Java Spring Expert Capítulo 6

Testes de API com MockMvc e RestAssured

## Tópicos

* Visão geral sobre testes de API
  + Contextualização
  + MockMvc
  + RestAssured
* Exercícios MockMvc
* Exercícios RestAssured

# Visão geral sobre testes de API

## Contextualização

* **Testes de API** são testes que executamos diretamente nos endpoints da nossa aplicação, de forma a validar comportamentos inesperados, evitando que usuários e aplicações desta API recebam resultados inesperados.
* Por meio dos testes é possível avaliar critérios da API como funcionalidade, desempenho, confiabilidade e segurança.
* Testes de API são um tipo de teste de integração
* Lembrando, que os testes de integração são chamados de testes em caixa preta, ou seja, baseado nos requisitos, de forma que especificamos as entradas e saídas da nossa API, buscando cobrir os cenários importantes.
* Vamos aprofundar nos testes de API usando duas ferramentas: **MockMvc** e **Rest Assured**

## MockMvc

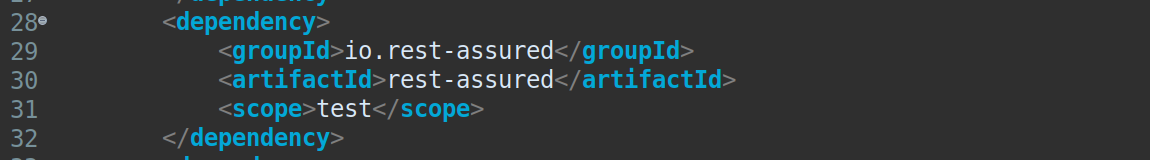
* MockMvc fornece suporte para testarmos nossa aplicação spring, encapsulando todos os beans necessários para testar a camada web;
* Amplamente usado para fazer o teste de integração;
* O teste de integração desempenha um papel importante no ciclo de vida de uma aplicação, verificando o comportamento de ponta a ponta de um sistema;

## Rest Assured

* REST Assured é uma biblioteca usada para automatizar testes de API em aplicações, permitindo testar e validar serviços REST de uma forma simples.
* Estrutura do projeto:
  + Projeto base DSCommerce (cap 5)
  + Projeto DSCommerce Rest Assured
* **Documentação oficial**

<https://rest-assured.io/>

* Para inclusão da dependência, basta incluir a seguinte dependência no arquivo pom.xml, apresentado na Figura 1 (abaixo):
  + <https://mvnrepository.com/artifact/io.rest-assured/rest-assured>



**Figura 1:** Exemplo de inclusão dependência REST Assured

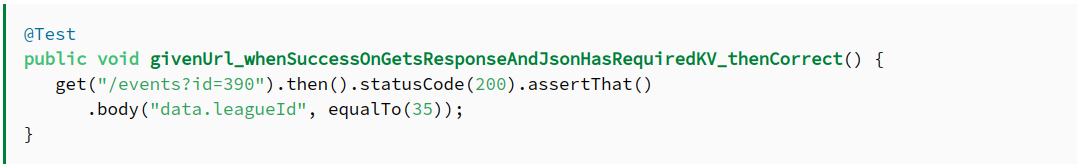
Exemplo básico utilização REST Assured

* Para este exemplo, vamos considerar uma API para consultar partidas entre dois times, conforme a Figura 2 (abaixo):



**Figura 2:** Exemplo API partidas

* Considere que o endpoint para consultar esta partida seja <http://localhost:8080/events?id=390>
* Desta forma, utilizando o Rest Assured podemos fazer uma requisição para o endpoint /events e consultar pelo evento de **id = 390**. Em seguida, podemos acessar os valores retornados no Json e validá-los.
* Na Figura 3 (abaixo), temos um teste em que dada à URI, deve retornar 200 (OK) e em seguida obter o corpo da resposta e validar se o valor **leagueId** corresponde a 35.



**Figura 3:** Exemplo de teste usando REST Assured

## Recursos importantes

* Classe **TokenUtil** (MockMvc)
  + Responsável por obter token de acesso;
  + <https://gist.github.com/oliveiralex/faeba65e214f7e6d738c01516ac7d6d2>
* Dependência **Rest Assured**

| <dependency>  <groupId>io.rest-assured</groupId>  <artifactId>rest-assured</artifactId>  <scope>test</scope>  </dependency> |
| --- |

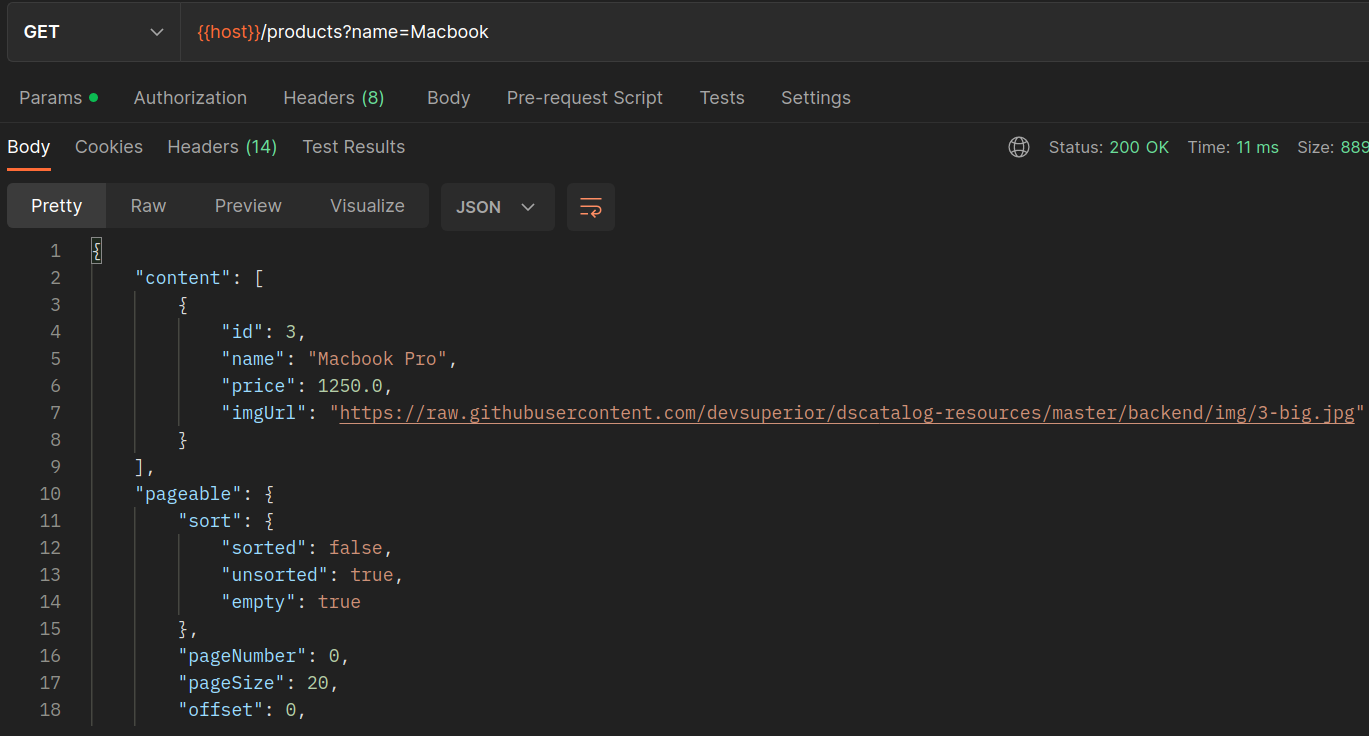
* Dependência para permitir criar um objeto JSON a partir de um Map

| <dependency>  <groupId>com.googlecode.json-simple</groupId>  <artifactId>json-simple</artifactId>  <version>1.1.1</version>  </dependency> |
| --- |

# Exercícios de fixação: Testes de API com MockMvc

**Problema 1:** Consultar produto por nome

Implemente o teste de API usando MockMvc para **consultar produtos por nome**. Desta forma, ao fazer a requisição do tipo **GET** no endpoint /products?name={productName} onde productName deve ser correspondente a string “Macbook” deve retornar como response o status 200 (Ok) e verificar se os campos id, name, price e imgUrl retornados no jsonPath correspondem aos valores da Figura 1 (abaixo).



**Figura 1:** Método findAll products passando como argumento name *Macbook*

**Problema 2:** Inserir produto

Implemente os testes de API usando MockMvc para **inserção de produto** (método **POST** do ProductController), considerando os seguintes cenários. Lembre-se de inserir o token no cabeçalho da requisição.

1. Inserção de produto insere produto com dados válidos quando logado como admin
2. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo *name* for inválido
3. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo *description* for inválido
4. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo *price* for negativo
5. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo *price* for zero
6. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e não tiver *categoria* associada
7. Inserção de produto retorna 403 quando logado como cliente
8. Inserção de produto retorna 401 quando não logado como admin ou cliente

**Problema 3:** Deletar produto

Implemente os testes de API usando MockMvc para **deleção de produto** (método **DELETE** do ProductController), considerando os seguintes cenários. Lembre-se de inserir o token no cabeçalho da requisição.

1. Deleção de produto deleta produto existente quando logado como admin
2. Deleção de produto retorna 404 para produto inexistente quando logado como admin
3. Deleção de produto retorna 400 para produto dependente quando logado como admin
4. Deleção de produto retorna 403 quando logado como cliente
5. Deleção de produto retorna 401 quando não logado como admin ou cliente

**Problema 4:** Consultar pedido por id

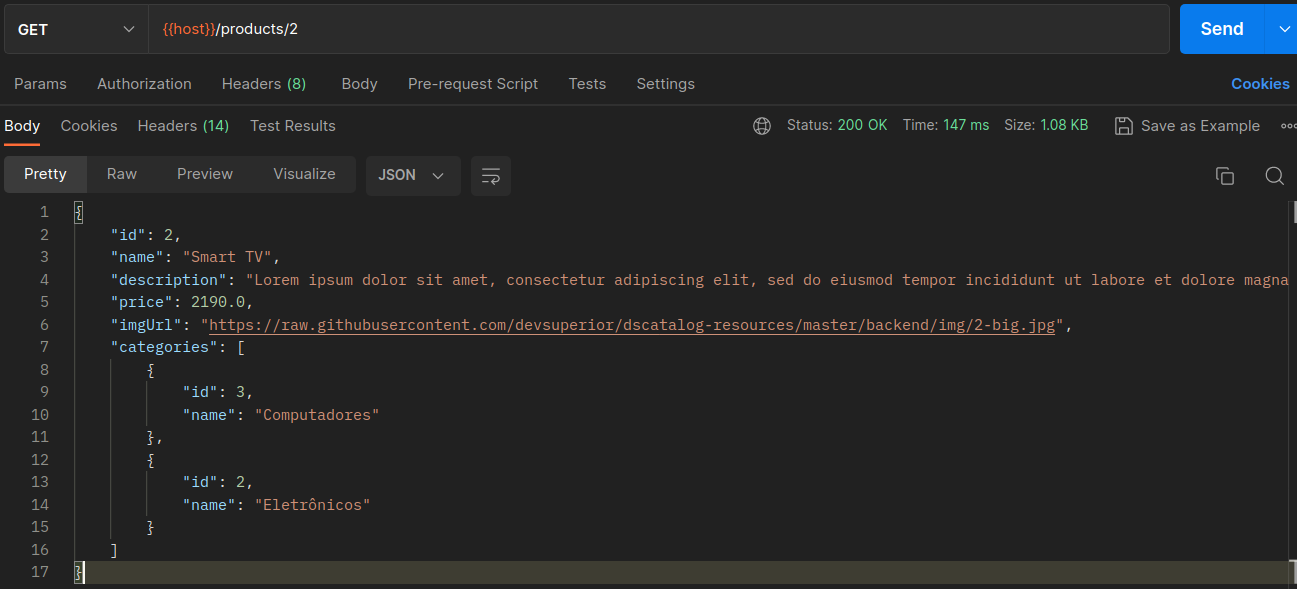
Implemente os testes de API usando MockMvc para **consulta de pedidos por id** (método **GET** do OrderController), considerando os seguintes cenários. Lembre-se de inserir o token no cabeçalho da requisição.

1. Busca de pedido por id retorna pedido existente quando logado como admin
2. Busca de pedido por id retorna pedido existente quando logado como cliente e o pedido pertence ao usuário
3. Busca de pedido por id retorna 403 quando pedido não pertence ao usuário (com perfil de cliente)
4. Busca de pedido por id retorna 404 para pedido inexistente quando logado como admin
5. Busca de pedido por id retorna 404 para pedido inexistente quando logado como cliente
6. Busca de pedido por id retorna 401 quando não logado como admin ou cliente

# Exercícios de fixação: Testes de API com RestAssured

**Problema 1:** Consultar produto por id

Implemente o teste de API usando o REST Assured para **consultar produto com id existente**. Para o teste, você deve fazer uma requisição do tipo **GET** no endpoint */products/{id}* onde id = 2, conforme ilustrado na Figura 1a (abaixo). Em seguida, você deverá verificar se o status da requisição corresponde a 200 (Ok), obter o corpo da resposta e verificar se os campos id, name, imgUrl, price, categories.id e categories.name correspondem aos valores apresentados na Figura 1b (abaixo).



**Figura 1a:** Exemplo consulta de produtos por id

| {  "id": 2,  "name": "Smart TV",  "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore",  "price": 2190.0,  "imgUrl": "https://raw.githubusercontent.com/devsuperior/dscatalog-resources/master/backend/img/2-big.jpg",  "categories": [  {  "id": 3,  "name": "Computadores"  },  {  "id": 2,  "name": "Eletrônicos"  }  ]  } |
| --- |
| **Figura 1b:** Dados do produto com id = 2 |

**Problema 2:** Consultar produtos

Implemente os testes de API usando Rest Assured para **consultar produtos** (método **GET** do ProductController), considerando os seguintes cenários.

1. Busca paginada exibe listagem paginada quando campo nome não preenchido e checa se os produtos *Macbook Pro* e PC *Gamer Tera* estão contidos
2. Busca paginada filtra produtos por nome e exibe listagem paginada quando campo nome preenchidos
3. Busca paginada filtra produtos de forma paginada e filtra produtos com preço maior que *2000.0*

**Problema 3:** Inserir produto

Implemente os testes de API usando Rest Assured para **inserção de produto** (método **POST** do ProductController), considerando os seguintes cenários. Lembre-se de inserir o token no cabeçalho da requisição.

1. Inserção de produto insere produto com dados válidos quando logado como admin
2. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo name for inválido
3. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo description for inválido
4. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo price for negativo
5. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e campo price for zero
6. Inserção de produto retorna 422 e mensagens customizadas com dados inválidos quando logado como admin e não tiver categoria associada
7. Inserção de produto retorna 403 quando logado como cliente
8. Inserção de produto retorna 401 quando não logado como admin ou cliente

**Problema 4:** Deletar produto

Implemente os testes de API usando Rest Assured para **deleção de produto** (método **DELETE** do ProductController), considerando os seguintes cenários. Lembre-se de inserir o token no cabeçalho da requisição.

1. Deleção de produto deleta produto existente quando logado como admin
2. Deleção de produto retorna 404 para produto inexistente quando logado como admin
3. Deleção de produto retorna 400 para produto dependente quando logado como admin
4. Deleção de produto retorna 403 quando logado como cliente
5. Deleção de produto retorna 401 quando não logado como admin ou cliente

**Problema 5:** Consultar pedido por id

Implemente os testes de API usando Rest Assured para **consulta de pedidos por id** (método **GET** do OrderController), considerando os seguintes cenários. Lembre-se de inserir o token no cabeçalho da requisição.

1. Busca de pedido por id retorna pedido existente quando logado como admin
2. Busca de pedido por id retorna pedido existente quando logado como cliente e o pedido pertence ao usuário
3. Busca de pedido por id retorna 403 quando pedido não pertence ao usuário
4. Busca de pedido por id retorna 404 para pedido inexistente quando logado como admin
5. Busca de pedido por id retorna 404 para pedido inexistente quando logado como cliente
6. Busca de pedido por id retorna 401 quando não logado como admin ou cliente