Java is a trademark of Sun Microsystems, Inc.

JavaOne℠

RESTful Transaction Systems

Mark Little (JBoss CTO)
Michael Musgrove (JBoss Tx)
JBoss by Red Hat

Presentation Goals

- Understand how to ensure consistency and reliability within REST-based systems
- How to coordinate changes to services running on different servers

Overview

- Web services and transactions
- REST and transactions
- Protocol description and demonstration
- Future work
- Summary

Overview

- **Web services and transactions**
- REST and transactions
- Protocol description and demonstration
- Future work
- Summary

Atomic Transactions

- Scoping mechanism that provides “all-or-nothing” semantics.
- Enables shared resources to be protected from concurrent use
- ACID properties
  - Atomic
  - Consistent
  - Isolated
  - Durable

WS-Transaction

- In the past, making traditional transaction systems talk to one another was a holy grail
  - Rarely achieved
  - Even JTS implementations didn't interoperate, despite common standard!
  - Web services provide opportunity to leverage unparalleled interoperability
  - Connect existing transactional systems at backbone of enterprise applications

B2B Interactions

- Business to business interactions may be complex
  - Involving many parties;
  - Spanning many different organizations;
  - Potentially lasting for hours or days
- Cannot afford to lock resources on behalf of an individual indefinitely
- May need to undo a subset of work

WS-TX

- WS-Coordination
  - Context propagation
  - Participant coordination
- WS-AtomicTransaction
  - ACID Web service transactions
  - Built upon WS-Coordination
- WS-BusinessActivity
  - Extended Web service transactions
  - Built upon WS-Coordination

Overview

- Web services and transactions
- **REST and transactions**
- Protocol description and demonstration
- Future work
- Summary

What is REST

- Architectural style
- Client server
- Stateless interactions
- Uniform interface
- Use of representations

Characteristics and Benefits

- Simplifies component development
- Simplifies component interactions
- No need for session management
- Exchange of representations using standard operations
- Complexity is pushed into the representations
- Messages are self-descriptive
- ...

Few standards in the REST space

- JAX-RS
  - Java language support for building REST apps
  - Annotation-based API for defining resources
  - A run-time for mapping HTTP requests to Java™ methods
  - Many implementations are available (RESTeasy)
- RFC 2616 and 3986 (HTTP and URI)
- **No standards for ensuring consistency in the presence of failures or protecting against concurrent updates across Web servers**

Possible Transaction Models for REST

- Atomic transactions (c.f. XA/JTA)
  - Status:- specified and implemented
    - https://svn.jboss.org/repos/labs/labs/jbosstm/workspace/resttx
  - Issues - availability, scalability, performance, latency
- Forward Compensation Based (c.f. WS-BP)
  - Status:- specified
    - https://svn.jboss.org/repos/labs/labs/jbosstm/workspace/resttx/docs/jfdi-spec.txt

Overview

- Web services and transactions
- REST and transactions
- **Protocol description and demonstration**
- Future work
- Summary

Resources and Components

- Transaction coordinator - responsible for:
  - Transaction URL
  - Transaction creation and completion
  - Participant enlistment
  - Coordinating participants across Web servers
- Recovery coordinator – drives recovery after failure of:
  - Coordinator
  - Participant

Resources and Components (2)

- Participant, responsible for
  - ensuring that changes to a resource can be driven through a 2PC protocol, that changes to resources are recoverable in the presence of failures and that changes are durable and isolated from other changes
  - exposing a URL for each transaction branch
- Client, responsible for
  - starting and stopping transactions and for propagating the 'transaction URL'

Uniform Interface with HTTP

- Create, modify and get resources using HTTP methods POST, PUT and GET, respectively:
  - a successful POST returns status code 201 (created) and a Location header containing the URL of the newly created resource (TC begin, participant prepare/commit/forget);
  - a successful PUT returns status code 200 (OK) and any XML data in the body (TC commit, transaction enlistment);
  - a successful GET returns status code 200 (OK) and any XML data in the body;

Example URL's

- List all active transactions
  - GET <s>://<auth>/transaction-coordinator/active
- Start a new transaction
  - POST <s>://<auth>/transaction-coordinator/begin
    - Alternatives:- response body contains multiple URLs for driving commit and rollback
- Commit a transaction: PUT <Tx URL>/commit
  - What it looks like with JAX-RS:

```
@PUT
@Path("transaction-coordinator/{TxId}/commit")
public Response commitTransaction(@PathParam("TxId")String txId) {...}
```

Example URL's (2)

- Enlist a participant in a transaction
  - PUT <Tx URL> (the body identifies the participant URL, the response contains a recovery URL)
    - Alternatives:- request body contains multiple participant URLs for driving prepare, commit, forget etc
- Prepare a participant:
  - POST <participant URL>/prepare - returns a 'status' URL which can be 'probed':
    - GET <status URL> (returns the status if the participant still exists - same as GET <participant URL>)

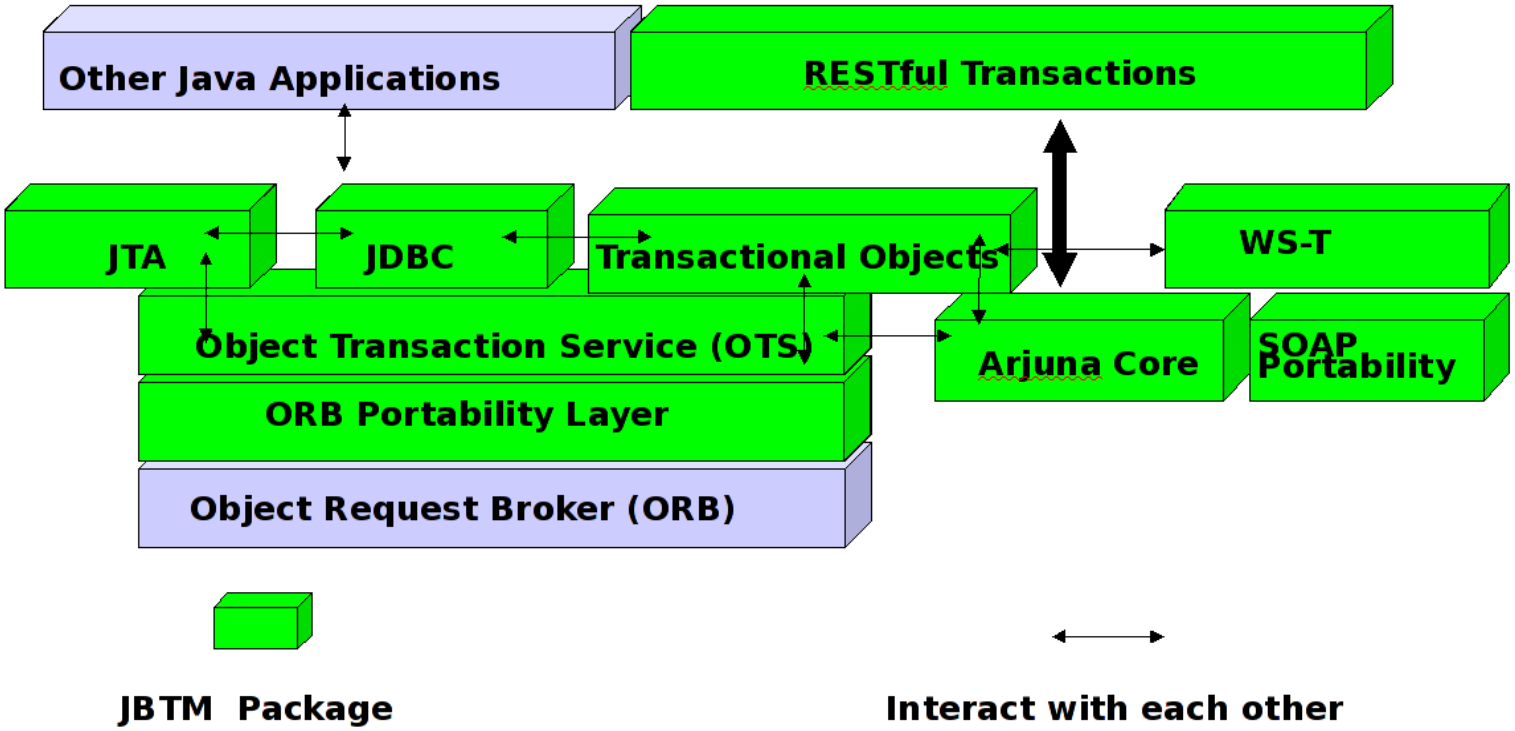
Demonstration

- **Protocol implementation packaged as a war file**
- **Transaction Service** - JBossTS
- **REST resource implementation** - RESTeasy
- **User interface** – GWT
- **Container** – any servlet container will do - JBoss Application Server, Jetty, etc
- **Source code** (for implementation and demo) -

  https://svn.jboss.org/repos/labs/labs/jbosstm/workspace/resttx

Transaction Service Implementation: JBossTS

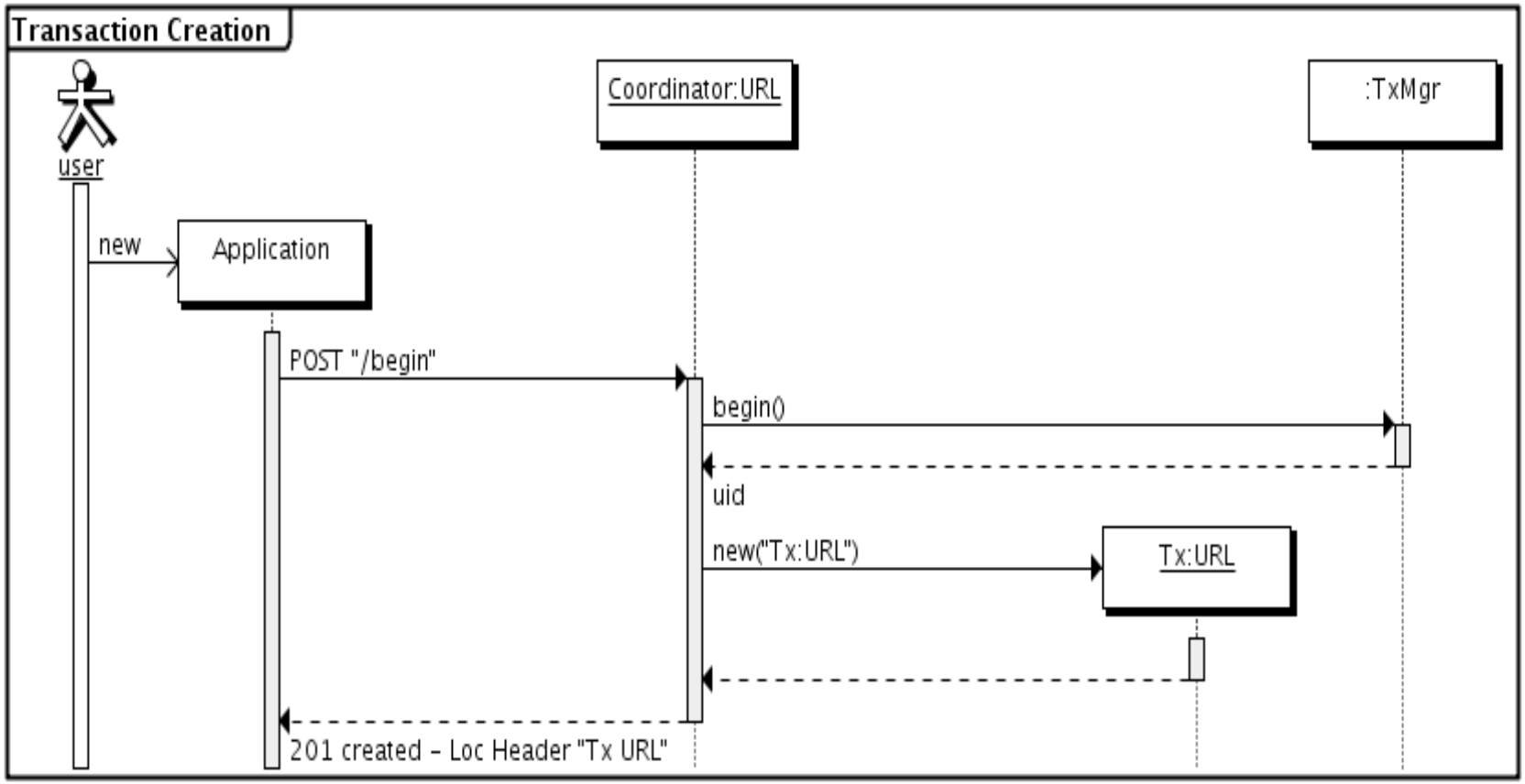
REST implementation integrates with Arjuna Core



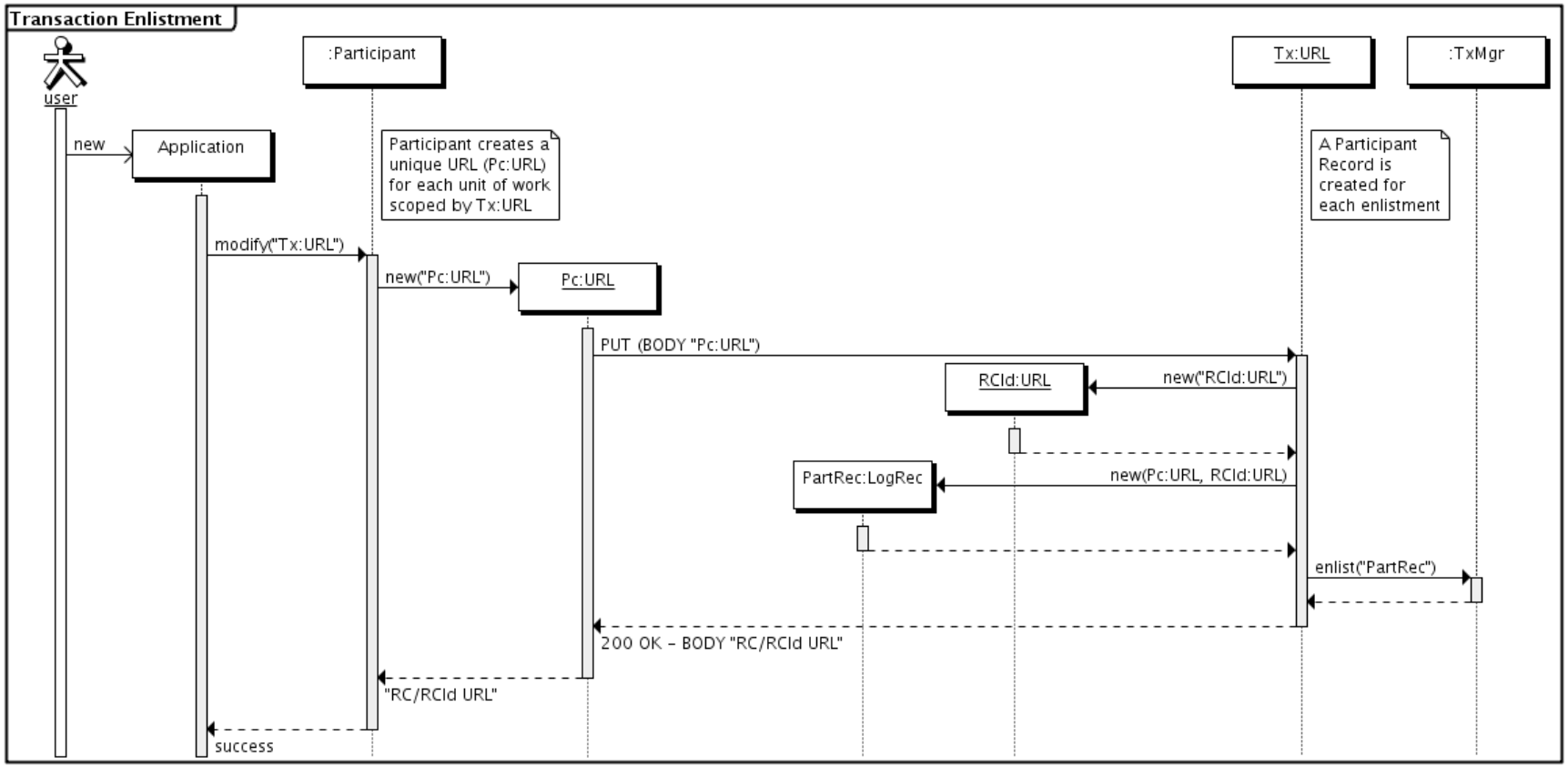
JAX-RS Implementation: RESTeasy

- Java implementation of the protocol (using RESTeasy)
  - JAX-RS compliant
  - Includes an embeddable server for JUnit testing
  - Other useful features

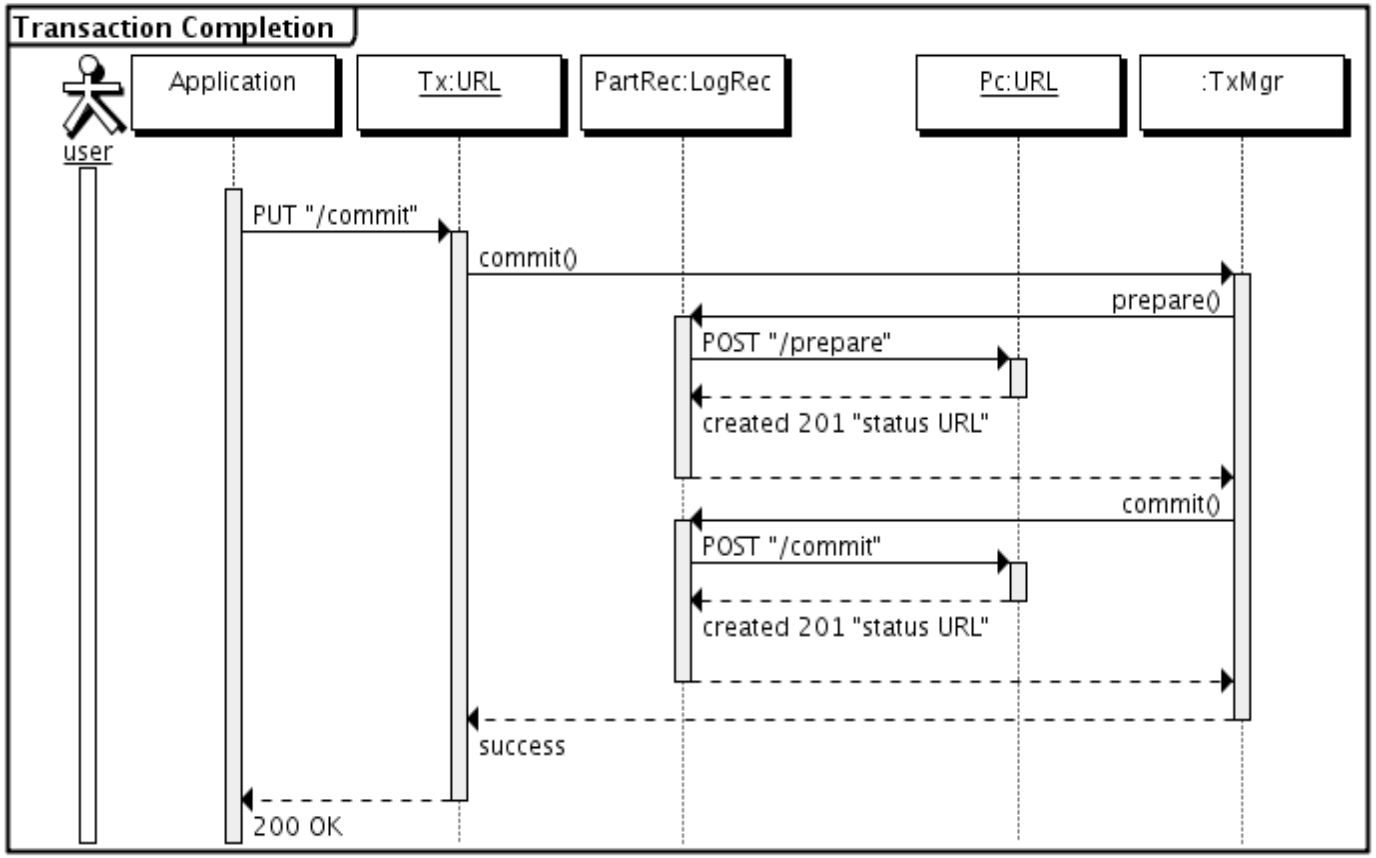
Transaction Creation



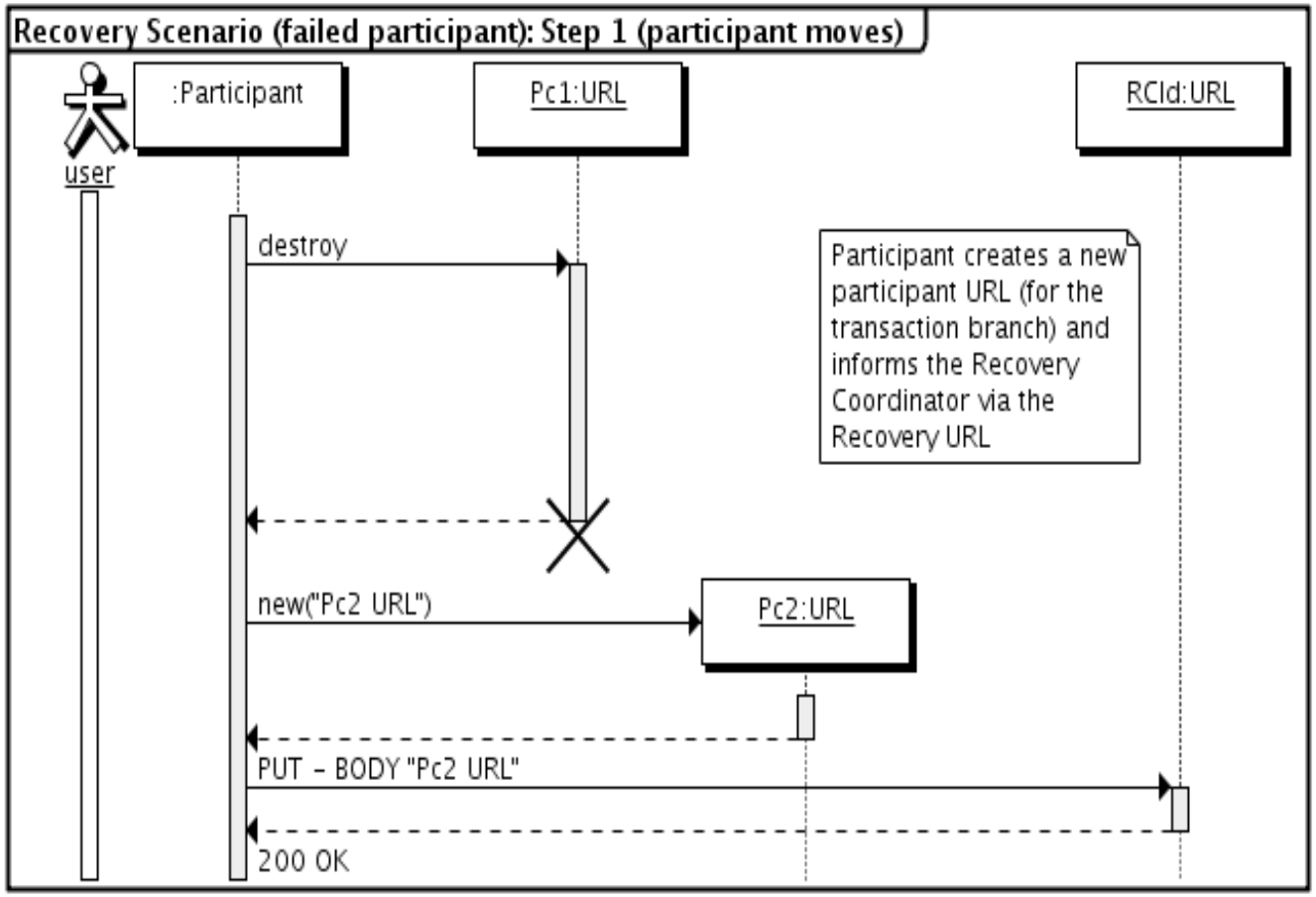
Transaction Enlistment



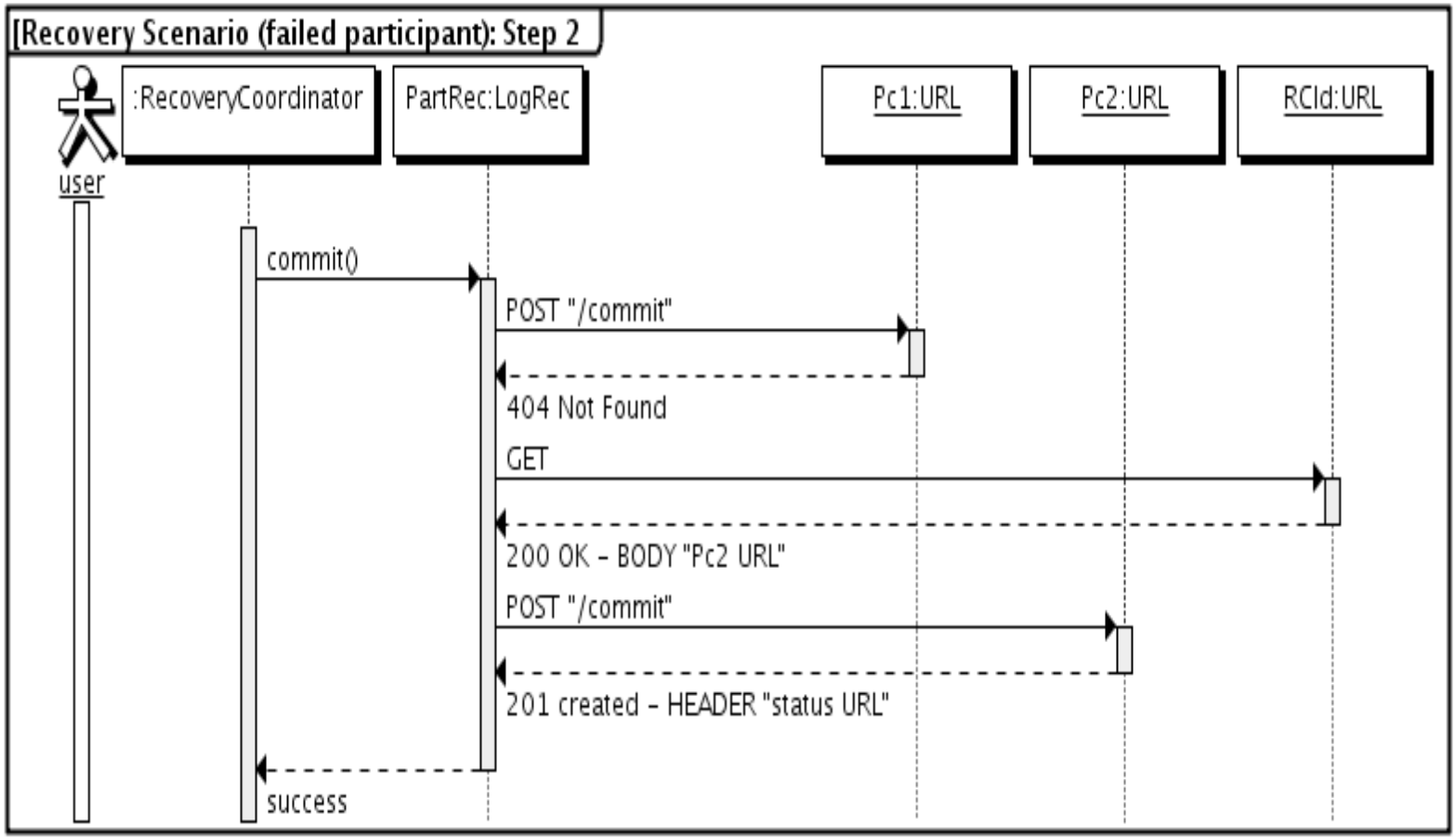
Transaction Completion



Participant Failure



Participant Recovery



Overview

- Web services and transactions
- REST and transactions
- Protocol description and demonstration
- **Future work**
- Summary

Future Work

- Gauge community interest - start a community project/productize accordingly
- Extended RESTful Models
  - Holding locks during an atomic transaction does not scale:
    - “The JDI transaction protocol is a forward-compensation based approach. No assumption about resource locking for the duration of the transaction”;
    - Best effort compensation
- Formalize MIME types for request/response bodies (in both atomic and extended models)

Overview

- Web services and transactions
- REST and transactions
- Protocol description and demonstration
- Future work
- **Summary**

Summary

- Some form of transactional support is required for a wide class of REST style systems;
- We have shown how to provide consistency and reliability guarantees whilst still adhering to REST principals;
- Implementing RESTful protocols is easy provided we use solid flexible tools like JbossTM and RESTeasy.

Thank You

Mark Little
Michael Musgrove
Jboss by Red Hat
http://www.jboss.org/jbosstm/