Item 123
git_comments:

1. * * Set the message exchange pattern on the exchange * @param pattern exchange pattern * @return exchange builder
2. * * Create the exchange by setting the camel context * @param context the camel context * @return exchange builder
3. * * Set the exchange property * @param pattern exchange pattern * @return exchange builder
4. * * Set the in message body on the exchange * @param body * @return exchange builder
5. * * Build up the exchange from the exchange builder * @return exchange
6. * * Set the message header of the in message on the exchange * @param key the key of the header * @param value the value of the header * @return exchange builder
7. * * Set the message exchange pattern on the exchange * @param pattern exchange pattern * @return exchange builder
8. * * Create the exchange by setting the camel context * @param context the camel context * @return exchange builder
9. * * Set the exchange property * @param pattern exchange pattern * @return exchange builder
10. * * Set the in message body on the exchange * @param body * @return exchange builder
11. * * Build up the exchange from the exchange builder * @return exchange
12. * * Set the message header of the in message on the exchange * @param key the key of the header * @param value the value of the header * @return exchange builder

git_commits:

1. **summary:** CAMEL-3104 moving the ExchangeBuilder to builder package and added java doc for the end user API
   message: CAMEL-3104 moving the ExchangeBuilder to builder package and added java doc for the end user API git-svn-id: https://svn.apache.org/repos/asf/camel/trunk@1378464 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** ExchangeBuilder to create messages using fluent builder style
   description: So you from the Java API using a fluent builder style, can create an Exchange to send, for example with the ProducerTemplate. Currently you can use a inlined Processor to do that but its a bit ugly and verbose. {code} Exchange exchange = ExchangeBuilder("activemq:queue:foo").withBody("Hello World").andHeader("JMSReplyTo", "bar").andHeader("foo", 123).create(); template.send(exchange); {code} The trick is the Exchange has to be created by the producer and hence we need to know the endpoint uri where to send it.

jira_issues_comments:

1. Here is a try. I couldnot figure out how to use the endpoint uri from your examples during Exchange creation.
2. Hi, sorry to grave dig this ticket; I was just wondering where this ticket stands currently, was there was anything wrong with Bilgin's suggested code? Thanks Alan
3. The patch looks good and I will commit it with some more enhancement to support set the exchange properties with it. As this change only add the new API we don't break the old ones, it could be safe for us to add it in new coming camel 2.11.x branch.

4. We have to consider Camel 2.x API stable, and therefore be more careful when adding stuffs to the core that is API facing to our end users. For example which package should it be placed? In the root package, or the builder package? Or in the util, which seems a bit misplaced. But it will be a nice addition, and as Willem says, does not affect other APIs in the core, and therefore not impose any risks.
5. **body:** I think the class should be moved to the builder package, and be declared as final (not ment for inheritance for end users). And it should have javadoc, as its an end user API. Also the headers should only be set if there is any headers (eg not empty). And I guess we could add API for attachments as well? Or do we want the builder API to be kept simple? And what if you want to set the message as a fault message? I know its seldom used? As well setting the messageId on the Message? I guess that is more often a component specific issue that they can do that. I guess we should not add these as ppl can set them manually afterwards directly using the existing Exchange API.
   label: code-design
6. @Claus, Yeah, we can keep it running once we start to working on the code :) I will try to address most of your suggestion in my next commit.