

# SessionMan решает проблему сохранения состояний в вебсервисах

Вячеслав Яковенко, Дмитрий Руденко (webtair.com)

Вебсервисы уже давно стали настолько популярными, что их поддержка вошла в Java SE 6. Сейчас уже трудно удивить кого-то гетерогенной архитектурой приложения, в которой серверная часть реализована на Java, а клиентская на .Net или наоборот. Веб сервисы обладают огромным списком преимуществ и не меньшим списком недостатков, устранение одного из которых и является целью данной статьи.

## 1. Проблема

*"By design, Web services are said to be stateless..."* — эта фраза говорит о многом! Это значит, что используя веб сервисы, не составит труда передать на сервер код города и в ответ получить прогноз погоды на неделю. Но, для того чтобы организовать систему клиент-банк с использованием [Axis2](#)\*, придется потрудиться т.к. такая система предполагает процедуру авторизации и поддержку сессии в течении некоторого периода времени.

### Замечание:

\* Axis2 — второе поколение наиболее популярного движка для веб сервисов, разрабатываемого под эгидой Apache Software Foundation.

## 2. Варианты решения проблемы для Axis2

Мои поиски решения этой проблемы привели к одной статье [1], в которой ребята из

## *SessionMan решает проблему сохранения состояний в вебсервисах*

IBM предлагают свое решение. Его смысл сводится к тому, что в сервис-класс, сгенерированный Axis2 добавляется статическое поле (HashMap), который и является хранилищем информации о сессии пользователя, а wsdl сервиса преобразуется таким образом, что появляется две дополнительных операции login и logout.

Единственным недостатком или скорее ограничением в приведенном примере [1], является то, что клиент должен явно выполнять операцию logout, в противном случае информация о сессии так и останется дожидаться перезагрузки сервера приложений, а HashMap станет источником утечки памяти и проблем в производительности.

Решением проблемы также может стать Open Source библиотека [SessionMan](#), предоставляющая базовый набор функций для работы с сессиями в любом Java-приложении и выполняющая периодическую сборку мусора, состоящую в удалении устаревших сессий.

Для рассмотрения примера использования [SessionMan](#), нам понадобятся [TomCat](#) и [Axis2](#). Установите [TomCat](#), [Axis2](#) и проверьте их работоспособность, воспользовавшись документацией к каждому из продуктов и мы продолжим наше повествование.

### **3. WSDL — каркас любого вебсервиса**

Мы начнем нашу работу с того, что создадим [wsdl-документ](#), описывающий наш будущий вебсервис:

```
<portType name="SessionManExamplePort">
  <operation name="login">
    <input name="loginOpIn" message="tns:loginSoapMessage" />
    <output name="loginOpOut" message="tns:sessionSoapMessage" />
  </operation>
  <operation name="disconnect">
    <input name="disconnectOpIn" message="tns:sessionSoapMessage"/>
    <output name="disconnectOpOut" message="tns:isOkSoapMessage" />
  </operation>
  <operation name="ping">
    <input name="pingOpIn" message="tns:sessionSoapMessage" />
    <output name="pingOpOut" message="tns:pingSoapMessage" />
  </operation>
</portType>
```

## SessionMan решает проблему сохранения состояний в вебсервисах

```
</operation>  
</portType>
```

Как видно из фрагмента [wsdl](#) — наш вебсервис будет содержать три операции: login, disconnect\*\* и ping. Первая операция принимает SOAP конверт в который упакованы имя пользователя и его пароль:

### Замечание:

\*\* имя метода выбрано не случайно, т.к. в одной из версий Axis2 v.1.0 были ошибки, и операция logout обрабатывалась некорректно.

```
<xsd:element name="UserAuthElement">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="login" type="xsd:string"  
        minOccurs="1" maxOccurs="1"/>  
      <xsd:element name="password" type="xsd:string"  
        minOccurs="1" maxOccurs="1" />  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

Оставшиеся же две принимают только SOAP конверт содержащий идентификатор сессии, что с одной стороны, необходимо для проверки актуальности сессии перед выполнением любой операции, и с другой — вполне достаточно для объяснения работы с библиотекой [SessionMan](#).

После того, как [wsdl-документ](#) создан, мы запускаем утилиту wsdl2java, прилагающуюся к Axis2 и дожидаемся окончания генерации всех необходимых классов.

Полнофункциональный пример, использующий библиотеку SessionMan, совместно с Axis2, доступен из SVN репозитория: <http://sessionman.googlecode.com/svn/trunk/sessionman-example>.

Инструкция по сборке проекта для работы в среде Eclipse 3.2:  
<http://sessionman.googlecode.com/svn/trunk/sessionman-example/readme.txt>.

## 4. SessionMan выходит на арену

Результатом работы `wSDL2Java` является целый пакет классов, среди которых `SessionManExampleServiceSkeleton` — сердце нашего будущего сервиса. Для дальнейшей работы мы унаследуем этот класс и создадим свой [SessionManExampleService](#).

Подключаем `SessionMan`:

```
import com.webtair.session.*;
import com.webtair.session.artifact.*;
import com.webtair.sessionman.ws.types.*;
```

Добавляем контроллер сессий к сервис классу:

```
public class SessionManExampleService extends
    SessionManExampleServiceSkeleton {

    ...
    /** SessionMan instance — main goal of this example */
    private static SessionMan<SimpleSessionInfo> sessions
        = new SessionMan<SimpleSessionInfo>();
```

Здесь необходимы пояснения:

1. Класс [com.webtair.session.SessionMan](#) имеет два конструктора: `public SessionMan(ConfigBean config)` и `public SessionMan()` — второй создает конфигурацию по умолчанию со временем жизни сессии 30мин и периодичностью очистки устаревших сессий 10мин. С помощью первого конструктора пользователю предоставляется возможность управлять этими параметрами, а так же определить стартовую размерность таблицы для сессий, передав, соответствующим образом настроенный,

## *SessionMan решает проблему сохранения состояний в вебсервисах*

[com.webtair.session.config.ConfigBean](#)

2. В качестве хранилища вашей информации может использоваться любой класс, реализующий интерфейс [com.webtair.session.artifact.SessionInfo](#), для базовых операций с сессией библиотека предоставляет класс [SimpleSessionInfo](#).

Таким образом, если Вам необходимо создать хранилище для, скажем состояния корзины, достаточно будет реализовать интерфейс SessionInfo с соответствующей функциональностью.

Создаем сессию и возвращем ее клиенту вебсервиса:

```
/**
 * Method that allow for client to login under this ws
 * @return IdSessionDocument – soap envelope with session id;
 */
@Override
public IdSessionDocument login
    (UserAuthElementDocument userAuthElementDocument) {
    /* default login and pass for examples only. At this place you
     * must recieve real pass from the real storage like RDBMS or
     * config-file or somthing else.
     */
    String defaultLogin = "sessionman";
    String defaultpass = "example";

    UserAuthElement userAuthElement =
        userAuthElementDocument.getUserAuthElement();

    if (userAuthElement.getLogin().equals(defaultLogin)
        && userAuthElement.getPassword().equals(defaultpass)) {

        // request for id session from SessionMan lib
        idSession =
            sessions.putSessionInfo(userAuthElement.getLogin(),
            new SimpleSessionInfo());
    }
    ...
}
```

```
}
```

Ключевым выражением в процедуре получения идентификатора является метод `putSessionInfo()`, "вкладывающий" в `HashMap` контроллера `SessionInfo` и возвращающий уникальный идентификатор сессии.

Проверяем актуальность сессии:

Вызов метода `validateSession(String idSession)`, позволяет не только проверить "жива" ли сессия, но и автоматически обновляет время ее последнего использования, что приводит к ее автоматическому пролонгированию на время указанное в `ConfigBean` (30 мин по умолчанию).

Вызов метода `isSessionExpired(String idSession)`, позволяет просто определить завершилось ли время жизни сессии, не обновляя ее последнего использования.

## 5. LogOut и "чистка" сессий

Если пользователь не выполнил явной операции `disconnect (logout)`, чисткой `HashMap` займется фоновый поток, который проснется через интервал времени, определенный в `ConfigBean`.

## 6. Заключительное слово

Полнофункциональный пример, использующий библиотеку `SessionMan`, совместно с `Axis2`, доступен из `SVN` репозитория: <http://sessionman.googlecode.com/svn/trunk/sessionman-example>.

## 7. Ресурсы

1. Английская версия этой статьи: [http://www.webtair.com/articles/sessionman\\_en.html](http://www.webtair.com/articles/sessionman_en.html)
2. Online banking with Apache Geronimo and Axis2:  
<http://www-128.ibm.com/developerworks/edu/os-dw-os-ag-onbank1.html>
3. WebTair SessionMan library: <http://www.webtair.com/sessionman>

*SessionMan* решает проблему сохранения состояний в вебсервисах

4. Example for SessionMan and Axis2: <http://sessionman.googlecode.com/svn/trunk/sessionman-example>
5. SessionMan online API JavaDoc: <http://www.webtair.com/sessionman/api/index.html>.