Item 120
git_comments:

git_commits:

1. **summary:** CAMEL-1184 fixed the build error
   message: CAMEL-1184 fixed the build error git-svn-id: https://svn.apache.org/repos/asf/activemq/camel/trunk@726339 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
2. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
3. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
   label: code-design
4. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
5. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
6. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
7. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
   label: documentation
8. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
   label: documentation
9. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
   description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
10. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
11. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
    label: code-design
12. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
    label: documentation

13. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
14. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
    label: code-design
15. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
16. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
17. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
    label: code-design
18. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
19. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.
20. **summary:** Add a new Dataformat - tidyMarkup - which allows us to unmarshal bad HTML to good (XML) Html.
    description: Using TagSoup, a competent 'bad html' to good well formed (xml) Html, we can create a new dataformat such that .. from("direct:fromSomeHttpSite") .unmarshal().tidyMarkup() .setBody().xpath("//table/tr/td[1]") .to("direct:foo") we get to turn the nasty HTML into goody HTML which can go through XSLT components and be xpathed and all the goodness we love.

jira_issues_comments:

1. Closing 2.0m1 tickets
2. Patch supplied as per Claus' recommendations. Please review and comment. regards Ramon
3. **body:** Review away! Go hard in there (with the review), and check to make sure I have not made any camel newbie mistakes (remember, this is my VERY first attempt at a camel-*). I have half considered adding in a component but saw after writing the dataFormat that for now it is probably enough. If need be I can write the component version of it too, simple enough. Cheers Ramon
   label: code-design
4. Nice work Ramon! I can review/commit this if you'd like.
5. Marking this as resolved for 2.0.0 only. If there is a specific need for this in 1.x, we can merge it to the fixes branch then.
6. Thanks James.. So Jonathan, it is missing this one liner. rbuckland@itasca:~/projects/external/001_camel/trunk$ svn diff camel-core/src/main/resources/org/apache/camel/model/dataformat/jaxb.index Index: camel-core/src/main/resources/org/apache/camel/model/dataformat/jaxb.index === --- camel-core/src/main/resources/org/apache/camel/model/dataformat/jaxb.index (revision 725686) +++ camel-core/src/main/resources/org/apache/camel/model/dataformat/jaxb.index (working copy) @@ -27,4 +27,5 @@ StringDataFormat XMLBeansDataFormat XStreamDataFormat -ZipDataFormat \ No newline at end of file +ZipDataFormat +TidyMarkupDataFormat
7. **body:** And of course, once (if) it gets committed, I will add the appropriate wiki documentation to boot, with good examples.
   label: documentation
8. **body:** Will let you know how it goes. Thanks for the contribution! And yes, documentation is the second part of any ticket, too bad we don't have a wiki checkbox for that :)
   label: documentation
9. patch file attached.
10. Already added in ;)
11. **body:** I've committed your patch in rev 725715. Thanks again! Some minor changes I made: - Checkstyle errors - Added dataformat to UnmarshalType + test case for this (TidyMarkupDataFormatWithUnmarshalTypeTest.java) - Added OSGi exports - Removed runtime dependency on camel-spring - you only need camel-core - Added log4j config file - Change to TidyMarkupDataFormat.java {code} public Object unmarshal(Exchange exchange, InputStream inputStream) throws Exception { if (dataObjectType.isAssignableFrom(String.class)) { return asStringTidyMarkup(inputStream); } else if (dataObjectType.isAssignableFrom(Node.class)) { - return asStringTidyMarkup(inputStream); + return asNodeTidyMarkup(inputStream); } else { throw new CamelException("The return type [" + dataObjectType.getCanonicalName() + "] is unsupported"); } } {code}
    label: code-design
12. **body:** Ramon do you have wiki edit rights? Do you mind taking a stab on documenting it in the wiki? I have created a subtask for the job.
    label: documentation
13. Source files and changes to pom.xml and DataFormat class. rbuckland@itasca:~/projects/external/001_camel/trunk$ svn update At revision 725686. rbuckland@itasca:~/projects/external/001_camel/trunk$ svn status | egrep "^[AM]" M components/pom.xml A

components/camel-tagsoup A components/camel-tagsoup/src A components/camel-tagsoup/src/test A components/camel-tagsoup/src/test/java A components/camel-tagsoup/src/test/java/org A components/camel-tagsoup/src/test/java/org/apache A components/camel-tagsoup/src/test/java/org/apache/camel A components/camel-tagsoup/src/test/java/org/apache/camel/dataformat A components/camel-tagsoup/src/test/java/org/apache/camel/dataformat/tagsoup A components/camel-tagsoup/src/test/java/org/apache/camel/dataformat/tagsoup/TidyMarkupTestSupport.java A components/camel-tagsoup/src/test/java/org/apache/camel/dataformat/tagsoup/TidyMarkupDataFormatAsDomNodeTest.java A components/camel-tagsoup/src/test/java/org/apache/camel/dataformat/tagsoup/TidyMarkupDataFormatAsStringTest.java A components/camel-tagsoup/src/test/resources A components/camel-tagsoup/src/test/resources/org A components/camel-tagsoup/src/test/resources/org/apache A components/camel-tagsoup/src/test/resources/org/apache/camel A components/camel-tagsoup/src/test/resources/org/apache/camel/dataformat A components/camel-tagsoup/src/test/resources/org/apache/camel/dataformat/tagsoup A components/camel-tagsoup/src/test/resources/org/apache/camel/dataformat/tagsoup/testfile1.html A components/camel-tagsoup/src/test/resources/org/apache/camel/dataformat/tagsoup/testfile2-evilHtml.html A components/camel-tagsoup/src/main A components/camel-tagsoup/src/main/java A components/camel-tagsoup/src/main/java/org A components/camel-tagsoup/src/main/java/org/apache A components/camel-tagsoup/src/main/java/org/apache/camel A components/camel-tagsoup/src/main/java/org/apache/camel/dataformat A components/camel-tagsoup/src/main/java/org/apache/camel/dataformat/tagsoup A components/camel-tagsoup/src/main/java/org/apache/camel/dataformat/tagsoup/TidyMarkupDataFormat.java A components/camel-tagsoup/src/main/resources A components/camel-tagsoup/src/main/resources/META-INF A components/camel-tagsoup/src/main/resources/META-INF/NOTICE.txt A components/camel-tagsoup/src/main/resources/META-INF/LICENSE.txt A components/camel-tagsoup/pom.xml M camel-core/src/main/java/org/apache/camel/builder/DataFormatClause.java M camel-core/src/main/java/org/apache/camel/model/dataformat/DataFormatsType.java A camel-core/src/main/java/org/apache/camel/model/dataformat/TidyMarkupDataFormat.java M pom.xml

14. **body:** Hi Ramon, I'm getting a failure in all tagsoup tests after applying this patch, mind taking a look? {code} Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.021 sec <<< FAILURE! testUnMarshalToStringOfXml(org.apache.camel.dataformat.tagsoup.TidyMarkupDataFormatAsStringTest) Time elapsed: 0.003 sec <<< ERROR! java.lang.IllegalArgumentException: TidyMarkupDataFormat only supports returning a String or a org.w3c.dom.Node object at org.apache.camel.model.dataformat.TidyMarkupDataFormat.<init>(TidyMarkupDataFormat.java:48) at org.apache.camel.builder.DataFormatClause.tidyMarkup(DataFormatClause.java:180) at org.apache.camel.dataformat.tagsoup.TidyMarkupDataFormatAsStringTest$1.configure(TidyMarkupDataFormatAsStringTest.java:70) {code} Also a handy tip is to run CheckStyle first so it picks up any formatting issues. You can add a profile to do this {code} mvn install -Psourcecheck {code}
    label: code-design
15. Reopening this so a patch to address Claus' suggestions can be attached.
16. great work Ramon! BTW the trickiest bit extending camel is the JAXB part for the XML version of things; it always bites you in the ass! I had a few hours banging my head against it. One thing that looks missing is when adding a class into org.apache.camel.model.* it typically needs to be added to the corresponding jaxb.index file in camel-core/src/main/resources. Also sometimes changing these classes leads to schema gen build failures in camel-spring - I had them yesterday! This looks great!
17. **body:** Ramon, thanks a lot for this great component. HTML is a mess to parse, so this come handy. A few review from me: - Consider using IllegalArgumentException for invalid configuration instead of CamelException (not normally used for this) - ObjectHelper.notNull(dataObjectType, "dataObjectType", this) should be added to unmarshal so we know it's set - javadoc state throws Exception but its not in the throws list - inputStream.close() we normally ignore with a debug/warn log - asNodeTidyMarkup when thrown the original exception the caused exception is missing, should be added as a 2nd parameter - javadoc for DataFormatClause the fluent builder methods is misleading for the one having a String as the type. - TidyMarkupDataFormat uses java assert. Please throw a IllegalArgumentException instead
    label: code-design
18. Ah silly me, it was a logic error :) I made this change to your patch: {code} - if (dataObjectType.isAssignableFrom(String.class) || dataObjectType.isAssignableFrom(Node.class)) { + if (!dataObjectType.isAssignableFrom(String.class) && !dataObjectType.isAssignableFrom(Node.class)) { throw new IllegalArgumentException("TidyMarkupDataFormat only supports returning a String or a org.w3c.dom.Node object"); } {code} I applied your patch in revision 725883. Thanks Ramon!