無料電子ブック

学習

apache-camel

Free unaffiliated eBook created from
Stack Overflow contributors.

#apache-camel

1: apache-camelをいめる

Apache Camelは、にエンターープライズののをにするフレームワークです。そのコアでは、ルーティングエンジンエンジンビルダーとえることができます。には、ルートでシステムエンドポイントをすることができます。これらのルートは、のデータのメッセージをけれます。

Apache Camelフレームワークには、スプリッタ、アグリゲータ、コンテンツベースのルーティングなどのEIPエンタープライズパターンのなセットもまれています。このフレームワークは、Javaアプリケーションのさまざまなスタンドアロン、WildFlyやTomcatなどのさまざまなアプリケーションサーバー、またはなエンタープライズサービスバスにできるため、フレームワークとみなすことができます。

フレームワークをいめるには、のいずれかのでプロジェクトにするがあります。

1. Maven
2. け
3. のブート
4. あなたのプロジェクトにいJARライブラリがされました。

Examples

インストールまたはセットアップ

なCamelののにするな。

Maven

Apache Camelをアプリケーションにみむもなの1つは、Mavenのをすることです。のブロックをすることで、MavenはCamelのライブラリとをします。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-core</artifactId>
  <version>2.17.3</version>
</dependency>
```

け

Apache Camelをアプリケーションにみむもうつのなは、Gradleをするです。のをするだけで、GradleはCamelライブラリとそのをインポートします。

```
// https://mvnrepository.com/artifact/org.apache.camel/camel-core
```

```
compile group: 'org.apache.camel', name: 'camel-core', version: '2.17.3'
```

のブート

Camel 2.15、Apache CamelのSpring Bootをできるようになりました。このCamelライブラリとのいは、それがCamelルートのをむ、のをすることです。

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-spring-boot</artifactId>
    <version>${camel.version}</version> <!-- use the same version as your Camel core version -
->
</dependency>
```

キャメルドメインの

CamelのDSLDomain Specific Languageは、Camelをのフレームワークからするの1つです。のいくつかのフレームワークは、XMLファイルのでDSLのもえていますが、DSLはにカスタムベースのでした。

Camelは、Java、Scala、Groovy、XMLなどのプログラミングでのDSLをしています。

たとえば、のリストにすように、なファイルコピールートをさまざまなでできます

- Java DSL

  ```
  from("file:data/in").to("file:data/out");
  ```

- Blueprint / Spring DSLXML

  ```
  <route>
    <from uri="file:data/inbox"/>
    <to uri="file:data/out"/>
  </route>
  ```

- スカラDSL

  ```
  from "file:data/inbox" -> "file:data/out"
  ```

オンラインでapache-camelをいめるをむ https://riptutorial.com/ja/apache-camel/topic/3511/apache-camelをいめる

2: Apache-CamelおよびSpringおよびDBUnitをしたのルートでのテスト

き

このwikiのポイントは、Apache Camelをしてテストをするをすことです。

よりには、これをすると、のルートをからまでのデータベースのにかかわらずちげたり、ルートのとのをしたり、ヘッダーやがしいかどうかをテストすることができます。

がこれまでってきたプロジェクトでは、xmlのなSpringとDBUnitをってテストデータベースをしています。これがあなたにいくつかのリードをえることをっています。

パラメーター

パラメータ/	
	このはラクダのプロセッサーのであなたのルートの
CamelContext	テストでは、ラクダのコンテキストをして、コンテキストをでおよびします。
ProducerTemplate	あなたのルートでメッセージをしたり、でなをしたり、ダミーのヘッダ/ボディをしたりすることができます
AdviceWith	のコンテキストでのルートをするのにちます
WeaveById	のアドバイスのでされ、あなたのルートのにるいをえます *weaveByToString*もえます
MockEndpoint	モックエンドポイントは、テストにしたポイントです。あなたのweaveByIdでは、あなたのルートにのをえることができ、のルートにうのではなく、mockEnpointにることができます。こので、メッセージ、ステータスをすることができます...

いくつかのはにはではありませんが、のコードをするのにちます。よりなをるためのリンクがいくつかあります

- *AdviceWith*と*weaveById* またはルートをトリガーするののについては、のapache-camelのドキュメントをてください。このリンクをしてください
- *ProducerTemplate*のについては、もうドキュメントをしてください このリンクをしてくだ

さい

- すべてのラクダについてにするために Entrepriseインテグレーションパターンのなドキュメント

このテストは、スタックのオーバーフローであっても、つけるのはかなりしいです。これはかなりですが、をねるのをためらうことはありません。おそらくはけることができます。

Examples

キャメルルートの

のルートにはながあります。

- まず、 **ImportDocumentProcess**オブジェクトがデータベースにするかどうかをチェックし、それをヘッダーとしてします
- に、データベースにのImportDocumentProcessにリンクされている **ImportDocumentTraitement**をします。

このルートのコードはのとおりです。

```
@Component
public class TestExampleRoute extends SpringRouteBuilder {

    public static final String ENDPOINT_EXAMPLE = "direct:testExampleEndpoint";

    @Override
    public void configure() throws Exception {
        from(ENDPOINT_EXAMPLE).routeId("testExample")
            .bean(TestExampleProcessor.class,
"getImportDocumentProcess").id("getImportDocumentProcess")
            .bean(TestExampleProcessor.class,
"createImportDocumentTraitement").id("createImportDocumentTraitement")
            .to("com.pack.camel.routeshowAll=true&multiline=true");
    }

}
```

ルートの*ID*はではありません。でBeanをすることもできます。しかし、はあなたのルートがされるにえて、 *ID*のはいとえることができるといます。

キャメルプロセッサの

プロセッサには、ルートでなメソッドだけがまれています。いくつかのメソッドをむなJava Beanです。 プロセッサをして 、 プロセスメソッドをオーバーライドすることもできます。

のコードをしてください。

```
@Component("testExampleProcessor")
public class TestExampleProcessor {
```

```
    private static final Logger LOGGER = LogManager.getLogger(TestExampleProcessor.class);

    @Autowired
    public ImportDocumentTraitementServiceImpl importDocumentTraitementService;

    @Autowired
    public ImportDocumentProcessDAOImpl importDocumentProcessDAO;

    @Autowired
    public ImportDocumentTraitementDAOImpl importDocumentTraitementDAO;

    // ---- Constants to name camel headers and bodies
    public static final String HEADER_ENTREPRISE = "entreprise";

    public static final String HEADER_UTILISATEUR = "utilisateur";

    public static final String HEADER_IMPORTDOCPROCESS = "importDocumentProcess";

    public void getImportDocumentProcess(@Header(HEADER_ENTREPRISE) Entreprise entreprise,
Exchange exchange) {
        LOGGER.info("Entering TestExampleProcessor method : getImportDocumentProcess");

        Utilisateur utilisateur = SessionUtils.getUtilisateur();
        ImportDocumentProcess importDocumentProcess =
importDocumentProcessDAO.getImportDocumentProcessByEntreprise(
                entreprise);

        exchange.getIn().setHeader(HEADER_UTILISATEUR, utilisateur);
        exchange.getIn().setHeader(HEADER_IMPORTDOCPROCESS, importDocumentProcess);
    }

    public void createImportDocumentTraitement(@Header(HEADER_ENTREPRISE) Entreprise
entreprise,
            @Header(HEADER_UTILISATEUR) Utilisateur utilisateur,
            @Header(HEADER_IMPORTDOCPROCESS) ImportDocumentProcess importDocumentProcess,
Exchange exchange) {
        LOGGER.info("Entering TestExampleProcessor method : createImportDocumentTraitement");

        long nbImportTraitementBefore =
this.importDocumentTraitementDAO.countNumberOfImportDocumentTraitement();
        ImportDocumentTraitement importDocumentTraitement =
this.importDocumentTraitementService.createImportDocumentTraitement(
                entreprise, utilisateur, importDocumentProcess, "md5_fichier_example_test",
"fichier_example_test.xml");
        long nbImportTraitementAfter =
this.importDocumentTraitementDAO.countNumberOfImportDocumentTraitement();

        exchange.getIn().setHeader("nbImportTraitementBefore",
Long.valueOf(nbImportTraitementBefore));
        exchange.getIn().setHeader("nbImportTraitementAfter",
Long.valueOf(nbImportTraitementAfter));
        exchange.getIn().setHeader("importDocumentTraitement", importDocumentTraitement);
    }
// Rest of the code contains getters and setters for imported dependencies
}
```

ここでうことはあまりありませんが、たちはをってオブジェクトをあるパートからのパートにします。これはがになプロセスをっているので、のプロジェクトでわれるです。

キャメルテストクラスの

あなたのプロジェクトのにラクダテストサポートとラクテルテストサポートをすることをれないでください。 Mavenユーザーの、をしてください。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-test</artifactId>
  <version>${camel.version}</version>
  <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-test-spring</artifactId>
    <version>${camel.version}</version>
     <scope>test</scope>
</dependency>
```

このクラスは、サンプルルートでテストをトリガしてします。これらのテストでは、 **DBUnit**をして、のまたはののデータベースをするようにコンテキストをできるように、データベースをシミュレートします。

まず、でするCamel Integration Testクラスでのをするためにクラスをします。

```
@RunWith(CamelSpringRunner.class)
@BootstrapWith(CamelTestContextBootstrapper.class)
@ContextConfiguration(locations = { "classpath:/test-beans.xml" })
@DbUnitConfiguration(dataSetLoader = ReplacementDataSetLoader.class)
@TestExecutionListeners({ DependencyInjectionTestExecutionListener.class,
DirtiesContextTestExecutionListener.class,
        DbUnitTestExecutionListener.class })
@DirtiesContext(classMode = ClassMode.AFTER_EACH_TEST_METHOD)
public abstract class AbstractCamelTI {

}
```

をれないようにするか、DAOがしくされません。つまり、コンテキストにされているデータベースをしないは、DBUnitアノテーションをにできます。

な `@DirtiesContext(classMode = ClassMode.AFTER_EACH_TEST_METHOD)` しました。そうすれば、テストごとにラクダのコンテキストがリロードされます。あなたはにルートのをにテストすることができます。しかし、にそれをむなら、あなたはしたくないされたルートので*remove*をうがあります。これはのテストではないとするもいれば、しいだろう。しかし、のように、あなたがリファクタリングするがあるなプロセッサをっているなら、そこからすることができます。

のコードは、テストクラスのをしていますのテストについては、を。

```
@DatabaseSetup(value = { "/db_data/dao/common.xml",
"/db_data/dao/importDocumentDAOCommonTest.xml" })
public class TestExampleProcessorTest extends AbstractCamelTI {
```

```
    @Autowired
    protected CamelContext camelContext;

    @EndpointInject(uri = "mock:catchTestEndpoint")
    protected MockEndpoint mockEndpoint;

    @Produce(uri = TestExampleRoute.ENDPOINT_EXAMPLE)
    protected ProducerTemplate template;

    @Autowired
    ImportDocumentTraitementDAO importDocumentTraitementDAO;

    // -- Variables for tests
    ImportDocumentProcess importDocumentProcess;

    @Override
    @Before
    public void setUp() throws Exception {
        super.setUp();

        importDocumentProcess = new ImportDocumentProcess();
        //specific implementation of your choice
    }
}
```

のテストはルートののをトリガして`mockEndpoint`くことになっているので、
*ImportDocumentProcess*がしくされ、ヘッダにかれているかどうかをテストできます

```
@Test
public void processCorrectlyObtained_getImportDocumentProcess() throws Exception {
    camelContext.getRouteDefinitions().get(0).adviceWith(camelContext, new
AdviceWithRouteBuilder() {

        @Override
        public void configure() throws Exception {
            weaveById("getImportDocumentProcess").after().to(mockEndpoint);
        }
    });

    // -- Launching the route
    camelContext.start();
    template.sendBodyAndHeader(null, "entreprise", company);

    mockEndpoint.expectedMessageCount(1);
    mockEndpoint.expectedHeaderReceived(TestExampleProcessor.HEADER_UTILISATEUR, null);
    mockEndpoint.expectedHeaderReceived(TestExampleProcessor.HEADER_IMPORTDOCPROCESS,
importDocumentProcess);
    mockEndpoint.assertIsSatisfied();

    camelContext.stop();
}
```

のテストはのルートをトリガーします

```
@Test
public void traitementCorrectlyCreated_createImportDocumentTraitement() throws Exception {
    camelContext.getRouteDefinitions().get(0).adviceWith(camelContext, new
AdviceWithRouteBuilder() {
```

```
        @Override
        public void configure() throws Exception {
            weaveById("createImportDocumentTraitement").after().to(mockEndpoint);
        }
    });

    // -- Launching the route
    camelContext.start();

    Exchange exchange = new DefaultExchange(camelContext);
    exchange.getIn().setHeader(TestExampleProcessor.HEADER_ENTREPRISE, company);
    exchange.getIn().setHeader(TestExampleProcessor.HEADER_UTILISATEUR, null); // No user in
this case
    exchange.getIn().setHeader(TestExampleProcessor.HEADER_IMPORTDOCPROCESS,
importDocumentProcess);

    long numberOfTraitementBefore =
this.importDocumentTraitementDAO.countNumberOfImportDocumentTraitement();

    template.send(exchange);

    mockEndpoint.expectedMessageCount(1);
    mockEndpoint.assertIsSatisfied();

    camelContext.stop();

    long numberOfTraitementAfter =
this.importDocumentTraitementDAO.countNumberOfImportDocumentTraitement();
    assertEquals(numberOfTraitementBefore + 1L, numberOfTraitementAfter);
}
```

のルートをのプロセスにリダイレクトすることもできます。しかし、は`mockEndpoint`リダイレクトするがき`mockEndpoint`。 あなたのとヘッダーでにテストをうことができるので、もうしいです。

な このでは、ルートをして`adviceWith`をするために、のコードをしています。

```
camelContext.getRouteDefinitions().get(0).adviceWith(camelContext, new
AdviceWithRouteBuilder() { [...] });
```

しかし 、にとしてされたIDでルートをすることはです

```
 camelContext.getRouteDefinition("routeId").adviceWith(camelContext, new
AdviceWithRouteBuilder() {  [...]  });
```

はこのをくおめします。なぜあなたのテストがしたのかをするのにくのをすることができます

オンラインでApache-CamelおよびSpringおよびDBUnitをしたのルートでのテストをむ
https://riptutorial.com/ja/apache-camel/topic/10630/apache-camelおよびspring-およびdbunit-をしたのルートでのテスト

3: Camel + Redisをしたpub / sub

サイトの

```
producerTemplate.asyncSendBody("direct:myprocedure", massageBody);
```

ManagedCamelで "createProducer"をしてproducerTemplateをする。

Examples

RedisPublisher

```
public class RedisPublisher extends RouteBuilder {

    public static final String CAMEL_REDIS_CHANNEL = "CamelRedis.Channel";
    public static final String CAMEL_REDIS_MESSAGE = "CamelRedis.Message";

    @Value("${redis.host}")
    private String redisHost;
    @Value("${redis.port}")
    private int redisPort;
    @Value("${redis.channel.mychannel}")
    private String redisChannel;

    private String producerName;

    @Required
    public void setProducerName(String producerName) {
        this.producerName = producerName;
    }

    @Override
    public void configure() throws Exception {
        from(producerName)
                .log(String.format("Publishing with redis in channel: %s, massage body:
${body}", redisChannel))
                .setHeader(CAMEL_REDIS_CHANNEL, constant(redisChannel))
                .setHeader(CAMEL_REDIS_MESSAGE, body())
                .to(String.format("spring-redis://%s:%s?command=PUBLISH&redisTemplate=#%s",
redisHost, redisPort, ManagedCamel.REDIS_TEMPLATE));
    }
}
```

RedisSubscriber

```
public class RedisSubscriber extends RouteBuilder {

    @Value("${redis.host}")
    private String redisHost;
    @Value("${redis.port}")
    private int redisPort;
    @Value("${redis.channel.mychannel}")
```

```
    private String redisChannel;

    private Object bean;
    private String method;

    @Required
    public void setBean(Object bean) {
        this.bean = bean;
    }

    @Required
    public void setMethod(String method) {
        this.method = method;
    }

    @Override
    public void configure() throws Exception {
        from(String.format("spring-
redis://%s:%s?command=SUBSCRIBE&channels=%s&serializer=#%s", redisHost, redisPort,
redisChannel, ManagedCamel.REDIS_SERIALIZER))
                .log(String.format("Consuming with redis in channel: %s, massage body:
${body}", redisChannel))
                .process(exchange -> {
                }).bean(bean, String.format("%s(${body})", method));
    }
}
```

されたのにあるメソッド「メソッド」は、けったマッサージをします。

ののコンテキスト

```
<bean id="managedCamel" class="com.pubsub.example.ManagedCamel" >
    <constructor-arg name="routes">
        <list>
            <ref bean="redisSubscriber"/>
        </list>
    </constructor-arg>
</bean>

<bean id="redisSubscriber" class="com.pubSub.example.RedisSubscriber" >
    <property name="bean" ref="myBean"/>
    <property name="method" value="process"/>
</bean>
```

パブリッシャーのスプリングコンテキスト

```
<bean id="managedCamel" class="com.pubSub.example.ManagedCamel" >
    <constructor-arg name="routes">
        <list>
            <ref bean="redisPublisher"/>
        </list>
    </constructor-arg>
</bean>

<bean id="redisPublisher" class="com.pubSub.example.RedisPublisher" >
    <property name="producerName" value="direct:myprocedure"/>
</bean>
```

ManagedCamel

```
public class ManagedCamel implements Managed {

    public static final String REDIS_TEMPLATE = "redisTemplate";
    public static final String LISTENER_CONTAINER = "listenerContainer";
    public static final String REDIS_SERIALIZER = "redisSerializer";
    private DefaultCamelContext camelContext;

    private List<RouteBuilder> routes;
    @Value("${redis.host}")
    private String redisHost;
    @Value("${redis.port}")
    private int redisPort;
    @Value("${redis.password}")
    private String redisPassword;

    public ManagedCamel(List<RouteBuilder> routes) throws Exception {
        this.routes = routes;
    }

    @PostConstruct
    private void postInit() throws Exception {
        JndiRegistry registry = new JndiRegistry();
        final StringRedisSerializer serializer = new StringRedisSerializer();
        RedisTemplate<String, Object> redisTemplate = getRedisTemplate(serializer);
        registry.bind(REDIS_TEMPLATE, redisTemplate);
        RedisMessageListenerContainer messageListenerContainer = new
RedisMessageListenerContainer();
        registry.bind(LISTENER_CONTAINER, messageListenerContainer);
        registry.bind(REDIS_SERIALIZER, serializer);

        camelContext = new DefaultCamelContext(registry);
        for (RouteBuilder routeBuilder : routes) {
            camelContext.addRoutes(routeBuilder);
        }
        start();
    }

    private RedisTemplate<String, Object> getRedisTemplate(StringRedisSerializer serializer) {
        RedisTemplate<String, Object> redisTemplate = new RedisTemplate<String, Object>();
        redisTemplate.setConnectionFactory(redisConnectionFactory());
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        redisTemplate.setValueSerializer(serializer);
        redisTemplate.setEnableDefaultSerializer(false);
        redisTemplate.afterPropertiesSet();
        return redisTemplate;
    }

    private RedisConnectionFactory redisConnectionFactory() {
        final JedisConnectionFactory jedisConnectionFactory = new JedisConnectionFactory();
        jedisConnectionFactory.setHostName(redisHost);
        jedisConnectionFactory.setPort(redisPort);
        jedisConnectionFactory.setPassword(redisPassword);
        jedisConnectionFactory.afterPropertiesSet();
        return jedisConnectionFactory;
    }

    public void start() throws Exception {
        camelContext.start();
```

```
    }

    public void stop() throws Exception {
        camelContext.stop();
    }

    public ProducerTemplate createProducer() {
        return camelContext.createProducerTemplate();
    }

}
```

オンラインでCamel + Redisをしたpub / subをむ https://riptutorial.com/ja/apache-camel/topic/7105/camel-plus-redisをしたpub---sub

クレジット

S. No		Contributors
1	apache-camelをいめる	Community, Michael Hoffman, Namphibian
2	Apache-CamelおよびSpringおよびDBUnitをしたのルートでのテスト	DamienB, Flanfl, matthieusb
3	Camel + RedisをしたPub / Sub	Lior