Priax Dashboards - Customização

- SQL Data Sources

SQL Data Sources

Documentação do Serviço de DataSource SQL para Priax Dashboards

Visão Geral

Este projeto é uma aplicação web simples desenvolvida em Java com JAX-RS para executar queries SQL em um banco de dados MySQL. Ele expõe um endpoint REST que permite enviar uma query via HTTP GET e retorna os resultados em formato JSON.

Inclui autenticação simples via parâmetros na URL.

Requisitos

- Java 8 ou superior
- Maven 3+
- MySQL Server
- WildFly 23

Configuração do Projeto

1. Clonar ou baixar o projeto

Descompacte o arquivo `sqlqueryapp_url_auth.zip` e abra em um editor de sua preferência.

2. Configurar conexão com o MySQL

No arquivo `QueryResource.java`, edite as constantes:

```
private static final String JDBC_URL = "jdbc:mysql://localhost:3306/seubanco";
private static final String JDBC_USER = "usuario";
private static final String JDBC_PASSWORD = "senha";
```

Altere para o nome do banco de dados, usuário e senha corretos.

3. Compilar o projeto

Execute o comando:

```
mvn clean package
```

Isso gerará o arquivo `sqlqueryapp.war` dentro da pasta `target`.

Deploy no WildFly 23

1. Iniciar o servidor WildFly

Certifique-se de que o WildFly 23 está instalado e iniciado.

2. Copiar o WAR para a pasta de deploy

Copie o arquivo `target/sqlqueryapp.war` para a pasta:

```
<WILDFLY_HOME>/standalone/deployments/
```

O WildFly fará o deploy automaticamente.

3. Configurar datasource do MySQL no WildFly

- Copie o JAR do MySQL Connector para `standalone/deployments/` ou `modules/`.
- Crie um datasource via Admin Console ou CLI com o mesmo nome e credenciais usadas no código.

Como consumir o serviço

1. Via browser ou linha de comando

Exemplo de URL:

```
http://localhost:8080/sqlqueryapp/api/query?sql=SELECT+*+FROM+clientes&user=admin&pass=senha
```

2. Via `curl`

```
curl
"http://localhost:8080/sqlqueryapp/api/query?sql=SELECT+*+FROM+clientes&user=admin&pass=senha"
```

3. Via Postman

1. Abra o Postman e crie uma nova requisição GET.
2. Na aba **Params**, adicione:
   - `sql`: `SELECT * FROM clientes`
   - `user`: `admin`
   - `pass`: `senha`
3. Clique em **Send**.

Resposta esperada (exemplo)

```
{
    "columns": [
        {"id": "id", "type": null},
        {"id": "nome", "type": null},
        {"id": "email", "type": null}
    ],
    "values": [
        ["1", "Ana Lima", "ana@example.com"]
    ]
}
```

Considerações de Segurança

- A autenticação por parâmetros na URL é recomendada apenas para ambientes controlados ou desenvolvimento.
- Para produção, recomenda-se o uso de autenticação via cabeçalhos ou JWT.

Se desejar, é possível adaptar o projeto para aceitar também autenticação via header `Authorization`. Basta pedir!