Arun B Chandrasekaran Seguir

17 de junho de 2019 · 4 minutos de leitura · Ouço

Foto de Fred Kearney no Unsplash

Configurar e executar testes do Cucumber no aplicativo Spring Boot

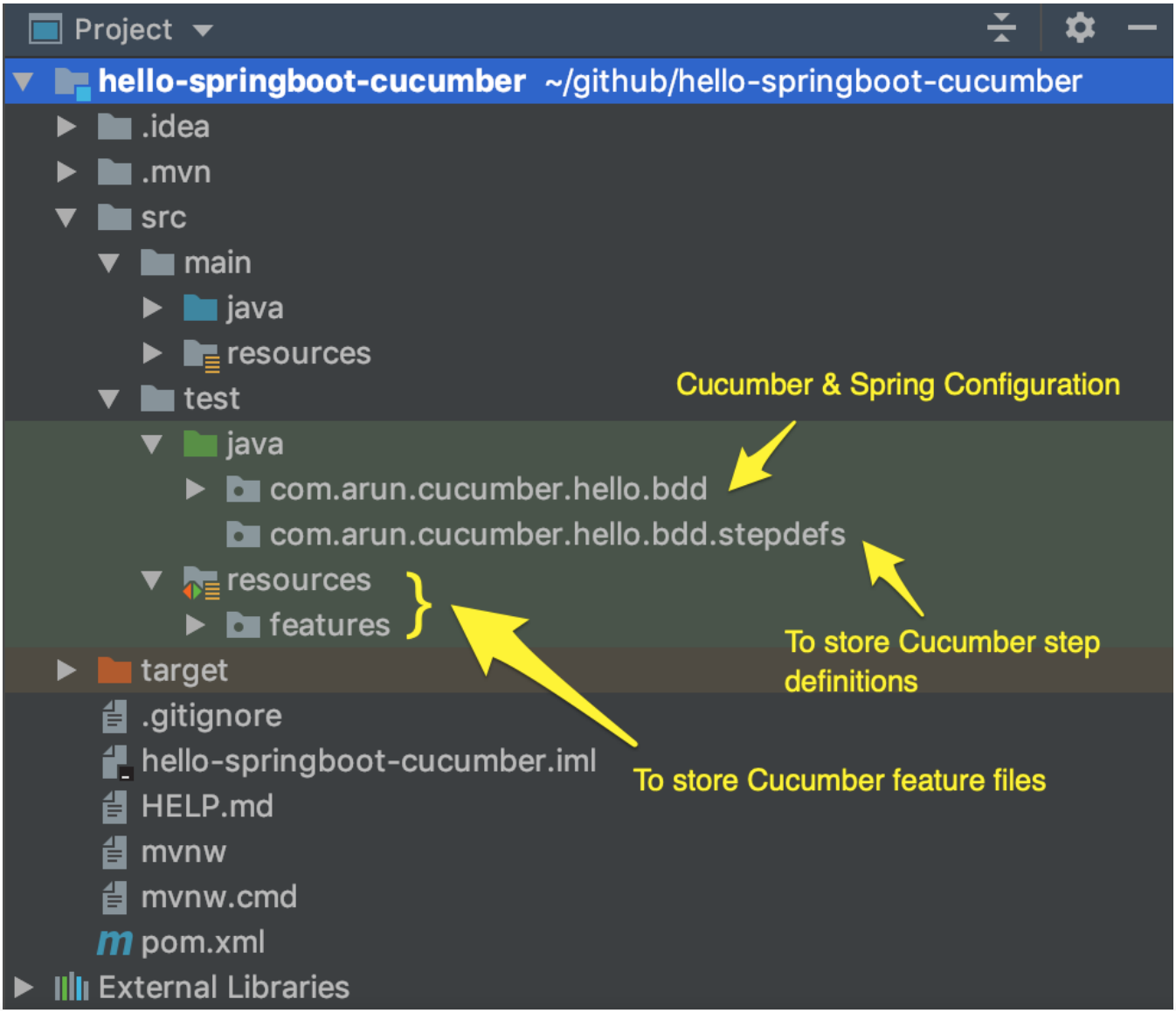
Teste artigo explica como configurar e executar arquivos de recursos do Cucumber no aplicativo Spring Boot .

Ele também explica como usar a classe Cucumber TypeRegistryConfigurer para converter Cucumber DataTable em objetos Java e usar o plugin Cucumber Report para gerar relatórios HTML bem formatados.

Usando o inicializador Spring Boot, crie um projeto Spring Boot com as seguintes dependências.

1. **spring-boot-starter-web** :jar:2.1.5.RELEASE:compile
2. **spring-boot-starter-data-jpa** :jar:2.1.5.RELEASE:compile
3. **h2** :jar:1.4.199:tempo de execução
4. **spring-boot-starter-test** :jar:2.1.5.RELEASE:test
5. **lombok** :jar:1.18.8:compile (opcional)

Estrutura do projeto



Etapa 2: adicione as seguintes dependências ao Maven pom.xml

spring-boot-starter-test : esta dependência importa módulos de teste Spring Boot, bem como JUnit 4, AssertJ, Hamcrest, Mockito, JSONAsset, JsonPath.

Observação: o módulo de teste inicial do Spring Boot contém a anotação ' **@SpringBootTest** ' . Usaremos esta anotação no código abaixo.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>teste</scope>
</dependency>
```

pepino-java8 : Esta dependência ajuda a criar definições de etapa de pepino como lambdas (funções puras).

```
<dependency>
  <groupId>io.cucumber</groupId>
  <artifactId>cucumber-java8</artifactId>
  <version>4.2.0</version>
  <scope>teste</scope>
</dependency>
```

pepino-spring : Para executar scripts do Cucumber no Spring Context.

```
<dependency>
  <groupId>io.cucumber</groupId>
  <artifactId>mola de pepino</artifactId>
  <version>4.2.0</version>
  <scope>teste</scope>
</dependency>
```

```
    <artifactId>pepino-junit</artifactId>
    <version>4.2.0</version>
    <scope>teste</scope>
  </dependency>
```

pepino-report (opcional): para gerar um relatório HTML a partir de pepino-report.json.

> *Eu sugiro fortemente usar este plugin de relatórios para seu projeto e expor o relatório HTML como conteúdo estático junto com o pacote do aplicativo. Os arquivos de recursos do Cucumber são documentação viva do software e este relatório ajuda a conseguir isso.*

```
<dependency>
  <groupId>net.masterthought</groupId>
  <artifactId>relatórios do pepino</artifactId>
  <version>4.2.3</version>
</dependency>
```

> ***Nota:*** *Se você se referir ao código-fonte, poderá encontrar outros utilitários/bibliotecas adicionados como dependências, estes apenas para que o código-fonte de exemplo funcione, você pode usá-los ou usar bibliotecas alternativas.*

Fonte: pom.xml

> *Para obter detalhes adicionais sobre o Cucumber Reporting, consulte a página do projeto no github .*

Etapa 3: Classe para fazer com que o Cucumber use o Spring Context para execução do cenário de teste

```
1  package com.arun.cucumber.hello.bdd;
2
3  import com.arun.cucumber.hello.Application;
4  import cucumber.api.java.Before;
```


```
10  import org.springframework.test.context.ContextConfiguration;
11
12  /**
13   * Class to use spring application context while running cucumber
14   */
15  @SpringBootTest(webEnvironment = WebEnvironment.DEFINED_PORT)
16  @ContextConfiguration(classes = Application.class, loader = SpringBootContextLoader.class)
17  public class CucumberSpringContextConfiguration {
18
19    private static final Logger LOG = LoggerFactory.getLogger(CucumberSpringContextConfiguration.
20
21    /**
22     * Need this method so the cucumber will recognize this class as glue and load spring context
23     */
24    @Before
25    public void setUp() {
26      LOG.info("-------------- Spring Context Initialized For Executing Cucumber Tests ----------
27    }
28  }
```

Fonte: CucumberSpringContextConfiguration.java

Etapa 4: Para configurar e executar o Cucumber

```
1   package com.arun.cucumber.hello.bdd;
2
3   import cucumber.api.CucumberOptions;
4   import cucumber.api.junit.Cucumber;
5   import org.junit.runner.RunWith;
6
7   /**
8    * To run cucumber test.
9    */
10  @RunWith(Cucumber.class)
11  @CucumberOptions(features = "classpath:features", plugin = {"pretty", "json:target/cucumber-rep
12  public class CucumberTest {
```

Fonte: CucumberTest.java

> *A anotação '@CucumberOptions' é usada para configurar o diretório que contém os arquivos de recursos e os plug-ins de saída.*
>
> *Um relatório JSON com o nome* ***"json:target/cucumber-report.json"*** *será gerado pelo Cucumber. É a entrada para o gerador do Cucumber Report.*
>
> *O Cucumber lê automaticamente todos os diretórios do pacote em que está configurado. Certifique-se de que todos os diretórios relacionados ao Cucumber estejam no diretório* ***'bdd'*** *para que sejam automaticamente colados ao contexto de execução do Cucumber. Se você observar o código-fonte, poderá encontrar todas as classes relacionadas ao Cucumber e as definições de etapas no pacote* ***com.arun.hello.bdd*** *neste exemplo.*

Etapa 5: Classe para converter Cucumber all DataTable para Corresponding Java Object (POJO)

```
1  package com.arun.cucumber.hello.bdd;
2
3  import static java.util.Locale.ENGLISH;
4
5  import com.fasterxml.jackson.databind.DeserializationFeature;
6  import com.fasterxml.jackson.databind.ObjectMapper;
7  import com.fasterxml.jackson.databind.SerializationFeature;
8  import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
9  import cucumber.api.TypeRegistryConfigurer;
10 import io.cucumber.cucumberexpressions.ParameterByTypeTransformer;
11 import io.cucumber.datatable.TableCellByTypeTransformer;
12 import io.cucumber.datatable.TableEntryByTypeTransformer;
13 import java.lang.reflect.Type;
14 import java.util.Locale;
15 import java.util.Map;
16 import org.springframework.beans.factory.annotation.Configurable;
17
18 /**
19  * Class used to convert DataTable to Java Object using Jackson (Json library).
```

```
25  public class CucumberTypeRegistryConfigurer implements TypeRegistryConfigurer {
26
27    private ObjectMapper mapper;
28
29    public CucumberTypeRegistryConfigurer() {
30      mapper = new ObjectMapper();
31
32      // To serialize and deserialize java.time.LocalDate, LocalDateTime etc.
33      mapper.registerModule(new JavaTimeModule());
34      mapper.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);
35
36      mapper.enable(DeserializationFeature.READ_UNKNOWN_ENUM_VALUES_AS_NULL);
37    }
38
39    @Override
40    public Locale locale() {
41      return ENGLISH;
42    }
43
44    @Override
45    public void configureTypeRegistry(cucumber.api.TypeRegistry typeRegistry) {
46      Transformer transformer = new Transformer();
47      typeRegistry.setDefaultDataTableCellTransformer(transformer);
48      typeRegistry.setDefaultDataTableEntryTransformer(transformer);
49      typeRegistry.setDefaultParameterTransformer(transformer);
50    }
51
52    private class Transformer implements ParameterByTypeTransformer, TableEntryByTypeTransformer,
53        TableCellByTypeTransformer {
54
55      @Override
56      public Object transform(String s, Type type) {
57        return mapper.convertValue(s, mapper.constructType(type));
58      }
59
60      @Override
61      public <T> T transform(Map<String, String> map, Class<T> aClass,
62          TableCellByTypeTransformer tableCellByTypeTransformer) {
63
64        return mapper.convertValue(map, aClass);
65      }
66
67      @Override
```


```
73  }
```

> *Essa classe pressupõe que você usará nomes de campo em Java POJO como cabeçalhos de coluna Cucumber DataTable.*

Etapa 6 (opcional): configuração do executor de relatórios do Cucumber

```
1   package com.arun.cucumber.hello.bdd;
2
3   import cucumber.api.junit.Cucumber;
4   import java.io.File;
5   import java.util.ArrayList;
6   import java.util.List;
7   import net.masterthought.cucumber.Configuration;
8   import net.masterthought.cucumber.ReportBuilder;
9   import org.junit.runner.notification.RunNotifier;
10  import org.junit.runners.model.InitializationError;
11
12  /**
13   * Class to generate html report from cucumber runner's json report output.
14   */
15  public class CucumberReportRunner extends Cucumber {
16
17    // Can be dynamically pulled from CI Server
18    private static final String PROJECT_NAME = "Hello Cucumber & Spring Boot";
19    private static final String BUILD_NUMBER = "1.0.0";
20    private static final String BRANCH_NAME = "master";
21
22    public CucumberReportRunner(Class clazz) throws InitializationError {
23      super(clazz);
24    }
25
26    @Override
27    public void run(RunNotifier notifier) {
28      super.run(notifier);
29      generateReport();
```


```
35      List<String> jsonFiles = new ArrayList<>();
36      jsonFiles.add("target/cucumber-report.json");
37
38      // set values from respective build tool
39      Configuration configuration = new Configuration(reportOutputDirectory, PROJECT_NAME);
40      configuration.setBuildNumber(BUILD_NUMBER);
41      configuration.addClassifications("Build Number", configuration.getBuildNumber());
42      configuration.addClassifications("Branch Name", BRANCH_NAME);
43
44      ReportBuilder reportBuilder = new ReportBuilder(jsonFiles, configuration);
45      reportBuilder.generateReports();
46    }
47  }
```

Fonte: CucumberReportRunner.java

> *1. Se você decidir usar o plugin Cucumber Report configurado acima, substitua '@RunWith(Cucumber.class)' por '@RunWith(CucumberReportRunner.class)' no arquivo CucumberTest.java.*
>
> *2. CucumberReportRunner.java estende Cucumber.java*
>
> *3. CucumberReportRunner gera relatórios do arquivo em **target/cucumber-report.json***
>
> *4. CucumberReportRunner copia os relatórios html gerados para o diretório target/classes/static para que os relatórios sejam empacotados junto com o aplicativo.*

Código Fonte Completo:

Consulte o github para obter o código-fonte.

Leitura adicional:


Usando Cucumber DataTable para operações CRUD

Os arquivos de recursos são criados usando o Gherkin. Esses arquivos de recursos suportam a representação de dados na forma...

medium.com

Compartilhando o estado entre as definições da etapa do Cucumber usando Java e Spring

No Behavior Driven Development criamos arquivos de recursos para documentar as especificações do produto usando o Gherkin....

medium.com

Referências:

Lista de objetos Java para Cucumber DataTable para executar diff

Divulgação completa: eu escrevi o módulo da tabela de dados para o Cucumber. Mapear objetos manualmente de e para tabelas de...

stackoverflow.com

Guia de referência do Spring Boot

Se você está começando com o Spring Boot, ou "Spring" em geral, comece lendo esta seção. Ele responde a...

docs.spring.io

Pepino