Demaq: A Foundation for Declarative XML Message Processing

Alexander Böhm  Carl-Christian Kanne
Guido Moerkotte

University of Mannheim
XML Messaging

Customer

Manufacturer

Supplier A

Supplier B

Supplier C

XML

SOA, Web Services, AJAX, RSS/Atom...
Networks of XML Queues
Messaging Rules

- "If a request for an offer comes in, forward it to the legal, finance, and planning departments"
- "If the delivery of all items has been confirmed, send a completion message to the customer"
∞-tier Architectures

XML Messaging
Networks of XML Queues
Messaging Rules
∞-tier Architectures
State of the Art

Introduction

- Introduction
- XML Messaging
- Networks of XML Queues
- Messaging Rules
- ∞-tier Architectures
- State of the Art

Demaq

Appendix

Carl-Christian Kanne, January 8, 2007

Demaq - p. 5/20
State of the Art

topic = ((QIjmsSession)t_sess).getTopic("strmadmin", "oe_queue");
t_pub = t_sess.createPublisher(topic);
db_conn = ((QIjmsSession)t_sess).getDBConnection();
agent = new QIjmsAgent("explicit_enq", null);
adt_msg = ((QIjmsSession)t_sess).createAdtMessage();
lcr_data = new StringBuffer();
lcr_data.append("<ROW_LCR>");
lcr_data.append("xmlns='http://xmlns.tentacle.com/streams/schemas/lcr'");
lcr_data.append("xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'");
lcr_data.append("xsi:schemaLocation='http://xmlns.tentacle.com/streams/schemas/lcr'");
lcr_data.append("http://xmlns.tentacle.com/streams/schemas/lcr/streamslcr.xsd'");
lcr_data.append("<source_database_name>source_dbname</source_database_name>");

... MORE DOCUMENT CONSTRUCTION HERE ...

xml_lcr = tentacle.xdb.XMLType.createXML(db_conn, lcr_data.toString());
adt_msg.setAdtPayload(xml_lcr);
((QIjmsMessage)adt_msg).setSenderId(agent);
System.out.println("Publish message 3 - XMLType containing LCR ROW");
recipList = new QIjmsAgent[1];
recipList[0] = new QIjmsAgent("explicit_dq", null);
((QIjmsTopicPublisher)t_pub).publish(topic, adt_msg, recipList);
t_sess.commit();
Demaq Application

- Complete
- Declarative
- Executable
Demaq Language(s)

Introduction

Demaq Application

Demaq Language(s)

Demaq QML Rules

Demaq Sample Rule

Demaq Server

Messages all the way

Demaq Project

Thank you

Appendix

Carl-Christian Kanne, January 8, 2007
"If the delivery of all items has been confirmed, send a completion message to the customer"
"If the delivery of all items has been confirmed, send a completion message to the customer"

XML messages → new XML messages
"If the delivery of all items has been confirmed, send a completion message to the customer"

XQuery Update Facility
+ Queuing Primitives

XML messages → new XML messages
create rule sendComplete for orderMsgs
if (// deliverymsg/@type = "confirm") then
    let $ordered := fn:count(qs:slice()//ordermsg//item)
    let $delivered := fn:count(qs:slice()//deliverymsg/item)
    where $ordered eq $delivered
    return do enqueue <done> { // orderID } </done>
        into customerReply
Demaq Server

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Messages all the way
- Demaq Project
- Thank you

Introduction

Demaq Application

Demaq Language

Application Rules (QML)

Message Queues (QDL)

Network

Demaq Server

Transaction Engine

Transport

Rule Engine

Transactional Store

Carl-Christian Kanne, January 8, 2007
Messages all the way

- Everything is an XML message
  - Rule Input
  - Rule Output
  - Errors
  - Timeouts

- Messages are processed once, but kept "forever"

- Message History
  - captures process state
  - organized into slices (virtual queues)
  - declarative expiration
http://demaq.net

http://db.informatik.uni-mannheim.de
create property customerID fixed queue requests, orders, deliveryNotifications value //customerID create slicing customers on customerID
Slice usage

- Merge parallel control flow

```plaintext
create property correlationID fixed
queue creditCheck, exportCheck, plantCheck, setupCheck value //correlationID
create slicing checkResults on correlationID

create rule merge for checkResults
if (count(qs:slice()) eq 4) then ...
```
Error Handling

- Plenty of error sources in distributed applications
  - Application-related (dynamic errors)
  - Message-related (invalid XML, wrong schema)
  - Network-related (disconnects, routing, ...)
  - ...

- Message-based error handling
- Error queues, e.g. for rules

```python
create rule errorSource for foo errorqueue errors
```
A Demaq Rule

```
create rule newOfferRequest for customerMsgs
if ( // offerRequest ) then
    let $customerInfo :=
        <requestCustomerInfo reqID="{ //requestID }" >
            <customer>{ //customerID } </customer>
        </requestCustomerInfo>
    let $exportRestrictionInfo := ...
    let $plantCapacityInfo := ...
    return do enqueue $customerInfo into finance,
            do enqueue $exportRestrictionInfo into legal,
            do enqueue $plantCapacityInfo into supplier
```
- Optimization across rules
- Optimization/verification across sites
- Template Folding [XIMEP06]
- Rules driven by XML Schema validation
Demaq Goals

- **Declarative XML message processing language**
  - Move work from programmer to system
  - Data independence
  - Optimizable

- **Execution Engine**
  - Reliability
  - Scalability
  - Reuse DB system knowledge