Example of AR for breakage and surprise bug

bug-report: https://issues.apache.org/jira/browse/CAMEL-1948

Description:

This hampers restart of services by example the overhauled JMX in CAMEL-1933.When a service was restarted it had the following **incorrect state: started = true starting = false stopped = true stopping = false** The stopped should have been changed to false as its started.

Code:

```
public void testServiceSupport() throws Exception {
        MyService service = new MyService();
        service.start();
        assertEquals(true, service.isStarted());
        assertEquals(false, service.isStarting());
        assertEquals(false, service.isStopped());
        assertEquals(false, service.isStopping());service.stop();
        assertEquals(true, service.isStopped());
        assertEquals(false, service.isStopping());
        assertEquals(false, service.isStarted());assertEquals(false, service.isStarting());
```

Example of CTL for breakage bug

bug-report: https://issues.apache.org/jira/browse/CAMEL-2935

Description:

The recent change in ProducerCache.sendProducerCache.java public void send(Endpoint endpoint Exchange exchange) **{ try { sendExchange(endpoint null null exchange); // RECENT CHANGE HERE: // ensure that CamelExecutionException is always thrown if (exchange.getException() != null) { exchange.setException(wrapCamelExecutionException(exchange exchange.getException())); } } catch (Exception e) { throw wrapCamelExecutionException(exchange e); } }that throws a CamelExecutionException if exchange.getException is not null makes it impossible for DefaultProducerTemplate.asyncCallback to report failures (other than fault messages) asynchronously via Synchronization.onFailureDefaultProducerTemplate.java public Future&lt;Exchange&gt; asyncCallback(final Endpoint endpoint final Exchange exchange final Synchronization onCompletion) { Callable&lt;Exchange&gt; task = new Callable&lt;Exchange&gt;() { public Exchange call() throws Exception** { // FIXME: exception is thrown in Camel 2.4 where a normal return with answer.getException != null was done in Camel 2.3 Exchange answer = send(endpoint exchange); I attached a patch for DefaultProducerTemplateAsyncTest that demonstrates the problem. I didn't commit a fix yet because I'm unsure at the moment about the best way to fix that. Of course I tried a naive fix in the DefaultProducerTemplate.asyncCallback methods which causes the test (in the patch) to pass but I'd like to hear other opinions before I continue. }

Code:

```
protected Object extractResultBody(Exchange result) {
    Object answer = null;
    if (result != null) {
        // rethrow if there was an exception
        if (result.getException() != null) {
            throw wrapRuntimeCamelException(result.getException());
        }

        // okay no fault then return the response
        if (result.hasOut()) {
            // use OUT as the response
            answer = result.getOut().getBody();
        } else {
            // use IN as the response
            answer = result.getIn().getBody();
        }
    }
    return answer;
}
```

Example of CI for surprise bug

bug-report: https://issues.apache.org/jira/browse/CAMEL-2016

Description:

The package is already exported by camel-core so the class in this package needs to be moved to another package. **An error occurred while defining the constructor declaration**

Code:

```
public class TraceInterceptorTest extends ContextTestSupport {

    // START SNIPPET: e1
    public void TraceInterceptorTest() throws Exception {
        template.sendBodyAndHeader("direct:start", "Hello London", "to", "James");
        template.sendBodyAndHeader("direct:start", "This is Copenhagen calling",
"from", "Claus");
    }
protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            public void configure() throws Exception {
                // enable tracing
                getContext().setTracing(true);

                from("direct:start").routeId("foo").
                        process(new Processor() {
                            public void process(Exchange exchange) throws Exception {
                                // do nothing
                            }

                            @Override
                            public String toString() {
                                return "MyProcessor";
                            }
                        }).
                        to("mock:foo").
                        to("direct:bar");

                from("direct:bar").routeId("bar").to("mock:bar");
            }
        };
    }
```

Example of CTL for surprise bug

bug-report: https://issues.apache.org/jira/browse/CAMEL-2529

Description:

This unit test can shows the issue that selector option don't work for **ConsumerTemplate@Test public void testConsumerTemplate() throws Exception { template.sendBodyAndHeader('activemq:queue:consumer' 'Message1' 'SIZE_NUMBER' 1505); template.sendBodyAndHeader('activemq:queue:consumer' 'Message3' 'SIZE_NUMBER' 1300); template.sendBodyAndHeader('activemq:queue:consumer' 'Message2' 'SIZE_NUMBER' 1600); // process every exchange which is ready. If no exchange is left break // the loop while (true) { Exchange ex = consumer.receiveNoWait('activemq:queue:consumer?selector=SIZE_NUMBER&lt;1500'); if (ex != null) { Message message = ex.getIn(); int size = message.getHeader('SIZE_NUMBER' int.class); assertTrue('The message header SIZE_NUMBER should be less than 1500' size &lt; 1500); assertEquals('The message body is wrong' 'Message3' message.getBody()); } else { break; } }** }And here is mail thread which discusses about it.

Code:

```
public static class Consumer {

    @Autowired
    protected ConsumerTemplate consumer;

    @Handler
    public String consume() {
        StringBuilder result = new StringBuilder();

        Exchange exchange;
        while ((exchange = consumer.receive("activemq:queue", 2000)) != null) {
            result.append(exchange.getIn().getBody(String.class));
        }

        return result.toString();

    }
}
```

Example of CI for breakage bug issue-id:6305

bug-report: https://issues.apache.org/jira/browse/CAMEL-6305

Description:

A test that extends CamelBlueprintTestSupport does not get its debugBefore() and debugAfter() methods called.

Code:

```
public class DebugBlueprintTest extends CamelBlueprintTestSupport {

    private boolean debugBeforeMethodCalled;
    private boolean debugAfterMethodCalled;

    // override this method, and return the location of our Blueprint XML file to be used for testing
    @Override
    protected String getBlueprintDescriptor() {
        return "org/apache/camel/test/blueprint/camelContext.xml";
    }



    @Override
    public boolean isUseDebugger() {
        // must enable debugger
        return true;
    }

    @Override
    protected void debugBefore(Exchange exchange, org.apache.camel.Processor processor, ProcessorDefinition<?> definition, String id, String label) {
        log.info("Before " + definition + " with body " + exchange.getIn().getBody());
        debugBeforeMethodCalled = true;
    }

    @Override
    protected void debugAfter(Exchange exchange, org.apache.camel.Processor processor, ProcessorDefinition<?> definition, String id, String label, long timeTaken) {
        log.info("After " + definition + " with body " + exchange.getIn().getBody());
        debugAfterMethodCalled = true;
    }
}
```