Metadata Support for Transactional Web Services

Maciej Machulak

Java Developer's Day (JDD 2007)
Kraków, Poland
26th October 2007
Agenda

- Introduction
  - Web Services and Transactions 101
- Motivation for metadata support
- Business Activity Framework
  - API
  - Design
  - Implementation
  - Building applications
- Future work
- Summary
- Questions and Discussion
Agenda

- Introduction
  - Web Services and Transactions 101
- Motivation for metadata support
- Business Activity Framework
  - API
  - Design
  - Implementation
  - Building applications
- Future work
- Summary
- Questions and Discussion
Web Services and Transactions 101

- Web Service environment encourages development of applications that would integrate different businesses.
- Such applications rely on complex Business-to-Business interactions that involve many parties, span many different organisations, and last for hours, days or even months.
- Web Services-based transactions differ from traditional transactions in that they execute over long periods; isolation levels of such transactions need to be relaxed.
- Atomic (ACID) transactions by themselves are not adequate for structuring long-lived applications (but can be used in WS environment - WS-AtomicTransaction).
- Web Services-based transactions are based on extended transaction models.
WS-Business Activity

- Defines extended transaction model – **Business Activity** (BA)
- BA designed specifically for **long-duration interactions**, where exclusively locking resources is impossible or impractical (B2B)
- Services are requested to do some work, and where those services have the ability to undo any work, they inform the BA about it
- If the Business Activity is to cancel its work then services execute their undo behaviour (**compensation** – forward recovery)
- Consistency is maintained by compensation actions (those must be explicitly provided by business programmers)
- Consensus within a single Business Activity is achieved through **agreement protocol**
WS-Business Activity (2)

- Agreement protocols:
  - Business Agreement with Participant Completion
  - Business Agreement with Coordinator Completion
Agenda

- Introduction
  - Web Services and Transactions 101

- Motivation for metadata support

- Business Activity Framework
  - API
  - Design
  - Implementation
  - Building applications

- Future work

- Summary

- Questions and Discussion
Problem statement

- In WS-AtomicTransaction business programmers are often not concerned with any transaction related code.

- Web Service methods may have required transactional behaviour specified using @TransactionAttribute annotation (e.g. @TransactionAttribute(REQUIRED)).

- Existing solutions (e.g. TxBridge – see JavaOne 2007) provide all necessary mechanisms (transaction termination) during runtime – WS-AT context is mapped to a local JTA context.

- What about WS-BusinessActivity?
Problem statement (2)

- Development of transaction-aware Web Services with accordance to Business Activity model is quite complex and requires substantial programming effort.
- Developers need to provide necessary transaction-related code required to expose Web Services as Business Activity tasks.
- This results in transaction-related code being mixed with the business logic.
- Compensation actions are application-specific – need to be provided explicitly by developers.
- Most commonly application has to be developed according to a transaction-related design pattern: core business logic, participant, manager.
Objectives

- Ease the task of writing transaction-aware Web Services with accordance to the Business Activity extended transaction model (achieve the same, or similar, simplicity as that in typical ACID transactions)

- Provide separation of concerns and release programmers from the necessity of developing any transaction-related code

- Programmers should be able to declaratively specify all transactional requirements of their Web Services (using metadata – e.g. annotations)

- All necessary transaction processing should be provided basing upon those requirements in a transparent way and should be applied at runtime
Agenda

- Introduction
  - Web Services and Transactions 101
- Motivation for metadata support
- Business Activity Framework
  - API
  - Design
  - Implementation
  - Building applications
- Future work
- Summary
- Questions and Discussion
Business Activity Framework

- Support for transaction-aware Web Services participating in Business Activity type extended transactions

- Transactional requirements specified using annotation-based API
  - Web Services can be wired up with their completion and compensation actions with few simple declarative statements
  - the application code preserves its elegance and simplicity (business logic is separated from any transaction-related code)

- Framework provides implementation of the "manager" and "participant" – developer is only concerned with the business logic

- Framework provides transparent transaction management
  - automated execution of completion and compensation actions as and when necessary (mechanisms are dynamically applied at runtime)

- Flexible management of data required by completion or compensation actions
Business Activity Framework – API

- Clean and intuitive annotation-based API - keeps with the current trend in enterprise environments – the use of annotations for declaratively configuring both Web Services and transactional behaviour is widespread in both Java and .NET

- \texttt{@BAService} - marks a class as the one that contains Web Service methods exposed as BA tasks; specifies common completion-related and compensation-related information;

- \texttt{@BAMethod} - specifies the Web Service in terms of its state management (if it modifies any data or not) and the agreement protocol the service wants to participate in;

- \texttt{@BACompletedBy} - specifies the completion action and the type of completion for a single Web Service method;

- \texttt{@BACompensatedBy} - specifies the compensation action and the type of compensation for a single Web Service method;
Business Activity Framework – API (2)

- @BAParam - annotates the service’s parameter so that it can be processed by the data management mechanism and used when executing completion or compensation action;

- @BAResult - annotates the service’s return value to be processed by the data management mechanism and used when executing completion or compensation action;

- @BADataManagement - marks manager object to enable transparent dependency injection.

- Leverages programmer familiarity with EJB3 and JSR-181 – reduces the learning curve for new users of the framework

- All annotations have reasonable default values – in simplest case the programmer may mark a Web Service method with @BAMethod annotation
Additional component – **Data Manager** can be used if more flexible data management is required – this component is transparently injected by the framework.

Data Manager assists the business programmer by making data from the original method call available to the corresponding completion or compensation method, facilitating e.g. commitment of previously conducted work or restoration of a prior state:

- **put(id, Object)** – storing arbitrary data (used within transactional Web Services)
- **get(id)** – getting arbitrary data (used within completion or compensation actions)
BA Framework – API – Example

- Data Manager

```java
@BADataManagement
DataManager dm;
```

- Transactional Web Service

```java
@WebMethod
@BAMethod(agreement=AgreementType.PARTICIPANT_COMPLETION)
@BACompensatedBy(value="cancelRoom", type=DataMatch.CUSTOM)
@BAResult("orderId")
public Integer bookRoom(@BAParam("username")String user,
                        @BAParam("password")String pass,
                        Integer roomNumber)
{
    // ... business logic
    dm.put("refund",refundValue);
    // ... business logic
    return reservationNumber;
}
```
Compensation action

```java
@WebMethod
public void cancelRoom(@BAParam("username") String user,
                        @BAParam("password") String pass,
                        @BAParam("orderID") Integer orderID)
{
    ... // business logic
    Integer refund = (Integer) dm.get("refund");
    if (refund != null)
    {
        // ... business logic
    }

    ... // business logic
```
Execution mode - EJB

```java
@WebService(name="Hotel")
@BAService(serviceClass=HotelBAImpl.class,
            ejbInterface=HotelBA.class,
            jndiName="bademo/HotelBAImpl/remote")
public class HotelImpl implements Hotel {
    // ... class implementation
}
```

Execution method - WS

```java
@WebService(name="Hotel")
@BAService(
    wsdl="http://host:8080/hotel/HotelService?wsdl",
    serviceName="HotelService",
    namespace="http://hotel.service.namespace",
    mode=ExecutionMode.DII)
public class HotelImpl implements Hotel {
    // ... class implementation
}
```
Execution mode - EJB

```java
@WebMethod
@BAMethod(AgreementType.COORDINATOR_COMPLETION,
            MethodType.MODIFY)
@BACompletedBy(value="checkout",
                ejbInterface=HotelValidator.class,
                jndiName="baremote/HotelValidatorImpl/remote",
                providerURL="jnp://192.168.1.154:1099")
@BACompensatedBy(value="cancelRoom",
                  type=DataMatchType.CUSTOM)
@BAResult("orderID")
public Integer bookRoom(@BAParam("username") String user,
                        @BAParam("password") String pass,
                        Integer roomNumber)
```
BA Framework – API - Demo

Method's class
com.jboss.ba.demo.service.HotelImpl

Method's signature
@WebMethod
@BAParam(AgreementType, PARTICIPANT_COMPLETION, MethodType, NULLIFY)
public Integer bookRoom(String username, String password, Integer roomNumber)
throws HotelCustomException, RoomAlreadyOccupiedException

Service description
This service books a single hotel room. It must be executed within a Business Activity. It also remembers
a refund value so that if the booking is cancelled within a scope of a Business Activity then the customer
does not lose all the money.

This service takes following parameters:
- Username
- Password
- Room Number

This service returns:
- Reservation number

It throws following exceptions:
- HotelCustomException
- RoomAlreadyOccupiedException

HotelCustomException is thrown if either the user cannot log in (username, password or both are
incorrect) or the room number is incorrect (room does not exist). If the user wants to book a room that is
already occupied then RoomAlreadyOccupiedException is thrown.

Agreement Protocol
Business Agreement With Participant Completion

Compensation action
cancelRoom()
Business Activity Framework – design

- Framework intercepts calls to transactional Web Services, such calls are subject to transaction processing mechanisms
- Calls are intercepted after being mapped from SOAP messages to actual calls on objects (AOP-based call interception)
- Transaction processing is based on requirements specified by the Web Service
Business Activity Framework – design

- Framework intercepts calls to transactional Web Services, such calls are subject to transaction processing mechanisms.
- Calls are intercepted after being mapped from SOAP messages to actual calls on objects (AOP-based call interception).
- Transaction processing is based on requirements specified by the Web Service.
Business Activity Framework

- Each intercepted call to a transactional Web Service is uniquely identified
- Calls are associated with participants – those participants are automatically enlisted to play role in a transaction
- Each participant stores data related to Web Service calls – such data may be later used during completion or compensation
- Participants are responsible for responding to transaction control messages (e.g. `complete()`, `close()`, `compensate()`)
- Framework manages execution of completion and compensation actions as and when necessary
BA Framework – transaction management

- Framework manages tasks
  - task: original invocation (+ completion action)
    (+ compensation action)

- 4-step processing:
  - BA Task Management
  - BA Completion Management
  - BA Compensation Management
  - BA Close Management
BA Task Management
BA Completion Management

1. handleCompletion()
2. completeWork()
3. execute()
4. get()

some additional data

complete() →

5. completed()
BA Compensation Management

1. handleCompensation()
2. compensateTask()
3. execute()
4. get()
5. compensated()
BA Close Management

Diagram:
- Participant
- BA Manager
- close() from Participant to CloseTransaction()
- CloseTransaction() to destroy
BA Framework - Implementation

- Built on top of the JBoss XML Transaction Service (JBoss XTS) component of the JBoss Transaction Service
- Call interception based on the JBoss AOP Framework – only one lightweight component (easily exchangeable)
- Dynamic execution of completion and compensation actions – Java Reflection API
BA Framework – Web Services

- Business Activity framework is divided into two parts:
  - Client's library *(baframework-client.jar)* – only API, included in the CLASSPATH of the business application
  - Core transaction mechanisms *(baframework.aop)* – API + all necessary transaction-related mechanisms – may be bundled with the application or deployed as a standalone component

- Web Services must register a handler that will extract transaction context from the SOAP header space (e.g. JaxWSHeaderContextProcessor provided by JBoss XTS)

- Web Service call interception is provided using the AOP concept – business programmers may either perform the AOP instrumentation by themselves or use an AOP deployer service (e.g. that bundled with the JBoss Application Server)
BA Framework – Web Services (2)

- jaxws-handlers-server.xml

```xml
<handler-chains xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    javaee_web_services_1_2.xsd">
    <handler-chain>
        <protocol-bindings>##SOAP11_HTTP</protocol-bindings>
        <handler>
            <handler-name>WebServicesTxContextHandler</handler-name>
            <handler-class>
                com.arjuna.mw.wst.service.JaxWSHeaderContextProcessor
            </handler-class>
        </handler>
    </handler-chain>
</handler-chains>
```

- JAX-WS handlers

```java
@WebService(serviceName="HotelService", targetNamespace="http://www.nmspc.com")
@SOAPBinding(style = SOAPBinding.Style.RPC)
@HandlerChain(file = "jaxws-handlers-server.xml")
public class HotelImpl implements Hotel {
    // ... class implementation
}
Including the framework in the EAR file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<application>
  <!-- omitted elements -->

  <module>
    <java>baframework.aop</java>
  </module>

</application>
```
Agenda

- Introduction
  - Web Services and Transactions 101
- Motivation for metadata support
- Business Activity Framework
  - API
  - Design
  - Implementation
  - Building applications
- Future work
- Summary
- Questions and Discussion
Future work

- Full support for transaction messages
  - support for coordinator’s close and cancel messages (@BAClosedBy and @BACanceledBy annotations)

- Dynamic Execution of Web Services
  - completion and compensation actions accessed as Web Services

- Exposing existing components as Transactional Web Services
  - transactional-requirements specified in XML configuration file

- Completion and Compensation chains
  - programmers specify multiple possible completion/compensation actions – framework executes only a single, currently available one

- Support for crash recovery
  - logging registered completion and compensation actions; making managed data serializable
Agenda

- Introduction
  - Web Services and Transactions 101
- Motivation for metadata support
- Business Activity Framework
  - API
  - Design
  - Implementation
  - Building applications
- Future work
- Summary
- Questions and Discussion
Summary

- Business Activity Framework provides full support for transaction management of Web Services exposed as Business Activity tasks (separation of concerns, declaratively specified transactional requirements, transparency)

- Released as an open-source project under GNU General Public License 2.0
  - [http://labs.jboss.com/jbosstm/baframework](http://labs.jboss.com/jbosstm/baframework) (official website)
  - [http://jira.jboss.com/jira/browse/JBTM](http://jira.jboss.com/jira/browse/JBTM) (bug tracking)

- Three public releases: 0.1.0GA, 0.2.0GA and 0.2.1GA: framework, demo application, documentation (Transactional Web Services Programmer's API)

- Framework scheduled to be included in the core JBoss Transaction Service component (programmers will not need to bundle framework with their applications); annotations to be reused with other technologies
Summary (2)

- Business Activity Framework's API to influence the JSR-156 (Java API for XML Transactions)


- „Metadata Support for Transactional Web Services“ short research paper written with Jonathan Halliday and Mark Little; paper presented at the 2007 Middleware for Web Services Workshop (at the 11th IEEE International EDOC Conference in Annapolis, Maryland, USA).
Questions and Discussion

???
Thank You

Maciej Machulak
mmachulak@redhat.com