Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

Владимирский государственный университет

Кафедра информационных систем

информационного менеджмента

**Методические указания к лабораторныой работе №2. SoapUI как инструмент эмуляции сервисов**

**по дисциплине «Тестирование программного обеспечения»**

Составители:

Котова Е.М.,

Конушин А.В.

Владимир 2012

**Лабораторная работа №2**

**SoapUI как инструмент эмуляции сервисов**

1. **Цель работы**

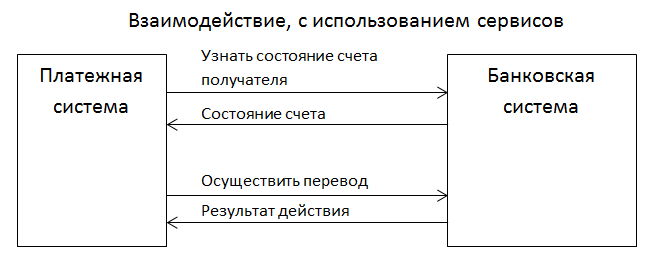
Изучить принципы работы со средством эмуляции работы веб-сервисов SoapUI.

1. **Общие сведения и ход работы**

Веб-сервис - идентифицируемая веб-адресом программная система со стандартизированными интерфейсами.

Как правило, системы не существуют сами по себе, они взаимодейтсвуют с другими системами, приложениями, базами данных для того, что бы получить или передать информацию.

Например, существует платежная система. Она предоставляет возможность осуществления платежей, переводов и другие услуги. И, например, для перевода денег на какой-нибудь счет, необходимо получить информацию о состоянии счета и возможности его пополнения, и выполнить сам перевод. Все эти задачи (получения информации) легко могут выполнить сервисы.



У каждого веб-сервиса существут адрес, он может принимать сообщения в формате xml, и отправлять ответы так же в формате xml. Взаимодействие осуществляется по протоколам, например http.

В тестировании ПС часто возникают такие ситуации, что необходимо протестировать только одну систему (например, только платежную или только банковскую систему) и доступа к другой системе нет. Для этого используются **эмуляторы** (или их еще называют **заглушками**), которые выполняют теже действия, что выполняла бы реальная система. Т.е. эмуляторы заменяют реальные системы во время тестирования.

Кроме заглушек существуют еще и драйверы. **Основное отличие драйвера от заглушки во взаимодействии:** драйвер вызывает тестируемую систему, тестируемая система вызывает заглушку.

**Драйвер –** компонент программного обеспечения или средство тестирования, которое заменяет компонент, обеспечивающий управление и/или вызов компонента или системы.

**Заглушка** – минимальная или специализированная реализация программного компонента. Использующаяся для подмены компонента, от которого зависит разработка или тестование другого компонента системы.

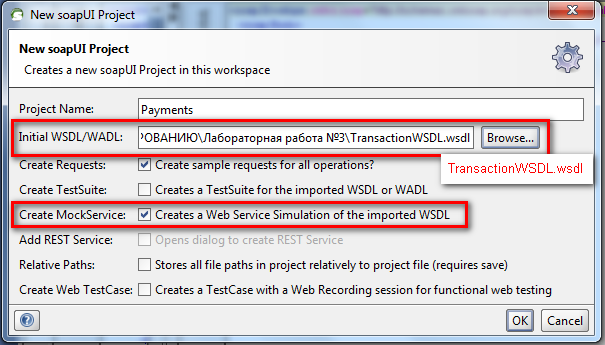


Эмулятор

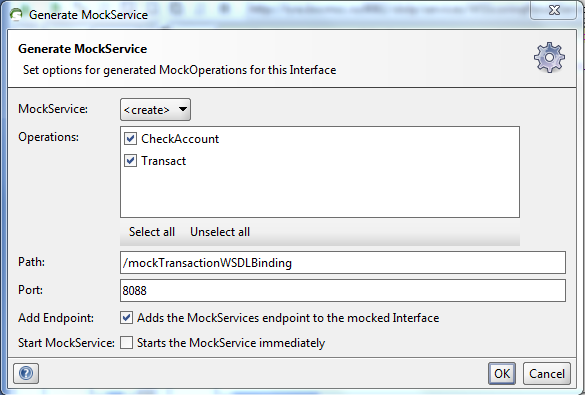
Программа SoapUI – одно из средств эмуляции ответов от веб-сервисов.

**Ход работы:**

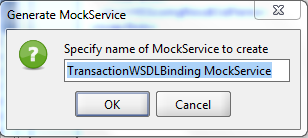
1. Установить программу SoapUI.
2. В открывшемся окне программы сделать: File – New SoapUI Project
3. Добавить wsdl сервиса, указать называние и проставить чек-бокс, что будет Mock-сервис. Нажать «ОК»:



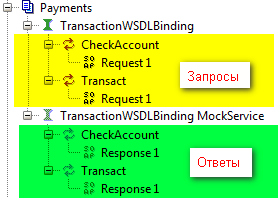
1. В следующем окне будут параметры сервиса. Путь к сервису и порт, на котором будет запущен. Нажать «ОК»:



1. Далее указывается название Mock-сервиса. Нажать «ОК».



1. Появились примеры запросов и Mock-сервис:



В сервисе две операции: проверить счет (CheckAccount) и провести перевод (Transact).

Операция CheckAccount, входные параметры:

* AccountNumber – номер счета;
* Name – имя клиента;
* Surname – фамилия клиента;
* Passport – номер паспорта.

Параметры ответа:

* AccountNumber – номер счета;
* Status – статус (может принимать значения: Close, Open, Blocked).

Операция Transact, входные параметры:

* AccountNumber
* Sum
* Target

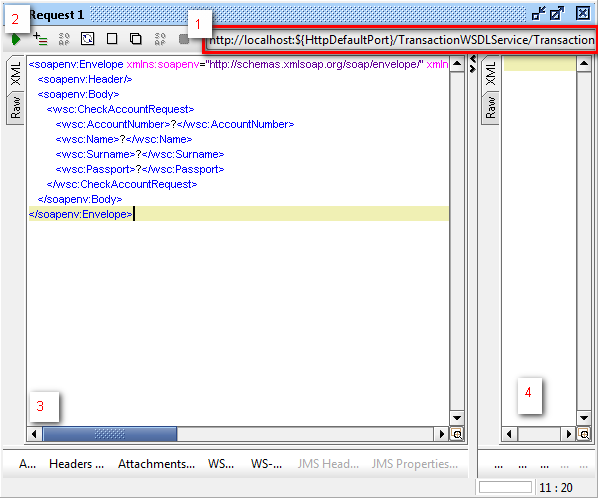
Параметры ответа:

* TransactionId – идентификационный номер транзакции;
* TransactionStatus – статус (Success).

Каждый сервис может вернуть ошибку состоящую из:

* ErrorCode – код ошибки;
* ErrorString – описание ошибки.

1. Если открыть Request1, то увидим окно:



1 – строка-ссылка на сервис (необходимо указать свой, т.е. он будет вторым из списка и содержать: http://<название комп.>:<порт>/<ссылка>, которые указывали в п. 4 при создании сервиса);

2 – кнопка для отправления запроса;

3 – окно запроса;

4 – окно ответа.

Можно изменить параметры запроса, например сделать его таким:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsc="http://bank/schema/WSCommonTypes">

<soapenv:Header/>

<soapenv:Body>

<wsc:CheckAccountRequest>

<wsc:AccountNumber>100500</wsc:AccountNumber>

<wsc:Name>Иванов</wsc:Name>

<wsc:Surname>Иван</wsc:Surname>

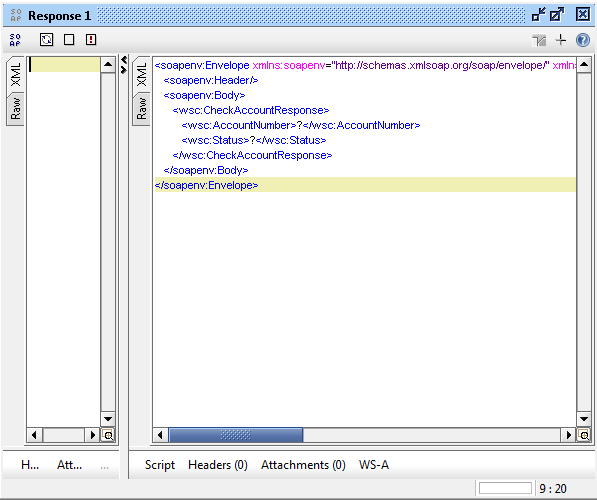
<wsc:Passport>17 10 020202</wsc:Passport>

</wsc:CheckAccountRequest>

</soapenv:Body>

</soapenv:Envelope>

1. Если открыть Response1, то увидим похожее окно, там указывается ответ сервиса:



Можно изменить ответ на следующий:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsc="http://bank/schema/WSCommonTypes">

<soapenv:Header/>

<soapenv:Body>

<wsc:CheckAccountResponse>

<wsc:AccountNumber>100500</wsc:AccountNumber>

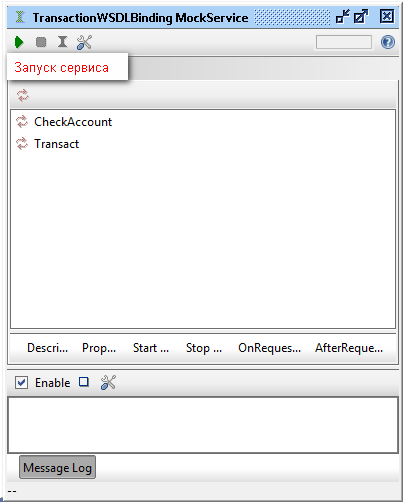
<wsc:Status>Open</wsc:Status>

</wsc:CheckAccountResponse>

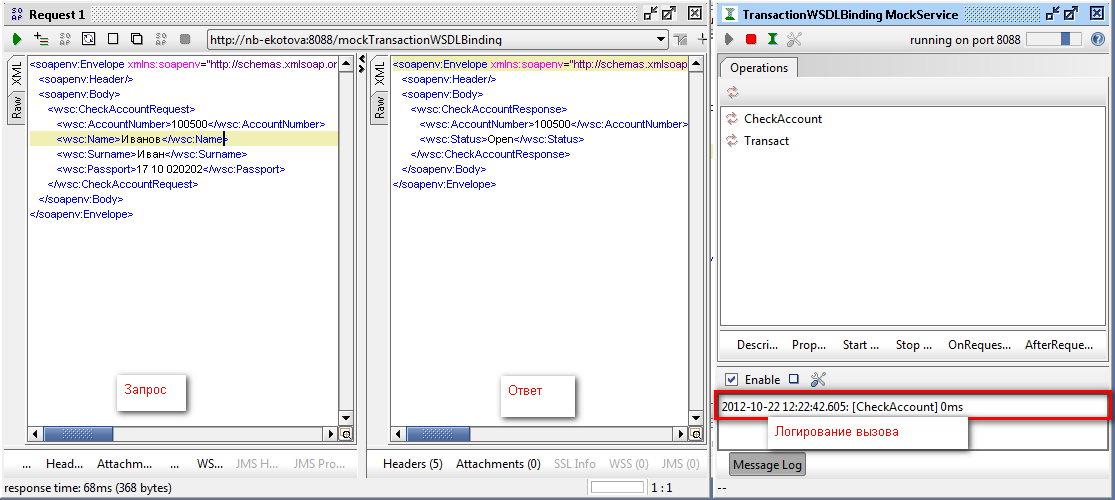
</soapenv:Body>

</soapenv:Envelope>

1. Откроем сам Mock-сервис и запустим его:



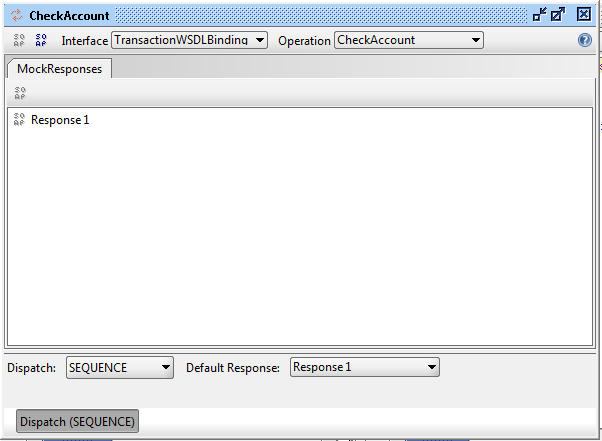
1. Если открыть запрос, который мы написали на п. 7, и попробовать отправить сообщение то получим следующее:



Таким образом, мы осуществили вызов эмулятора, и получили ответ.

Так же в тестировании есть необходимость, что бы на определенные запросы приходили определенные ответы. В SoapUI есть возможность с использованием языка Groovy обрабатывать те или иные запросы, и возвращать те или иные ответы.

Для этого необходимо открыть операцию веб-сервиса на редактирование:



Dispatch – указывает способ получения ответа. Можно выбрать Sequence, тогда все ответы будут выдаваться последовательно (если указано несколько ответов). Так же можно выбрать Script, тогда запрос будет обрабатываться скриптом.

DefaultResponse – ответ по умолчанию.

Пример скрипта для операции Transact:

//возвращает рандомное значение (для использования в ответе: <wsc:TransactionId>${TransactionId}</wsc:TransactionId>):

context.TransactionId = Math.random()

context.TransactionId = context.TransactionId.toString().substring(5,15)

//Определяем name space’s:

def groovyUtils = new com.eviware.soapui.support.GroovyUtils(context);

def holder = groovyUtils.getXmlHolder(mockRequest.getRequestContent());

holder.declareNamespace("wsc", "http://bank/schema/WSCommonTypes");

//Определяем переменную AccountNumber из входного сообщения:

def accountNumber = holder.getNodeValue("//wsc:TransactRequest/wsc:AccountNumber");

//Определяем ответ, в зависимости от AccountNumber (Error и Success – названия ответов)

if (accountNumber == "100500"){

return "Success";

}

if (accountNumber == "555555"){

return "Error";

}

Примеры ответов:

Success:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsc="http://bank/schema/WSCommonTypes">

<soapenv:Header/>

<soapenv:Body>

<wsc:TransactResponse>

<wsc:TransactionId>${TransactionId}</wsc:TransactionId>

<wsc:TransactionStatus>Success</wsc:TransactionStatus>

</wsc:TransactResponse>

</soapenv:Body>

</soapenv:Envelope>

Error:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsc="http://bank/schema/WSCommonTypes">

<soapenv:Header/>

<soapenv:Body>

<wsc:TransactResponse>

<wsc:ErrorCode>911</wsc:ErrorCode>

<wsc:ErrorString>Ошибка</wsc:ErrorString>

</wsc:TransactResponse>

</soapenv:Body>

</soapenv:Envelope>

**Задание:**

1. Для операции ChechAccount реализовать:

- В случае, если AccountNumber = 999999999, возвращать Close;

- В случае, если AccountNumber = 777777777, возвращать Blocked;

- В случае, если AccountNumber = 666666666, возвращать ошибочный ответ.

- Для других значений AccountNumber возвращать Open;

- В AccountNumber всегда возвращать входное значение.

1. **Содержание отчета по лабораторной работе**
2. Цель работы
3. Описание выбранной операции
4. Код программы, скриншоты (если требуются)
5. Выводы по лабораторной работе