Chappy project

Contents

[2 Purpose of the project 2](#_Toc478034231)

[3 Project modules 2](#_Toc478034232)

[3.1 SERVICES 2](#_Toc478034233)

[3.1.1 REST Service 3](#_Toc478034234)

[3.1.2 HTTP Service (not yet implemented) 3](#_Toc478034235)

[3.2 INTERFACES 3](#_Toc478034236)

[3.2.1 Flows package 3](#_Toc478034237)

[3.2.2 Resources package (not yet implemented) 3](#_Toc478034238)

[3.2.3 Transformers package 3](#_Toc478034239)

[3.2.4 Services package 4](#_Toc478034240)

[3.2.5 Resources folder 4](#_Toc478034241)

[3.3 PROVIDERS 4](#_Toc478034242)

[3.3.1 FLOW.RUNNERS package 4](#_Toc478034243)

[3.3.2 REST.RESOURCES package 4](#_Toc478034244)

[3.3.3 SERVICES package 4](#_Toc478034245)

[3.4 FLOW 4](#_Toc478034246)

[3.4.1 TRANSFORMERS.RUNNERS package 5](#_Toc478034247)

[3.4.2 DYNAMICFLOWS packages 5](#_Toc478034248)

[3.4.3 STATICFLOW.RUNNERS package 5](#_Toc478034249)

[3.4.4 UNITESTS 5](#_Toc478034250)

[3.5 CONFIGURATIONS 5](#_Toc478034251)

[3.5.1 TRANSFORMERS package 5](#_Toc478034252)

[3.5.2 SYSTEM package 5](#_Toc478034253)

[3.6 TRANSFORMERS 5](#_Toc478034254)

[3.6.1 JSON package 5](#_Toc478034255)

[3.7 MAPPINGS 6](#_Toc478034256)

[3.7.1 XSLT package 6](#_Toc478034257)

[3.8 TESTS 6](#_Toc478034258)

[3.8.1 Manual tests 6](#_Toc478034259)

[3.8.2 UNITESTS suite 6](#_Toc478034260)

[3.9 UTILS 6](#_Toc478034261)

[3.9.1 STREAMS package 6](#_Toc478034262)

[3.9.2 REST package 6](#_Toc478034263)

[3.9.3 WRAPPERS package 6](#_Toc478034264)

# Purpose of the project

Chappy has been design as a testbed for long running servers.

The transformation server was choose as type of long running server. The transformation server will take request using REST.

Chappy wish to implement the following characteristics of the long running server:

* REST server using Jersey and Jetty.
* Predefined service for transformation using staxon and saxon.
* Hot-plugin of Transformation steps.
* Hot-plugin of Transformations step by user with returning a cookie which will be used to run the flow. (not implemented yet)
* Hot-plugin of Transformation steps with dependencies. (not implemented yet)
* Auto-upgrade of the transformation flows/step. (not implemented yet)
* Persistence of the upgrade. (not implemented yet)
* HTTP upgrade service. (not implemented yet)

Chappy uses the following libraries and technologies:

* Jetty for the embedded http server.
* Jersey for the REST server.
* Staxon for json2xml and xml2json transformations.
* Saxon for xml2xml mapping.
* Apache Digester for running a flow in one step.
* JAXB for data-binding.
* Maven for building.
* OBJECTWEB.ASM for changed java bytecode.
* Reflections library is used for package inspections.

# Project modules

Chappy use maven build and maven modules to isolate functionality.

## SERVICES

This module holds the services that are available from outside.

### REST Service

The rest service is used primary to send messages and retrieve back the transformed message.

The rest service is used secondary to update/patch modules and to post new transformers and flows (is not yet implemented).

#### Transformation services

This service is used to send a message and retrieve back the converted messages.

The transformation flow is defined by one or more steps defined into an xml. This flow could be executed using digester as fire and forget or using standard flow. The standard flow could be upgraded and defined by the customers using upgrade service.

The path to the transformation services are

* Base URL : /rest/transform
* Digester flow : /rest/transform/digesterFlow
* Flow : /rest/transform/flow
* Step of transformation (xml2json or json2xml using staxon) : /rest/transform/staxon

Those path are based on the package : chappy.services.servers.rest.resources

#### Upgrade service (not yet implemented)

This service is used to upgrade steps, flow and other systems.

### HTTP Service (not yet implemented)

The service is used primary to upgrade the server and services.

## INTERFACES

This module holds the interfaces or abstract classes that could be inherited or implemented by external tools/clients.

### Flows package

This holds in this moment two interfaces used by the system to run the flow :

* IFlowRunner which is used the run the flow.
* IFlowRunnerProvider which is used to identify and provide the required runner.

### Resources package (not yet implemented)

This holds the interface that is used to provide the REST resource classes.

This is used by resource auto-discovery service to register resource class into the Jersey server.

### Transformers package

This holds the interfaces for the steps that are used into the flow.

The AbstractStep is preferred to be inherited because it contains the disable feature.

If the custom step could not use inheritance then the implementation ITransformerStep could be used.

### Services package

This holds the IServiceServer interface which is used by every service/server.

### Resources folder

This holds the xsd schema used for validation of different configurations.

Flow.xsd is the schema definition of the flow configuration.

SystemConfiguration.xsd is the schema definition of the configuration of the system and also of services.

## PROVIDERS

This module holds the providers/factories for dynamic implementations that is used by the resources module. Those are singletons.

### FLOW.RUNNERS package

Factory/provider for a flow runner. It will give the instance of the actual implementation of the flow runner interface. This could be extended by custom flows to give the correct runner.

Available flows:

* DynamicFlowRunnerProvider provides a dynamic flow: in the first implementation it will provide a digester flow based on the path from the resource.
* StaticFlowRunnerProvider provides a dynamic flow: in the first implementation it will provide a static flow based on the path from the resource.
* SimpleFlowRunnerProvider provides a dynamic flow: in the first implementation it will provide a pure staxon transformation flow based on the path from the resource.

### REST.RESOURCES package

Factory/provider for resources used into the REST calls.

RestResourceProvider singleton is called to give REST resources based on package name.

TransformRestResouceProvider give the annotated resource classes from the transformation rest package.

### SERVICES package

Factory/provider for services instances like rest using jetty.

## FLOW

This module holds the actual runners of the flows and configurations for basic flows.

### TRANSFORMERS.RUNNERS package

This hold the runners for the flows.

Convention is the Class is the name of the runner on which is added the name Runner.

Default runners:

StaxonSimpleFlowRunner which run staxon transformations like xml2json and json2xml.

StaticFlowRunner is used to run flow using JAXB.

DigersterFlowRunner is used to run flow using Apache Digester 3.0.

### DYNAMICFLOWS packages

This package also contains the factory for the steps and parameters used by digester to create the flow.

### STATICFLOW.RUNNERS package

This packages contain the runners and configurations need to create and run the static flow (created using jaxb or other data binding package).

The configurations: FlowConfiguration, StepConfiguration, StepParameters and StepsConfiguration are annotated using JAXB. Those use databinding to transform the xml flow configuration received into data classes used by the flow runner.

### UNITESTS

Those tests are hold inside src/test/java and have tests for JAXB un-marshal operations on the configurations of the flows.

## CONFIGURATIONS

This module hold the configurations for transformers steps. Those will be un-marshal by JAXB or digester using field annotation or setter/getter.

### TRANSFORMERS package

This holds the configurations for saxon engine (it has a SaxonConfiguration as master configuration and input/output properties as ConfigurationProperties.

### SYSTEM package

This holds the system configuration, and services configurations.

## TRANSFORMERS

This hold the step used by transformers. The transformer step represent a 1:1 transformation of the input message like json2xml and xml2json.

### JSON package

This holds the staxon transformation step and the two steps xml2json an json2xml.

Those steps are used into the Simple Flow as single step for the staxon REST path.

Those steps could be used also in StaticFlow or DynamicFlow if they are defined into the configuration xml of the flow.

## MAPPINGS

This holds the mapping steps for engines.

A mapping flow is defined as a transformation of an input message into another input message (it could be different or same type of message.

### XSLT package

This hold the XSL/XSLT transformers based on saxon HE xslt engine.

The class which should be called is XslStep. The external xsl parameters could be send to xsl engine using query parameters.

## TESTS

This module holds the unitests and manual tests used to validate the chappy.

### Manual tests

The package manual.rest.transformers.test contains manual tests.

This test should be used after the server is started.

### UNITESTS suite

The packages like rest.transformers.test contains unitests on the specific protocol and for specific service (this is for send message to be transformed using REST protocol).

## UTILS

This module contains the utils classes need by chappy (classes are derived from from standard JAVA packages to adapt to the Holders) or just utility classes.

### STREAMS package

This package contains tools for easy manipulation of IO stream and resources

### REST package

This package contains classes to deal with streams on http.

The RestStreamingOutput is class that is used to send the response back on REST using streams.

### WRAPPERS package

This package is used by chappy to reuse buffers or streams in chain processing.

#### ByteArrayIOStream classes

ByteArrayInputStreamWrapper and ByteArrayOutputStreamWrapper are wrappers over ByteArrayInputStream and ByteArrayOutputStream to reuse the internal buffer.

#### StreamHolder class

This class is used by steps in transformers to send/retrieve the message.

It has internally a mechanism to transform the input stream into output stream as demand.

Internally the same buffer is used as input/output message and one stream holder is used to hold only one copy of the message.