Annotation Based Expression Language

Annotation Based Expression Language

You can also use any of the Languages supported in Camel to bind expressions to method parameters when using Bean Integration. For example you can use any of these annotations:

Annotation	Description
@Bean	Inject a Bean expression
@BeanShell	Inject a BeanShell expression
@Constant	Inject a Constant expression
@EL	Inject an EL expression
@Groovy	Inject a Groovy expression
@Header	Inject a Header expression
@JavaScript	Inject a JavaScript expression
@MVEL	Inject a MVEL expression
@OGNL	Inject an OGNL expression
@PHP	Inject a PHP expression
@Python	Inject a Python expression
@Ruby	Inject a Ruby expression
@Simple	Inject an Simple expression
@XPath	Inject an XPath expression
@XQuery	Inject an XQuery expression

Example:

```
public class Foo {

    @MessageDriven(uri = "activemq:my.queue")
    public void doSomething(@XPath("/foo/bar/text()") String correlationID, @Body String body) {
                // process the inbound message here
    }
}
```

Advanced example using @Bean

And an example of using the the @Bean binding annotation, where you can use a POJO where you can do whatever java code you like:

```
public class Foo {

    @MessageDriven(uri = "activemq:my.queue")
    public void doSomething(@Bean("myCorrelationIdGenerator") String correlationID, @Body String body) {
                // process the inbound message here
    }
}
```

And then we can have a spring bean with the id **myCorrelationIdGenerator** where we can compute the id.

```
public class MyIdGenerator {

    private UserManager userManager;

    public String generate(@Header(name = "user") String user, @Body String payload) throws Exception {
       User user = userManager.lookupUser(user);
       String userId = user.getPrimaryId();
       String id = userId + generateHashCodeForPayload(payload);
       return id;
   }
}
```

The POJO MyIdGenerator has one public method that accepts two parameters. However we have also annotated this one with the @Header and @Body annotation to help Camel know what to bind here from the Message from the Exchange being processed.

Of course this could be simplified a lot if you for instance just have a simple id generator. But we wanted to demonstrate that you can use the Bean Binding annotations anywhere.

```
public class MySimpleIdGenerator {

    public static int generate()  {
       // generate a unique id
       return 123;
   }
}
```

And finally we just need to remember to have our bean registered in the Spring Registry:

```
   <bean id="myCorrelationIdGenerator" class="com.mycompany.MySimpleIdGenerator"/>
```

Example using Groovy

In this example we have an Exchange that has a User object stored in the in header. This User object has methods to get some user information. We want to use Groovy to inject an expression that extracts and concats the fullname of the user into the fullName parameter.

```
    public void doSomething(@Groovy("$request.header['user'].firstName $request.header['user'].familyName)
String fullName, @Body String body) {
               // process the inbound message here
    }
```

Groovy supports GStrings that is like a template where we can insert $ placeholders that will be evaluated by Groovy.