ser implementada para garantir que a lista não esteja vazia antes de acessar o índice.
- **Falta de validação de entrada**: O método `persistPurchase` não valida o ID fornecido. Isso pode levar a problemas de segurança ou erros inesperados. É recomendável validar o ID antes de usá-lo na consulta SOQL.