Extras

A LONG TIME AGO

in a Camel route

far, far away

A LONG TIME AGO

```
public class MyRouteBuilder extends RouteBuilder {

  @Override
  public void configure() throws Exception {
    from("ftp://planet.hoth")
      .choice()
        .when(header("destiny").contains("jedi"))
          .bean(XWing.class, "fly")
          .to("https://planet.dagobah")
        .otherwise()
          .bean(MilleniumFalcon.class, "fly")
          .to("file://cloud.city");
  }

}
```

Wait a second...

- We want to test this, don't we?
  - Who arrives where?
  - What spacecraft are they using?

Step 1

Deal with external resources

External resources

- Use inheritance to separate
  - External bindings
  - Routing logic
- Glue together with
  - **`direct:`**
  - **`direct-vm:`**
  - **`seda:`**

Abstract base class

```
public abstract class BaseRouteBuilder extends RouteBuilder {

  @Override
  public void configure() throws Exception {
    from("direct:hoth")
      .choice()
        .when(header("destiny").contains("jedi"))
          .method(XWing.class, "fly")
          .to("direct:dagobah")
        .otherwise()
          .method(MilleniumFalcon.class, "fly")
          .to("direct:cloudcity");
  }

}
```

External bindings

```
public class MyRouteBuilder extends BaseRouteBuilder {

  @Override
  public void configure() throws Exception {
    // first, add the routes from the parent class
    super.configure();

    // now, add the external bindings
    from("ftp://planet.hoth").to("direct:hoth");
    from("direct:dagobah").to("https://planet.dagobah");
    from("direct:cloudcity").to("file://cloud.city");
  }

}
```

Other options

- **`AdviceWith`**
- Stub out with **`stub:`**
- Use the actual transport
  - **`amq:`**
  - **`http:`**

AdviceWith

```
RouteDefinition definition = context.getRouteDefinition("hoth-route");

definition.adviceWith(context, new RouteBuilder() {
  @Override
  public void configure() throws Exception {
    // intercept sending to cloud city and do something else
    interceptSendToEndpoint("file://cloud.city")
        .skipSendToOriginalEndpoint()
        .to("log:foo")
        .to("mock:cloudcity");
  }
});
```

Step 2

Setup test classes

Test classes

- Provided by Camel
  - Base test classes
  - `@RunWith(...)`
- What's included?
  - Factory methods for context, routes, …
  - Sending/receiving exchange
  - Helper methods

Test classes

```
public class StarWarsUnitTest extends CamelTestSupport {

  //TODO: Add your unit tests

  @Override
  protected RouteBuilder createRouteBuilder() throws Exception {
    return new BaseRouteBuilder() {
      @Override
      public void configure() throws Exception {
        super.configure();

        //TODO: Add your bindings here
      }
    };
  }
}
```

Test endpoints

- Endpoints for testing
  - **`mock:`**
  - **`test:`**
  - **`dataset:`**

TEST ENDPOINTS

```
public class StarWarsUnitTest extends CamelTestSupport {

  //TODO: Add your unit tests

  @Override
  protected RouteBuilder createRouteBuilder() throws Exception {
    return new BaseRouteBuilder() {
      @Override
      public void configure() throws Exception {
        super.configure();

        from("direct:dagobah").to("mock:dagobah");
        from("direct:cloudcity").to("mock:cloudcity");
      }
    };
  }
}
```

Step 3

Write tests

Write tests

- Workflow
  - Expectations
  - Send messages
  - Assertions

EXPECTATIONS

- Record expectations on mock endpoints
  - (min/max) number of messages
  - Message contents
  - Message ordering
  - Processor to do validation

EXPECTATIONS

```
public class StarWarsUnitTest extends CamelTestSupport {

    @Test
    public void testSimpleRoute() throws InterruptedException {
        getMockEndpoint("mock:dagobah")
          .expectedBodiesReceived("Luke Skywalker");
        getMockEndpoint("mock:cloudcity").expectedMessageCount(2);
    }
}
```

Send messages

- Convenience method **`sendBody`**
- Use **`ProducerTemplate`**
  - **`sendXxx`**
  - **`requestXxx`**
  - **`asyncXxx`**

SEND MESSAGES

```
public class StarWarsUnitTest extends CamelTestSupport {

    @Test
    public void testSimpleRoute() throws InterruptedException {
        getMockEndpoint("mock:dagobah")
          .expectedBodiesReceived("Luke Skywalker");
        getMockEndpoint("mock:cloudcity").expectedMessageCount(2);

        sendBody("direct:hoth", "Leia Organa");
        sendBody("direct:hoth", "Han Solo");
        template.sendBodyAndHeader("direct:hoth", "Luke Skywalker",
                                   "destiny", "jedi");

    }
}
```

Assertions

- Assert methods on mock endpoints
  - (not) satisfied
  - (no) duplicates
  - await
- **`CamelTestSupport`**
  - Convenience method for assertions

ASSERTIONS

```
public class StarWarsUnitTest extends CamelTestSupport {

    @Test
    public void testSimpleRoute() throws InterruptedException {
        getMockEndpoint("mock:dagobah")
          .expectedBodiesReceived("Luke Skywalker");
        getMockEndpoint("mock:cloudcity").expectedMessageCount(2);

        sendBody("direct:hoth", "Leia Organa");
        sendBody("direct:hoth", "Han Solo");
        template.sendBodyAndHeader("direct:hoth", "Luke Skywalker",
                                   "destiny", "jedi");

        assertMockEndpointsSatisfied();
    }
}
```

ASSERTIONS

```
public class StarWarsUnitTest extends CamelTestSupport {

    @Test
    public void testSimpleRoute() throws InterruptedException {
        // SNIP : make some room on the slide
        assertMockEndpointsSatisfied();

        List<Exchanges> exchanges =
          getMockEndpoint("mock:dagobah").getExchanges();

        for (Exchange exchange : exchanges) {
          assertEquals("XWing", exchange.getIn().getHeader("vehicle"));
        }
    }
}
```

NotifyBuilder

- Another option: **`NotifyBuilder`**
- Workflow with build pattern
  - express conditions (for exchanges) in DSL
    - exchanges done/completed/failed
    - from endpoint
    - bodies received (with expressions)
    - …
  - **`create`**
  - **`matches`**

NotifyBuilder

```
NotifyBuilder notify = new NotifyBuilder(context)
   .from("direct:hoth").whenDone(5)
   .create();

notify.matches(10, TimeUnit.SECONDS);
notify.reset();

NotifyBuilder notify = new NotifyBuilder(context)
   .from("direct:hoth").whenDone(5)
   .and().from("direct:tatatooine").whenDone(7)
   .create();
```

The End

May the Camel Unit Testing

Force be with you