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 é composta por:

- Métodos estáticos para manipulação de registros do objeto customizado `Vehicles__c`.
- Uso de SOQL para consulta de dados no Salesforce.
- Tratamento de exceções para garantir a robustez do método `persistPurchase`.

Dependências

A classe depende do seguinte:

- Objeto customizado `Vehicles__c` com os campos:
  - `Id`
  - `Name`
  - `Year__c`
  - `Preview__c`
  - `Brand__c`
  - `Sold__c`
- Framework Lightning para expor métodos com `@auraEnabled`.

Imports

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

Variáveis

- `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

1. `getVehicles`

Descrição:
Este método retorna uma lista de veículos que ainda não foram vendidos (`Sold__c = false`). Ele é marcado como `@auraEnabled` e `cacheable = true`, permitindo que seja usado em componentes Lightning com cache para melhorar o desempenho.

Assinatura:

```
@auraEnabled(cacheable = true)
public static List<Vehicles__c> getVehicles()
```

Funcionamento:

- Executa uma consulta SOQL para buscar veículos com o campo `Sold__c` definido como `false`.
- Retorna a lista de veículos.

Exemplo de uso:

```
// Chamando o método em um componente Lightning
let action = component.get("c.getVehicles");
action.setCallback(this, function(response) {
    let state = response.getState();
    if (state === "SUCCESS") {
        console.log(response.getReturnValue());
    }
});
$A.enqueueAction(action);
```

---

2. `persistPurchase`

Descrição:
Este método atualiza o status de um veículo específico para indicar que ele foi vendido. Ele é marcado como `@auraEnabled`, permitindo que seja chamado de componentes Lightning.

Assinatura:

```
@auraEnabled
public static void persistPurchase(Id carId)
```

Funcionamento:

- Recebe o `Id` de um veículo como parâmetro.
- Realiza uma consulta SOQL para buscar o veículo correspondente.
- Atualiza o campo `Sold__c` para `true`.
- Realiza a operação de atualização no banco de dados.
- Lança uma exceção tratada (`AuraHandledException`) em caso de erro.

Exemplo de uso:

```
// Chamando o método em um componente Lightning
let action = component.get("c.persistPurchase");
action.setParams({ carId: "a0123456789XYZ" });
action.setCallback(this, function(response) {
    let state = response.getState();
    if (state === "SUCCESS") {
        console.log("Veículo marcado como vendido.");
    } else {
        console.error(response.getError());
    }
});
$A.enqueueAction(action);
```

Exemplo

Abaixo está um exemplo de como os métodos desta classe podem ser usados em um componente Lightning:

```
// Exemplo de uso do método getVehicles
let action = component.get("c.getVehicles");
action.setCallback(this, function(response) {
    if (response.getState() === "SUCCESS") {
        let vehicles = response.getReturnValue();
        console.log("Veículos disponíveis:", vehicles);
    }
```

```
});
$A.enqueueAction(action);

// Exemplo de uso do método persistPurchase
let purchaseAction = component.get("c.persistPurchase");
purchaseAction.setParams({ carId: "a0123456789XYZ" });
purchaseAction.setCallback(this, function(response) {
    if (response.getState() === "SUCCESS") {
        console.log("Compra registrada com sucesso.");
    }
});
$A.enqueueAction(purchaseAction);
```

Diagrama de Dependência

O diagrama abaixo ilustra as dependências entre os métodos e o objeto customizado `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 : Usa
```

Notas

- Certifique-se de que o objeto customizado `Vehicles__c` e seus campos estejam configurados corretamente no Salesforce.
- O método `persistPurchase` assume que o `Id` fornecido é válido e que o registro existe no banco de dados.

Vulnerabilidades

- **Consulta SOQL sem limite**: Embora as consultas sejam específicas, não há um limite explícito no número de registros retornados. Isso pode causar problemas de desempenho em ambientes com muitos registros.
- **Tratamento de exceções genéricas**: O método `persistPurchase` captura todas as exceções e lança uma `AuraHandledException` com a mensagem original. Isso pode expor informações sensíveis em logs ou para o usuário final.