VEM SER

Testes de API Rest com REST Assured


DBC

Aula 3 - Rest Assured (Parte 2)

Sumário

1. Montando o Ambiente
2. BaseTest
3. Data Factory
4. Serialização
5. Desserialização
6. Task RA-03


Montando
o
Ambiente

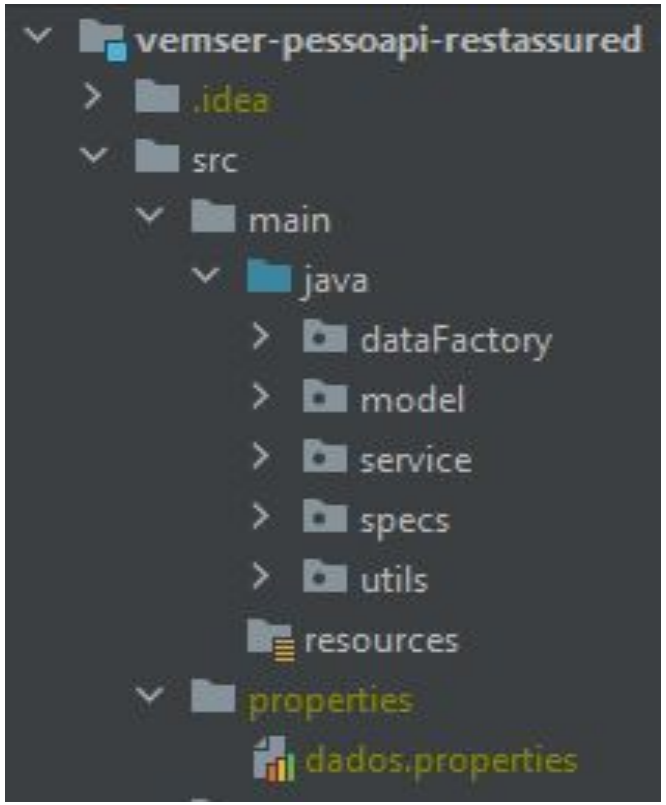
Montando o Ambiente:

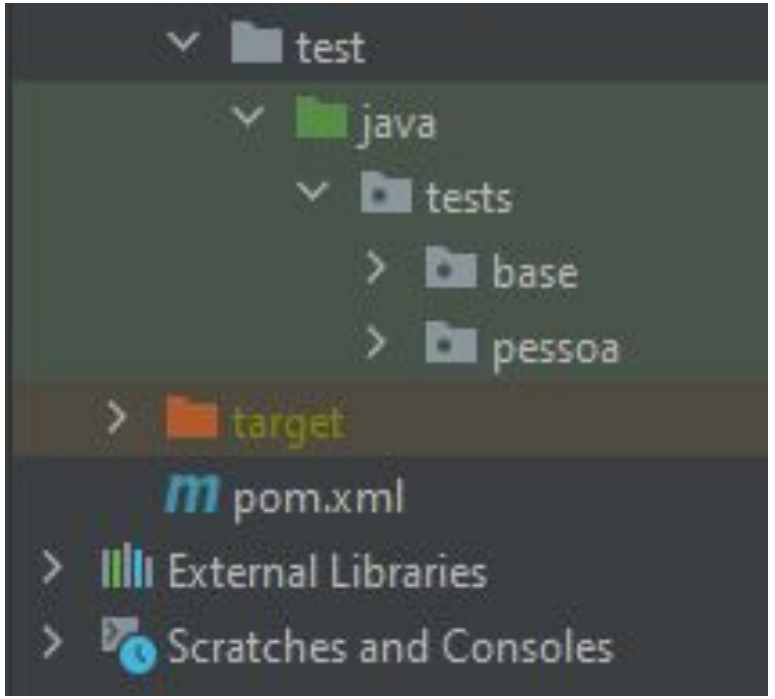
- Criar projeto no **Intellij** com gerenciador de pacotes **Maven**.
  - Nome do projeto: **vemser-pessoaapi-restassured**

- Adicionar dependências ao arquivo POM do projeto:
  - **Rest Assured >> 4.4.0**
    https://mvnrepository.com/artifact/io.rest-assured/rest-assured/4.4.0

  - **JUnit Jupiter (Aggregator) >> 5.9.1**
    https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter/5.9.1

Montando o Ambiente: Estrutura de pastas






Iniciando
os
Testes

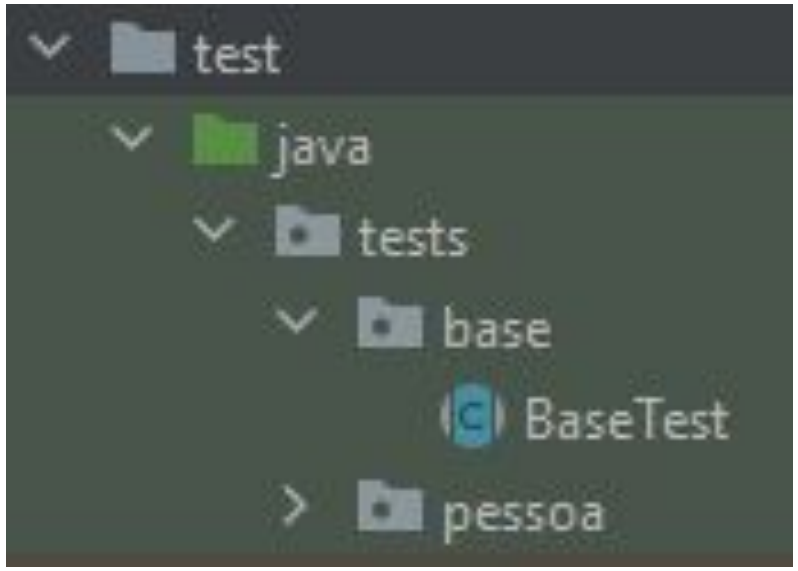
DBC Pessoa API

- **DBC Pessoa API**:

http://vemser-dbc.dbccompany.com.br:39000/vemser/dbc-pessoa-api

Projeto: BaseTest

Classe BaseTest



- BaseTest pode ser definido como um **Padrão de Teste**. Essa Classe define aspectos iniciais para fazer as requisições usando Rest Assured.

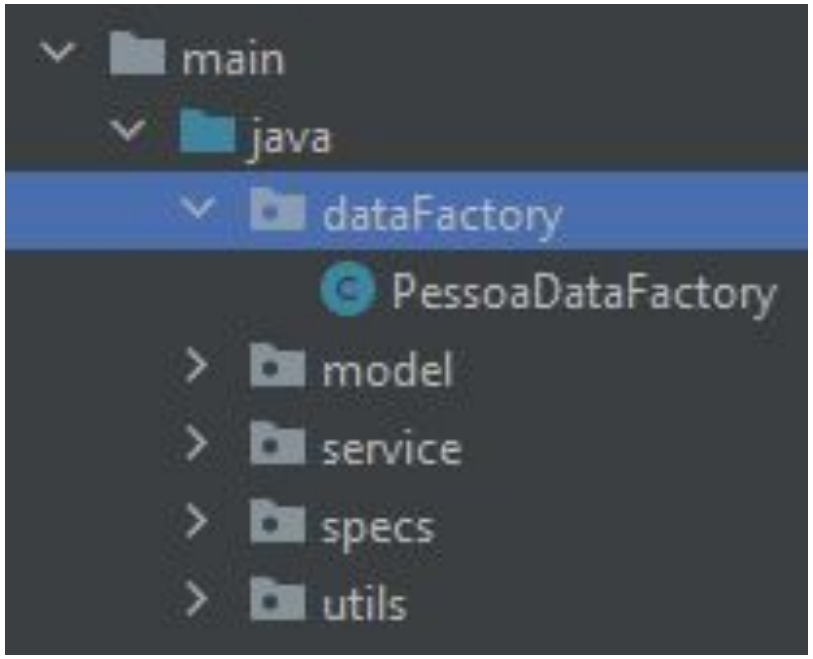
- Evita a duplicação de código, reutilizando comportamentos comuns. E ao centralizar essas ações em um ponto, facilita a manutenção do código.

Para saber mais:

http://www.eliasnogueira.com/base-test-class-testing-pattern-why-and-how-to-use/

Projeto: Data Factory

Data Factory



- Data Factory é uma abordagem de teste cujo objetivo é resolver um dos pontos mais problemáticos da automação, que é a gestão da massa de dados.

- Como o próprio nome diz, trata-se de uma fábrica de dados, ou seja, uma forma fácil e extensível de gerar dados de teste.

Projeto: Data Factory

- **Como implementar:**

  1. **Criar um objeto para modelar os dados;**
  2. **Criar uma classe de fábrica para consumir os dados;**
  3. **Usar a classe de fábrica no teste.**

- **Dependência datafaker:**
  https://mvnrepository.com/artifact/net.datafaker/datafaker/1.7.0

- **Para saber mais:**
  http://www.eliasnogueira.com/test-data-factory-why-and-how-to-use/

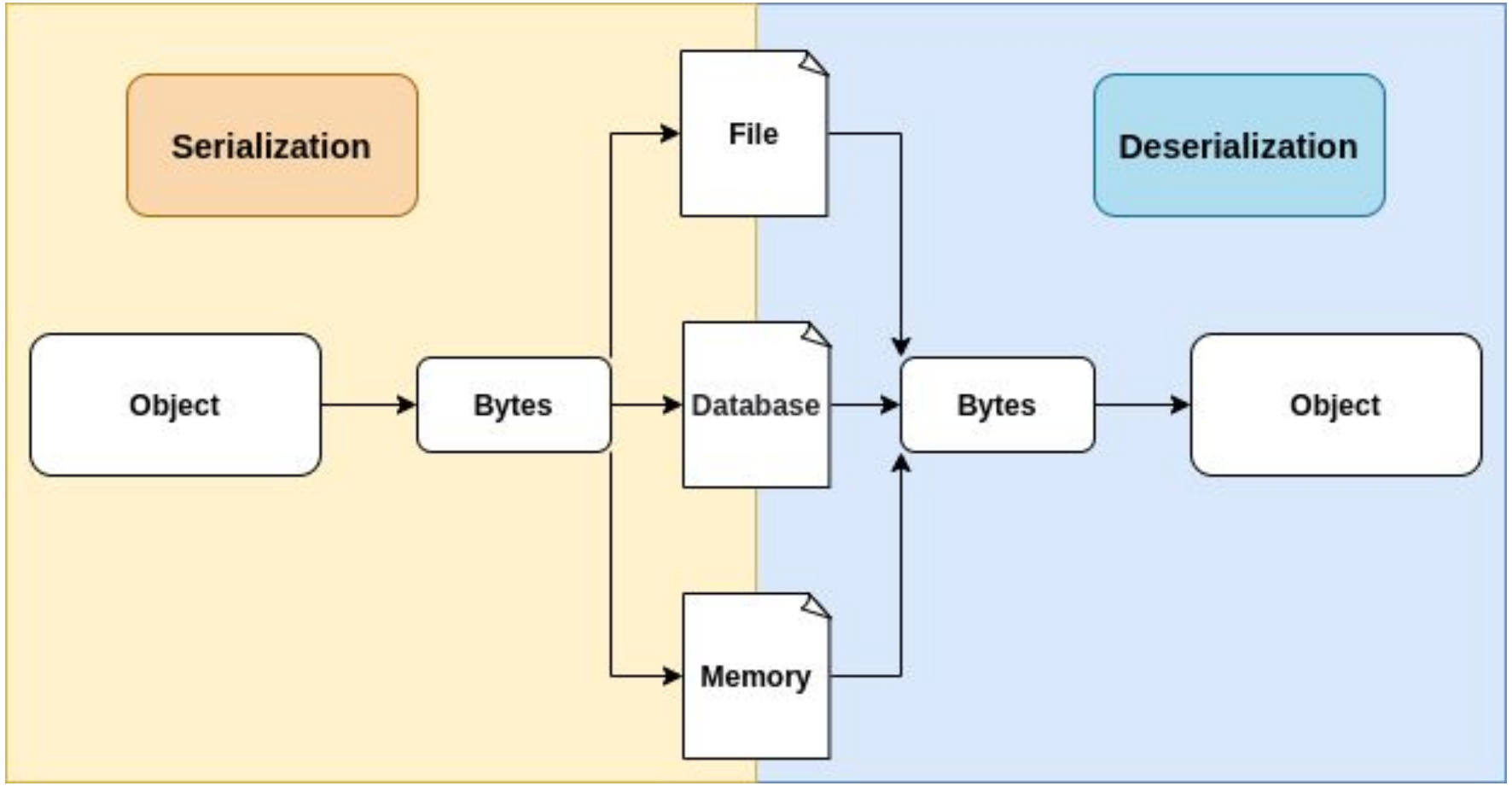
Projeto: Serialização e Desserialização

- **Serialização:** é um processo onde você converte uma instância de uma Classe (Objeto de uma classe) em um fluxo de bytes para armazenar o objeto ou transmiti-lo para a memória, um banco de dados ou um arquivo.

- **Desserialização:** é um processo onde você converte um fluxo de bytes em uma instância de uma Classe (Objeto).

- **Dependência:** Jackson Databind >> 2.14.1
  https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.14.1

Projeto: Serialização e Desserialização



Projeto: Desserialização

- Criar uma Classe que tenha todas as chaves do JSON response;

- Método "**as()**":
  a. Cria uma instância da Classe;
  b. Copia as variáveis do nó JSON para as respectivas variáveis de instância da Classe.

- No Rest Assured:
  - ...then()
    .extract().as(Class.class)

Projeto: Extração do Token

```java
public static String tokenAdmin() {
    AuthModel auth = new AuthModel();
    auth.setLogin(Manipulacao.getProp().getProperty("prop.login"));
    auth.setSenha(Manipulacao.getProp().getProperty("prop.senha"));

    String tokenAdmin =
            given() RequestSpecification
                    .contentType(ContentType.JSON)
                    .body(auth)
            .when()
                    .post(baseURL) Response
            .then() ValidatableResponse
                    .extract().asString()
            ;
    return tokenAdmin;
}
```

Projeto: Utilização do Token

- No Rest Assured:
    - given()
      .auth().oauth2(token)
      .when()..;
    - given()
      .header("Authorization", "Bearer " + token)
      .when()..;

Projeto: Specs

- **Request e Response Specifications**

- **Request Specification:** É uma interface fornecida pelo Rest Assured para agrupar e extrair ações repetitivas, como configurar cabeçalhos, autenticação, verbos HTTP, etc., que podem ser comuns para várias chamadas.

- **Response Specification:** É útil em situações em que um conjunto semelhante de asserções precisa ser feito para várias solicitações Rest.

Projeto: Specs

```java
public static RequestSpecification requestSpec() {

    return new RequestSpecBuilder()
            .addHeader( headerName: "Authorization", Autenticacao.tokenAdmin())
            .setContentType(ContentType.JSON)
            .build();
}
```

1 usage

```java
public static ResponseSpecification responseSpec() {

    return new ResponseSpecBuilder()
            .expectHeader( headerName: "content-type",  expectedValue: "application/json")
            .expectContentType(ContentType.JSON)
            .build();
}
```

Projeto: Assertions

- **Documentação:** https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html

- **Para saber mais:** https://www.petrikainulainen.net/programming/testing/junit-5-tutorial-writing-assertions-with-junit-5-api/


TF - REST
ASSURED

TF REST ASSURED

- Você faz parte da Squad Vem Ser DBC e a equipe de desenvolvimento liberou 5 endpoints de uma nova API que está sendo desenvolvida. Sua atividade, como QA do time, é fazer análise do Swagger e criar testes automatizados para os referidos endpoints.

- **Entrega:**
  - 13/03 - 11:00 - Classroom
  - Slide de apresentação (máximo 4 slides) com métricas:
    - Quantidade de endpoints, quantidade de casos de testes por endpoint, quantidade de bugs encontrados, quantidade de testes com sucesso.

TF REST ASSURED

- **Requisitos:**
  - Trello:
    - Acessar e **fazer cópia**:
      <https://trello.com/invite/b/G5ZS4go6/ATTIb174d93381d589430e990e85b2839f207375D6F5/tf-rest-assured>
    - Praticando o que aprendeu no módulo de Ágil, faça o controle e gestão do fluxo do seu trabalho com Kanban;
    - Praticando o que aprendeu no módulo de Teoria, escreva cenários de teste e faça o Report dos bugs encontrados.

TF REST ASSURED

- **Requisitos:**
  - REST Assured:
    - Utilize o padrão visto em aula, bem como a estrutura de pastas;
    - Utilize:
      - BaseTest
      - DataFactory (com dados dinâmicos)
      - Serialização e Desserialização
      - Spec
      - Assertions

TF REST ASSURED

- API: http://vemser-dbc.dbccompany.com.br:39000/vemser/dbc-pessoa-api

- Lista de endpoints liberados:
  - **POST /pessoa**
  - **GET /pessoa/{cpf}/cpf**
  - **GET /pessoa/byname**
  - **PUT /pessoa/{idPessoa}**
  - **DELETE /pessoa/{idPessoa}**

Referências

- https://rest-assured.io/
- https://github.com/rest-assured/rest-assured/wiki/GettingStarted
- https://www.toolsqa.com/rest-assured/rest-assured-library/
- http://www.eliasnogueira.com/base-test-class-testing-pattern-why-and-how-to-use/
- http://www.eliasnogueira.com/test-data-factory-why-and-how-to-use/
- https://www.dezlearn.com/rest-assured-request-and-response-specifications/
- https://www.baeldung.com/junit5-assertall-vs-multiple-assertions
- https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html


Let's Tech Up Together