Заглушка + Инструкция

Скачать original **jstub.rar**
fork: https://sbt-gitlab.ca.sbrf.ru/ESB/unistub

Инструкция к универсальной системе заглушек.

Решаемая проблема:

Заглушки служат для эмуляции автоматизированных систем. Применяются на этапах автоматизированного, нагрузочного, функционального тестирования. Позволяют формировать ответ, аналогичный ответу от реальной системы.

Способны получать сообщения из разных источников (HTTP, MQ, Socket) и отвечать на них (как отправителю, так и любому заданному получателю), предварительно проведя интеллектуальную предобработку ответа.

Возможно получение сообщения по HTTP и отправка ответа в MQ (Эмуляция DataPower)

Вся логика обработки описывается в файле конфигурации. Любое поведение может быть задано и изменено без необходимости править исходный код.

Взгляд сверху:

Заглушка представляет собой .jar файл, который при запуске в качестве параметра принимает .xml файл конфигурации.

Пример команды для запуска: **java -jar Stub.jar StubConfig.xml**

Для создания N различных заглушек нужен один Stub.jar и N файлов конфигурации.

Порядок работы заглушки

- Получение сообщения из внешнего источника (HTTP, MQ, Socket).
- Анализ входящего сообщения (определение типа операции, выбор шаблона для ответа).
- Предобработка входящего сообщения (декодирование и т.д.) (Preprocessors).
- Сохранение значимых данных из входящего сообщения (Extractors).
- Генерация дополнительных данных для ответа (Extractors).
- Извлечение данных из файлов CSV (PostExtractors).
- Замена необходимых полей в шаблоне ответа (Replacers).

- Предобработка ответа (кодирование, подписание, внешняя обработка и т.д.) (Postprocessors).
- Отправка ответа получателю(HTTP, MQ, Socket).

Структура файла конфигурации:

Структура конфигурационного файла

```
<?xml version="1.0" encoding="UTF-8"?>
<StubConfig>
        <GetIncomingFrom>MQ</GetIncomingFrom>
        <BypassReplyTo>false</BypassReplyTo>
        <Persistence>NOT_PERSISTENT</Persistence>
        <BypassDestination>false</BypassDestination>

                <!-- Описание параметров очередей, с которыми работаем -->
                <Queues>
                    <Queue>
                            <Label>InQ</Label>
                            <Server>127.0.0.1</Server>
                            <Port>8080</Port>
                            <Qmanager>MGR.NAME</Qmanager>
                            <Channel>SC.TEST</Channel>
                            <Queue>A.B.C</Queue>
                            <User></User>
                            <Password></Password>
                </Queue>

                            <!-- Определения выходных очередей -->
                    <Queue>
                            <Label>OutQ</Label>
                            <Server>...</Server>
                            <Port>...</Port>
                            <Qmanager>...</Qmanager>
                            <Channel>...</Channel>
                            <Queue>...</Queue>
                    </Queue>

                            <!-- Метка единственной входящей очереди -->
                    <StubInputQLabel>InQ</StubInputQLabel>
        </Queues>

                <!-- Блок обработки -->
        <Processing>

                            <!-- Путь к шаблонам ответов -->
                    <TemplatesPath>X:/A/b</TemplatesPath>

                            <!-- Извлечение данных из входящих сообщений -->
                    <IncomingOperationNameExtractors>
                        <Extractor>
                        </Extractor>
                    </IncomingOperationNameExtractors>
                    <Operations>

                                                <!-- Процессинг по умолчанию, параметры м.б. перезаписаны при определении
конкретных операций -->
                            <Operation Name="DEFAULT">

                                                        <!-- Временная задержка, "think time" -->
                                <Delay></Delay>

                                                        <!-- Выходная очередь -->
                                <Respond2Queue>OutQ</Respond2Queue>

                                                        <!-- Извлечение данных из входящих сообщений -->
                                <Extractors>
                                    <Extractor>
                                    </Extractor>
```

```
                                        </Extractors>

                                                                    <!-- Замена данных в шаблонах -->
                                        <Replacers>
                                            <Replacer>
                                            </Replacer>
                                        </Replacers>
                                    </Operation>

                                                                <!-- Описание списка операции и соответствующих шаблонов; Может содержать
также и процессинг -->
                                    <Operation Name="OperationRq" Template="template1.xml">
                                        <Respond2Queue>OutQ</Respond2Queue>
                                    </Operation>
                                </Operations>
        </Processing>
</StubConfig>
```

Файл конфигурации служит для описания логики обработки входящих сообщений и состоит из следующих блоков:

1. Выбор источника входящих сообщений

Указывает, из какого источника заглушка будет читать сообщения для обработки.

Для HTTP:

> *<GetIncomingFrom>HTTP</GetIncomingFrom>*
> *<Port>8084</Port>*

Обрабатываются все сообщения, отправленные на заданный порт по HTTP.

Для MQ:

> *<GetIncomingFrom>MQ</GetIncomingFrom>*
>
> *<Persistence>NOT_PERSISTENT</Persistence> - метод отправки сообщения, PERSISTENT - сообщения сохраняются, NOT_PERSISTENT - сообщения не сохраняются и могут быть потеряны во время передачи.*
>
> *<BypassReplyTo>false</BypassReplyTo> - записать JMSReplyTo запроса, в ответное сообщение.*
>
> *<BypassDestination>false</BypassDestination> - отправлять сообщения в пункт назначения, указанный в JMSReplyTo.*

Обрабатываются все сообщения из указанной очереди, параметры очереди указываются в блоке «Настройка параметров очередей»

Для Socket:

> *<GetIncomingFrom>Socket</GetIncomingFrom>*
>
> *<Port>8040</Port>*
>
> *<InputStreamEncodeType>EncodeToHexString</InputStreamEncodeType>*

2. Настройка параметров очередей

В случае, если логика заглушки подразумевает чтение или запись в очереди MQ, то параметры этих очередей должны быть заданы в этом блоке

> *<Queues>*
>
> *<Queue>*

```
        <Label>InQ</Label>
        <Server>10.66.127.109</Server>
        <Port>1415</Port>
        <Qmanager>M99.ESB.ADP.BO1</Qmanager>
        <Channel>SC.TEST</Channel>
        <Queue>ESB.ASBS.RESPONSE1</Queue>
        <User>lt_tester1</User>
        <Password>654321</Password>
      </Queue>
       <Queue>
            ….
      </Queue>
      <StubInputQLabel>InQ</StubInputQLabel>
    </Queues>
```

StubInputQLabel указывает, какая из описанных очередей должна быть использована в качестве источника для обработки сообщений (для случая, если **GetIncomingFrom** = MQ см. Выбор источника входящих сообщений).

3. Настройка обработки входящих сообщений

3.1. Путь к шаблонам

Заглушка отвечает на каждое входящее сообщение каким-либо ответом. Ответ формируется путем замены заданных полей в шаблоне. Необходимо указать путь, по которому доступны шаблоны ответов для данной заглушки (можно сетевой)

```
<TemplatesPath>D:\КСШ\COD\templates</TemplatesPath>
```

3.2. Загрузка файлов

Если вы используете PostExtractors, необходимо загрузить файлы в память:

```
<Files>
            <File id="file1" src="./datapools/" />
            <File id="file2" src="./datapools/" />
 </Files>
```

3.3. Определение типа операции

Для того, чтобы знать какой шаблон использовать для ответа, необходимо по входящему сообщению определить тип операции. Для этого из тела входящего сообщения, при помощи одного или нескольких **<Extractor>** (см. Extractors) получается строка с именем операции. Строка получается путем последовательной конкатенации строк, возвращаемых каждым из Extractor`ов.

```
    <IncomingOperationNameExtractors>
      <Extractor>
            …
```

```
        </Extractor>
        <Extractor>
            …
        </Extractor>
    </IncomingOperationNameExtractors>
```

Соответствие имени операции определенному шаблону описывается в пункте Настройка обработки для операций

3.4. Настройка обработчиков для операций

Каждая операция подразумевает использование своего шаблона, поэтому для каждого имени операции необходимо указать, какой шаблон используется.

```
<Operation Name="CreateInsuranceNf" Template="template1.xml">
…
</Operation>
```

3.4.1. Получатель ответа

В случае, если ответ на входящее сообщение должен быть отправлен в очередь получателю, то этот получатель может быть задан в блоке **Respond2Queue.** Блок принимает на вход значение **Label** очереди получателя (см. Настройка параметров очередей).

```
<Respond2Queue>OutQ</Respond2Queue>
```

В случае HTTP заглушки ответ автоматически получает и отправитель запроса, но отправка в очередь тоже возможна.

3.4.2. Задержка ответа

Задержка перед отправкой ответа задается, для эмуляции времени, необходимого реальной системе для выполнения заданного типа операции. Задается в милисекундах.

```
<Delay>
    <Min>150</Min>
    <Max>200</Max>
</Delay>
```

Существует возможность не останавливая работу заглушки менять параметры задержки, чтобы эмулировать отказ\замедление сети.

Для этого на порт, по которому слушает заглушка нужно отправить служебный запрос вида:

```
http://127.0.0.1:8080/Delay?from=100&to=200
```

3.4.3. Preprocessors

Используются для предварительной подготовки входящего сообщения к обработке.

Если требуется расширить функционал обработки тела сообщения, необходимо добавить следующий элемент:

External_jar

```
<Preprocessor>
    <Type>External_jar</Type>
    <Jar_Path>jar_path</Jar_Path> - путь к jar.
    <Class_name>class_name</Class_name> class в котором объявлен метод: public static String process(String body){}
</Preprocessor>
```

Если вам требуется декодировать текст в кодировке Base64 используете следующий препроцессор:

Decode

```
<Preprocessor>
   <Type>Decode</Type>
   <LB></LB> - левая граница тела сообщения
   <RB></RB> - правая граница тела сообщения
</Preprocessor>
```

Закодировать тело сообщения в шестнадцатеричное значение:

EncodeToHexString

```
<Preprocessor>
   <Type>EncodeToHexString</Type>
</Preprocessor>
```

3.4.4. Extractors

Используются для получения данных из входящего сообщения, а так же для генерации дополнительных данных, для подстановки в шаблон ответа.

Пример входящего сообщения:

```
<Incoming>
    <RqUID>#########RQUID#########</RqUID>
    <NS2:operation-name Value="OpNameValue">SrvSendNaturalPersonApplicationStateChangeMessage</NS2:operation-name>
    <RqTm>#########TIME#########</RqTm>
    <Status>
        <StatusCode>0</StatusCode>
        <StatusDesc>Ok</StatusDesc>
    </Status>
</Incoming>
```

Если нам необходимо получить значение тега <RqUID>,
для этого в конфигурационном файле заглушки нужно прописать соответствующий Extractor:

ByTagName

```
<Extractor>
    <Type>ByTagName</Type> -- получить значение, соответствующее содержанию тега
    <TagName>RqUID</TagName> -- с именем "RqUID"
    <ExtractedName>extractedRqUID</ExtractedName> -- сохранить это значение в переменную "extractedRqUID"
</Extractor>
```

ByXpath

```
<Extractor>
    <Type>ByXpath</Type> -- получить значение по строке Xpath
    <Xpath>/Incoming/RqUID</Xpath> -- строка Xpath
    <ExtractedName>extractedRqUID</ExtractedName> -- сохранить это значение в переменную "extractedRqUID"
</Extractor>
```

ByTagAttribute

```
<Extractor>
    <Type>ByTagAttribute</Type> -- получить значение, соответствующее содержанию атрибута тега из входящего
сообщения
    <TagName>NS2:operation-name</TagName> – у тега с именем "NS2:operation-name"
    <AttributeName>Value</AttributeName> -- получить значение атрибута "Value"
    <ExtractedName>extractedRqUID</ExtractedName> -- сохранить это значение в переменную "extractedRqUID"
</Extractor>
```

RootTag

```
<Extractor>

     <Type>RootTag</Type> – получить имя корневого тега из входящего сообщения

    <ExtractedName>RootName</ExtractedName> – сохранить это значение в переменную "RootName"

</Extractor>
```

ByBoundaries

```
    <Extractor>
        <Type>ByBoundaries</Type> – получить значение между двумя строками из входящего сообщения

        <LB>&#60;NS2:operation-name&#62;</LB> - левая граница

        <RB>&#60;/NS2:operation-name&#62;</RB> - правая граница

        <ExtractedName>Boundaries</ExtractedName> – сохранить это значение в переменную "Boundaries"
    </Extractor>
```

Спецсимволы должны быть заменены их числовыми кодами (ограничение xml):

Спецсимвол	Числовой код
<	&#60
>	&#62
"	&#34
&	&#38
'	&#39

RandSequence

```
<Extractor>
    <Type>RandSequence</Type> - последовательность символов
    <Prefix>BBMO-</Prefix> - фиксированное значение перед случайной последовательностью
    <Charset>0123456789</Charset> - набор символов из которых будет производится случайный выбор
    <Length>5</Length> - длина случайного набора символов
    <ExtractedName>requestID</ExtractedName>
</Extractor>
```

Примеры выполнения:

BBMO-95863

BBMO-70950

BBMO-56652

BBMO-12941

Date

```
<Extractor>
   <Type>Date</Type> - строка даты
   <DateFormat>yyyy/MM/dd HH:mm:ss</DateFormat> -формат строки
   <ShiftMin>120</ShiftMin> - смещение от текущего времени в минутах, может быть отрицательным
   <ExtractedName>requestID</ExtractedName>
</Extractor>
```

Пример выполнения:

2015/04/09 06:10:14

HardcodedString

Всегда возвращает заданное в теге <Value> значение

```
<Extractor>
   <Type>HardcodedString</Type>
   <Value>OperationName</Value>
   <ExtractedName>Hardcoded</ExtractedName>
</Extractor>
```

WholeMessage

Возвращает сообщение целиком, например для прокси функционала.

```
<Extractor>
   <Type>WholeMessage</Type>
   <ExtractedName>AllIncoming</ExtractedName>
</Extractor>
```

ReplyTo

```
<Extractor>
   <Type>ReplyTo</Type>
   <Part>Manager</Part>
   <ExtractedName>ReplyTo</ExtractedName>
</Extractor>
```

Если <Part>Manager</Part> то вернет “M99.ESB.MDM1”

Если <Part>Queue</Part> то вернет “ESB.MDM.RESPONSE”

Если ReplyTo пустой, то вернет “NA”

RegexpMatcher

```
<Extractor>
   <Type>RegexpMatcher</Type>
   <Regexp>StatusCode.[0-9]..StatusCode</Regexp>
   <ValueIfTrue>match</ValueIfTrue>
   <ValueIfFalse>notMatch</ValueIfFalse>
   <ExtractedName>regexp</ExtractedName>
</Extractor>
```

Проверяется, возвращает ли какой-либо результат регулярное выражение примененное к телу входящего сообщения.

Если результат выполнения не пустой – то экстрактор будет равен значению ValueIfTrue

Если ничего найти не удалось, то экстрактор будет равен значению ValueIfFalse

CorrelationId

```
<Extractor>
   <Type>CorrelationId</Type>
   <ExcludeID>true</ExcludeID>
   <ExtractedName>CorrelationID</ExtractedName>
</Extractor>
```

Допустим CorrelationId=ID:213131313100000000000

Если ExcludeID=true, то вернет “213131313100000000000”

Если ExcludeID=false, то вернет “ID:213131313100000000000”

MessageId

```
<Extractor>
   <Type> MessageId</Type>
   <ExtractedName> MessageId</ExtractedName>
</Extractor>
```

JMSProperty

```
<Extractor>
   <Type>JMSProperty</Type>
   <PropertyName>Name</PropertyName> - имя jms свойства.
```

```
    <ExtractedName>MessageId</ExtractedName>
</Extractor>
```

3.4.5. PostExtractors

Извлекают данные из файлов CSV.

Пример file.csv:

```
A,B,C
1,2,3
4,5,6
```

ByValueFromCache

```
<PostExtractor>
        <Type>ByValueFromCache</Type>
        <FileId>id</FileId> - id файла в элементе File
        <Delimiter>,</Delimiter > - разделитель
        <FindColumnName>A</ FindColumnName > - имя колонки в которой происходит поиск FindValue
        <FindValue>extractedIdClient</ FindValue> - значение которое хотим найти (ссылаемся на переменные Extractor-ов)
        <ExtractedColumnName>B</ExtractedColumnName> - колонка из которой извлекаем значение
        <ExtractedName>val</ExtractedName> - название переменной
</PostExtractor>
```

*если extractedIdClient = 1, тогда val = 2. (**extractedIdClient - переменная Extractor-а**)*

3.4.6. Replacers

Указывают, какое место шаблона должно быть заменено значением какого Extractor`а.

После того, как получены необходимые значения из входящего сообщения, их нужно подставить в шаблон.

Пример шаблона для ответа:

```
<DWLControl>
  <clientSystemName>  System  </clientSystemName>
  <clientTransactionName>  SPName  </clientTransactionName>
  <externalCorrelationId>  RsUID  </externalCorrelationId>
  <requesterLanguage>2200</requesterLanguage>
  <requesterLocale>ru</requesterLocale>
  <requesterName>  System  ":38:6901:166:1:</requesterName>
  <requestID>20130111134615964</requestID>
  <transactionCorrelatorId>1</transactionCorrelatorId>
  <ControlExtensionProperty name="Service">BaseSvcRq</ControlExtensionProperty>
</DWLControl>
```

ByTagName

```
<Replacer>
    <Type>ByTagName</Type> -- заменить значение тега в шаблоне
    <TagName>externalCorrelationId</TagName> -- имя тега, значение которого будет заменено
    <ExtractedName>extractedRqUID</ExtractedName> -- имя Extractora, значением которой будет заменено значение тега в
шаблоне
</Replacer>

было:  <externalCorrelationId>  RsUID  </externalCorrelationId>
стало:  <externalCorrelationId>########RQUID########></externalCorrelationId>
```

ByTagAttribute

```
<Replacer>

    <Type>ByTagAttribute</Type> -- заменить значение атрибута у тега в шаблоне
    <TagName>ControlExtensionProperty</TagName> -- у тега с именем "ControlExtensionProperty"
    <AttributeName>name</AttributeName> -- заменить значение его атрибута "name"
    <ExtractedName>extractedRqUID</ExtractedName> -- заменить значением extractedRqUID, из extractora
</Replacer>
```

ByBoundaries

```
    <Replacer>
        <Type>ByBoundaries</Type> – заменить значение между двумя строками в шаблоне

                <LB>&#60;NS2:operation-name&#62;</LB> - левая граница

                <RB>&#60;/NS2:operation-name&#62;</RB> - правая граница

        <ExtractedName>Boundaries</ExtractedName> – имя экстрактора
    </Replacer>
```

AllTemplate

Заменить весь шаблон целиком, например для прокси функционала

```
    <Replacer>
        <Type>AllTemplate</Type> – заменить значение между двумя строками в шаблоне

        <ExtractedName>Boundaries</ExtractedName> – имя экстрактора
    </Replacer>
```

3.4.7. Postprocessors

Используются для модификации сообщения перед отправкой получателю.

Если требуется расширить функционал обработки тела сообщения, необходимо добавить следующий элемент:

External_jar

```
<Postprocessor>

    <Type>External_jar</Type>

    <Jar_Path>jar_path</Jar_Path> - путь к jar.

    <Class_name>class_name</Class_name> class в котором объявлен метод: public static String process(String body){}

</Postprocessor>
```

Если вам требуется закодировать текст (кодировка Base64) используете следующий постпроцессор:

Encode

```
<Postprocessor>
    <Type>Encode</Type>
    <LB></LB> - левая граница тела сообщения
    <RB></RB> - правая граница тела сообщения
</Postprocessor>
```

Добавление JMS свойств в исходящее сообщение:

JMSProperty

```
<Postprocessor>
    <Type>JMSProperty</Type>
    <PropertyName></PropertyName> - название jms свойства
    <ExtractedName></ExtractedName> - название переменной, извлекаемой Extractor-ом (type: JMSProperty)
</Postprocessor>
```