




  12 oktober 2004

BPEL: Orchestrating XML Web Services

Hugo Brand
Principal Product Sales Consultant
Oracle Application Server
 **ORACLE** Belgium
hugo.brand@oracle.com




 **BPEL: orchestrating XML Web Services**  12 oktober 2004

“SOA and BPEL extend the reach of the application server platform to service-based, process-centric applications.”

Agenda

- What is SOA? Examples
- Introduction to BPEL
- Demo



jug **What is SOA?** **J-Fall**
12 oktober 2004

Reduce friction, enhance visibility, thrive on change

The diagram illustrates four main SOA components, each with associated technologies:

- INTERACT/ACCESS:** Portal, Web Application, API. Technologies: PORTAL JSR-168, B2B - CDL, Struts/JSF.
- ORCHESTRATE:** Create new cell phone plan. Technologies: BPEL, XSLT/XQuery.
- GATEWAY:** Security, Reliability, Logging, SLA, Dynamic Routing. Technologies: WS-Policy/Security, WS-ReliableMessaging.
- BUSINESS SERVICES:** ERP - Billing, Activation, Payment. Technologies: XML/XML Schema, WSDL/WSIF, SOAP, JCA, JMS.


ORACLE

jug **What is SOA?** **J-Fall**
12 oktober 2004

“a legacy application is an application that runs”

Hugo Brand
Oracle
at J-Fall 2004, NL JUG

ORACLE


SOA
 Core principles

J-Fall
 12 oktober 2004

**Service-enable existing applications and
 Build applications that expose services**
 (that are prepared to become legacy...)


- Decouple (Service) Interface from (Service) Implementation
- Service Interface is the contract between Services and Service Consumers
- Services encoded using WSDL are Web Services
 - SOAP is used in service stub and service proxy communication

Assemble services to build the application...

- Promotes Declarative, Model Driven Development: Platform and Language Neutral
- Process-Oriented Orchestration

Self-describing	Process Centric	Asynchronous
Maturing Standards	Loosely Coupled	Model Driven

ORACLE


SOA
 Core principles

J-Fall
 12 oktober 2004

**Service-enable existing applications and
 Build applications that expose services**
 (that are prepared to become legacy...)

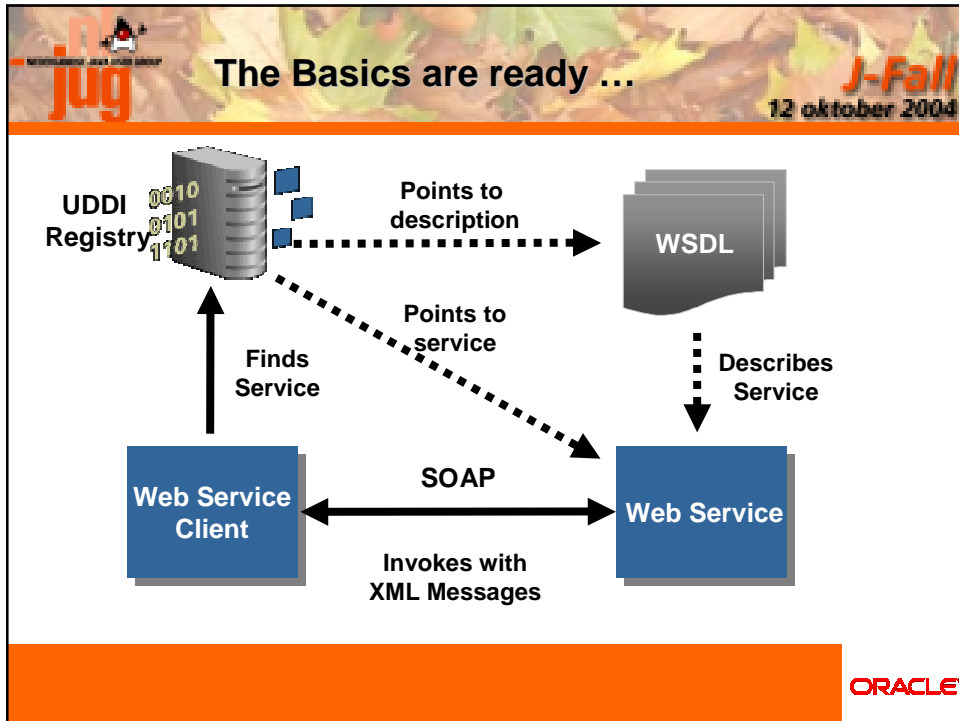
- Using standards: **J2EE**

Assemble services to build the application...

- Using standards: **BPEL**

Self-describing	Process Centric	Asynchronous
Maturing Standards	Loosely Coupled	Model Driven

ORACLE

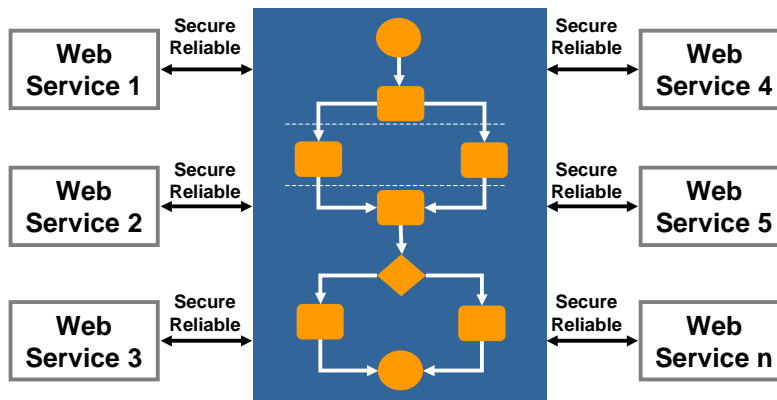
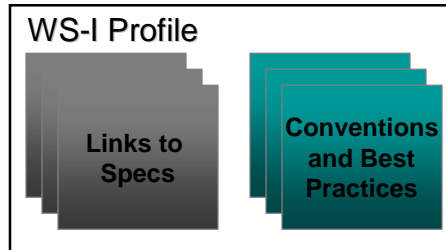


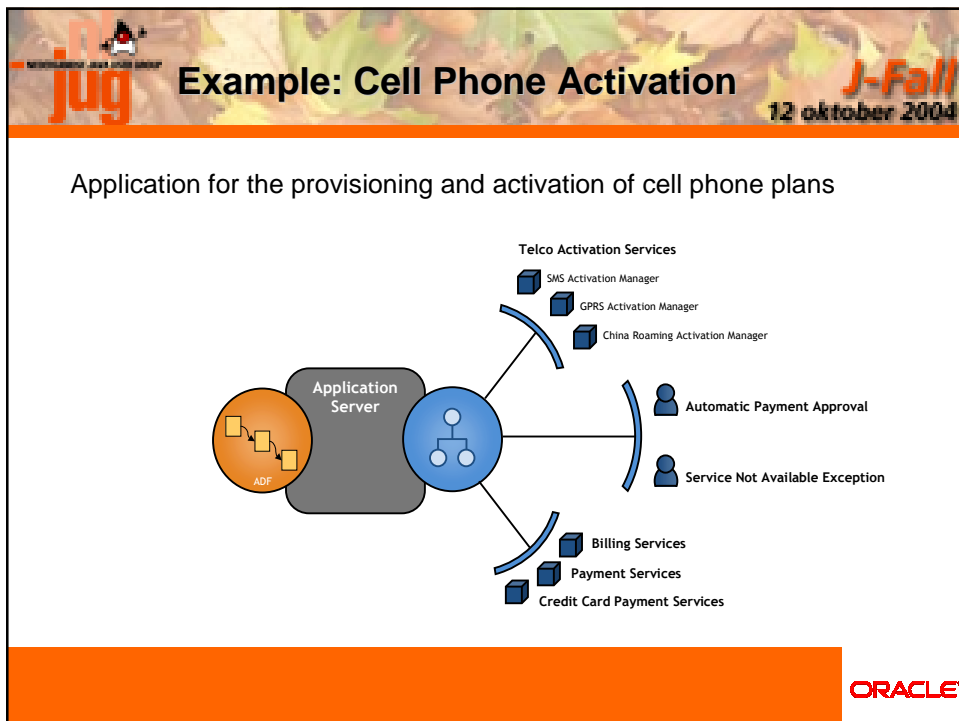
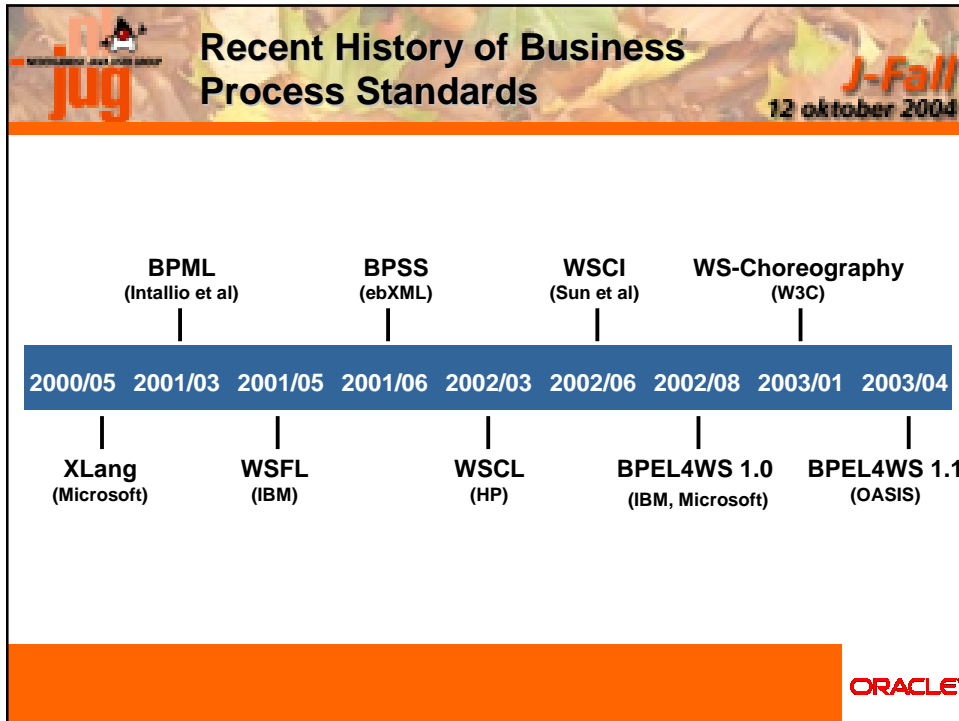
J2EE 1.4 is ready ... **J-Fall**
12 oktober 2004

Java APIs for XML	Description
JAXP	Java API for XML Parsing
JAXB	Java API for XML Data Binding
JAX-RPC 1.1	Java API for XML Remote Procedure Call
SAAJ 1.2	SOAP API for Attachments in Java
JAXR	Java API for XML Registries
EJB 2.1	Stateless Session EJB Endpoint Model
JSR 109	Web Services Deployment Model

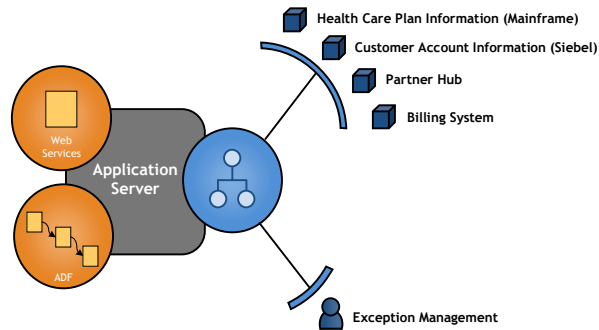
ORACLE

- WS-I
 - Profiles
 - Best practices
 - Testing
- SOAPBuilders
 - WSDL 1.1
 - SOAP 1.1
 - RPC/encoded, Doc/literal

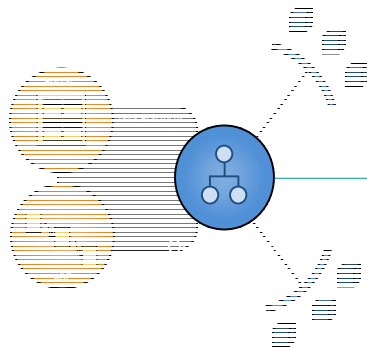




Streamline the processing of 200M prescriptions per year



ORACLE



Connectivity

Heterogenous Back Ends
 Silos of API and mechanisms
 Opaque/heterogeneous data definitions
 Synchronizing multiple data stores

Orchestration

Asynchrony, Flow Coordination, Data Transformation, Compensation, Version Control, Auditing

Scalability

Unpredictable loads
 Asymmetric performance capabilities

Management and Security

Access control, Encryption, Logging, Metering
 Independent of the service

Interaction/Access

Catalog, Customization, Access

ORACLE

- Markup language for composing a set of discrete services into an end-to-end process flow
- 10+ years of research and development from Microsoft (XLANG) and IBM (WSFL, FDML)
- The best integration solution for XML and Web services but also Java, JCA and JMS.
- Rich support for async interactions, parallel processing and exception management.
- Leverages XML Schema, XSLT, XML Query, WS-Security, WS-Addressing and WSIF.
- Composability: A process flow is automatically a service.

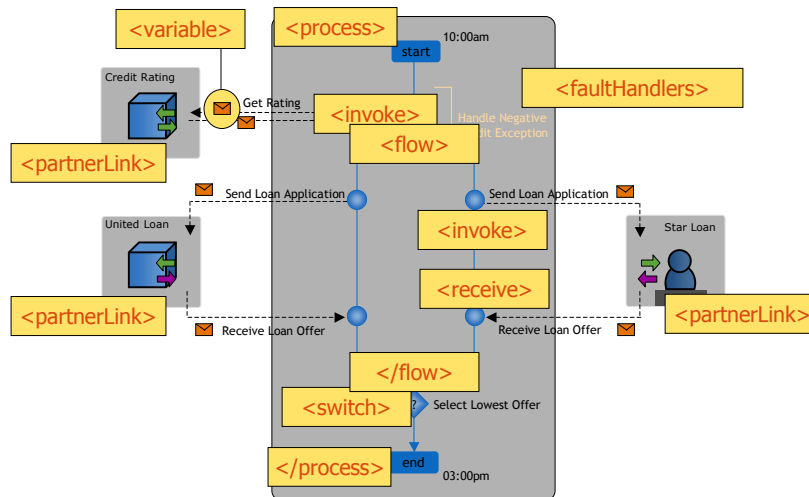
“Gartner believes that BPEL will emerge as the leading industry standard for Web service orchestration and coordination of business processes.”

- David Smith, Research Vice President and fellow, Gartner

“BPEL is the future of the integration space in my view...Why? Because the value is so much higher when you provide not only a way to integrate applications, but also a way to create services from them and put them into business processes”

- John Rymer, Vice President, Forrester Research, Inc.

ORACLE

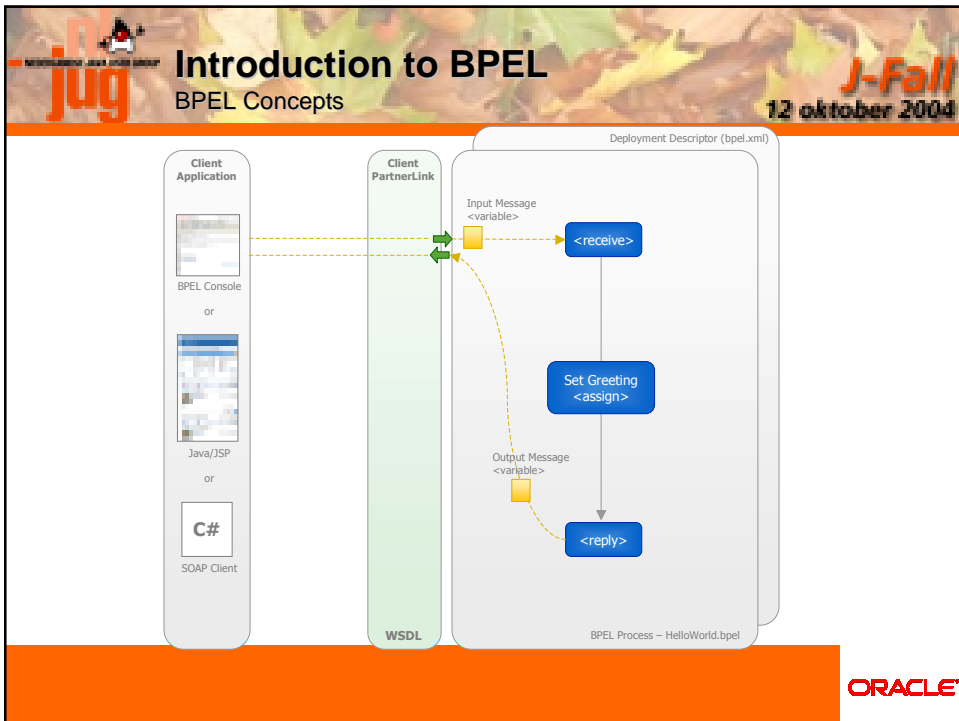


ORACLE

jug **Introduction to BPEL** **J-Fall**
 Use case 12 oktober 2004

“How do I implement, compile, deploy and run my first BPEL Process? I would like that BPEL Process to generate and return a greeting”

ORACLE



jug **Introduction to BPEL** **J-Fall**
 BPEL Source code 12 oktober 2004

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<process xmlns="http://schemas.xmlsoap.org/wsdl2003-03-07/bpel" name="CreditRating" targetNamespace="http://schemas.xmlsoap.org/wsdl2003-03-07/bpel">
  <partnerLink name="Client" type="Client" role="Client"/>
  <partnerLink name="CreditRatingService" type="CreditRatingService" role="CreditRatingService"/>
  <variable name="Greeting" type="xsd:string"/>
  <variable name="Response" type="xsd:string"/>
  <sequence>
    <receive partner="Client" message="CreditRatingRequest"/>
    <assign/>
    <send partner="CreditRatingService" message="CreditRatingRequest"/>
    <receive partner="CreditRatingService" message="CreditRatingResponse"/>
    <assign/>
    <send partner="Client" message="CreditRatingResponse"/>
  </sequence>
</process>
  
```

- <process> is the top level element
- <partnerLink>, channel use to interact with client (and services integrated in process)
- <variable>- reference to an XML message receive or sent to the <partnerLink>s.
- Process flow: sequence of activities defining the process logic.
- Initiate a new instance of the process when a process request is received
- Create and assign greeting to output message.
- Send synchronous reply to client passing output variable as response

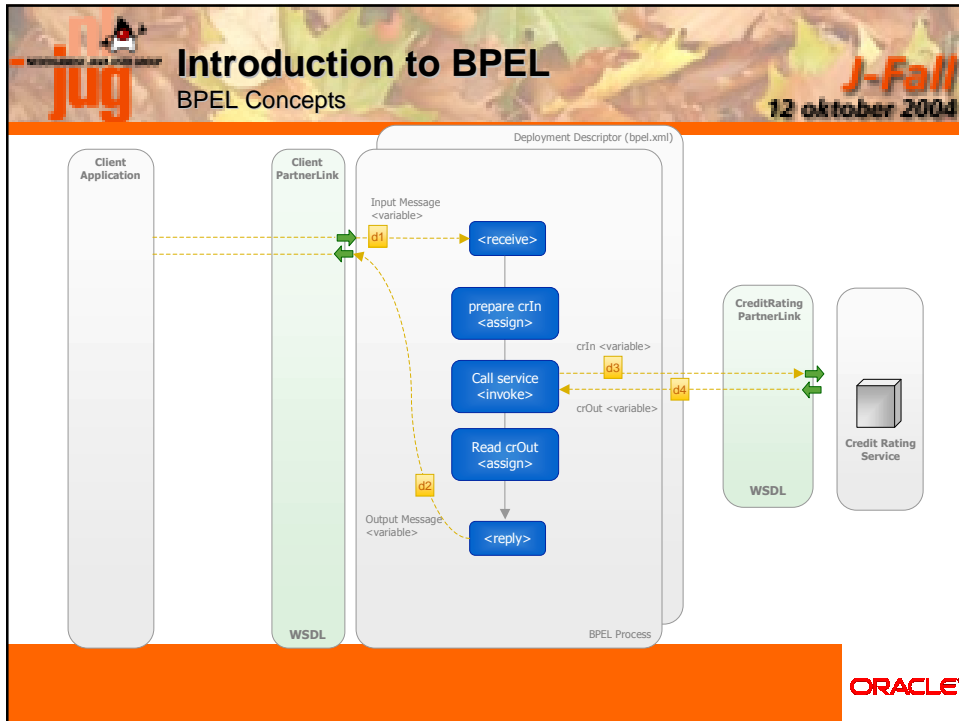
ORACLE

jug **Introduction to BPEL** **J-Fall**
 Use case 12 oktober 2004

"How do I invoke a synchronous credit rating web service from within a BPEL process?"

The diagram illustrates the interaction between a BPEL process and a web service. On the left, a vertical line represents the BPEL process boundary. An arrow labeled 'CreditRatingInput' points from the process into a green 3D block representing the 'Web Service Functional Building Block process operation returns credit rating'. Above this block, a dashed arrow points to a box labeled 'Self-Described Interface (WSDL)'. Below the block, an arrow labeled 'CreditRatingResponse' points from the block back to the process boundary. At the bottom left, a dashed arrow points to the text 'Transport (SOAP Over HTTP)', indicating the communication protocol used.

ORACLE



Introduction to BPEL
BPEL Source code

J-Fall
12 oktober 2004

Syntax/Example

```
<plnk:partnerLinkType name="CreditRatingService">
  <plnk:role name="CreditRatingServiceService">
    <plnk:portType name="tns:CreditRatingService" />
  </plnk:role>
</plnk:partnerLinkType>
```

Extract of the Credit Rating WSDL file

```
<partnerLinks>
  <partnerLink name="creditRatingService"
    partnerLinkType="crs:CreditRatingService"
    partnerRole="CreditRatingServiceService" />
</partnerLinks>
```

Extract of LoanFlow.bpel

```
<?xml version="1.0" encoding="UTF-8"?>
<bpel-process src="LoanFlow.bpel LoanFlow.wsdl">
<properties id="creditRatingService">
  <property name="wsdl-location">
    http://localhost:9700/orabpel/default/CreditRatingService?wsdl
  </property>
</properties>
</bpel-process>
```

Extract of LoanFlow's deployment descriptor - bpel.xml

Tips and Tricks
Cross reference between partner WSDL, .bpel implementation and deployment descriptor. Make sure the namespaces match!

ORACLE

jug **Introduction to BPEL** **J-Fall**
 Use case 12 oktober 2004

“American Loan exposes a web service that can take anywhere from a couple of minutes to a couple days to process a loan application into a loan offer. How can I leverage that asynchronous loan processor service as part of my BPEL Process?”

Initiate Port
initiate

Callback Port
onResult callback

American Loan Asynchronous Web Service

[2:05] receive
[2:06] process...
[2:22] callback

ORACLE

jug **Introduction to BPEL** **J-Fall**
 BPEL Concepts 12 oktober 2004

Client Application
BPEL Console

Client PartnerLink
WSDL

Deployment Descriptor (bpel.xml)

BPEL Process

LoanService PartnerLink
WSDL

Async Loan Processor Service

Dehydration Point
For scalability and reliability, in-flight instances are pushed to DB until callback is received.

ORACLE

jug **Introduction to BPEL** **J-Fall**
 Use case 12 oktober 2004

“Given that AmericanLoan and UnitedLoan can take up to 5 days to process a loan request, is it possible to invoke those services in parallel?”

The diagram illustrates a parallel execution structure. A horizontal bar at the top is labeled "In Parallel". Two vertical arrows point downwards from this bar to two separate circular nodes. Each node contains a 3D building icon representing a service. The left node is labeled "United Loan" and the right node is labeled "American Loan".

ORACLE

jug **Introduction to BPEL** **J-Fall**
 BPEL Concepts 12 oktober 2004

The diagram shows a BPEL process flow. A central box labeled "BPEL Process" contains a flow structure. At the top, a horizontal bar is labeled "<flow>". Below it, two vertical bars represent parallel flows. Each flow contains a sequence of two activities: "Initiate service <invoke>" and "Wait for callback <receive>". Dashed boxes group these two activities for each flow. On the left, a box labeled "United Loan" is connected to the first flow by a green arrow pointing right and a purple arrow pointing left. On the right, a box labeled "American Loan" is connected to the second flow by a green arrow pointing right and a purple arrow pointing left.

ORACLE

jug **Introduction to BPEL** **J-Fall**
 BPEL Source code 12 oktober 2004

```

<flow name="Flow1" from="Start" to="End" type="parallel">
  <branch name="Branch1">
    <task name="Task1"/>
  </branch>
  <branch name="Branch2">
    <task name="Task2"/>
  </branch>
</flow>
  
```

Use <flow> to define parallel branching

This is one branch

This is a second branch

ORACLE


jug **Introduction to BPEL** **J-Fall**
 Use case 12 oktober 2004

“I have a Customer Entity Bean that allows me to retrieve a SSN based on an email id. How can I invoke that bean from within my BPEL process?”

```

graph LR
  BPEL[A BPEL process] -- call --> Bean[Customer Entity Bean]
  
```


ORACLE


 **Introduction to BPEL**
Java integration **J-Fall**
12 oktober 2004

There are 3 solutions to this problem:

- Wrapping the Java component into a full blown web service using JDeveloper or Axis
- Using the WSIF Java Binding
- Using the Java BPEL exec extension (<bpelx:exec>)


ORACLE

 **Introduction to BPEL**
BPEL Concepts **J-Fall**
12 oktober 2004

 **<bpelx:exec>**


1. BPEL Extension which allows developers to embed a snippet of Java code within a BPEL process
2. When the <bpelx:exec> is reached, the server executes the snippet of Java code (within its JTA transaction context)
3. A set of built-in methods allow developers to read and update scope variables, instance metadata and audit trail
4. XML Façade can be used to simplify DOM manipulation
5. Java exceptions are converted into BPEL faults and bubbled into BPEL process
6. The snippet can propagate its JTA transaction to session and entity beans that it calls

ORACLE



Introduction to BPEL

BPEL Source code



12 oktober 2004

```

10 <!-- BPEL SOURCE CODE -->
11 <!-- BPEL SOURCE CODE -->
12 <!-- BPEL SOURCE CODE -->
13 <!-- BPEL SOURCE CODE -->
14 <!-- BPEL SOURCE CODE -->
15 <!-- BPEL SOURCE CODE -->
16 <!-- BPEL SOURCE CODE -->
17 <!-- BPEL SOURCE CODE -->
18 <!-- BPEL SOURCE CODE -->
19 <!-- BPEL SOURCE CODE -->
20 <!-- BPEL SOURCE CODE -->
21 <!-- BPEL SOURCE CODE -->
22 <!-- BPEL SOURCE CODE -->
23 <!-- BPEL SOURCE CODE -->
24 <!-- BPEL SOURCE CODE -->
25 <!-- BPEL SOURCE CODE -->
26 <!-- BPEL SOURCE CODE -->
27 <!-- BPEL SOURCE CODE -->
28 <!-- BPEL SOURCE CODE -->
29 <!-- BPEL SOURCE CODE -->
30 <!-- BPEL SOURCE CODE -->
31 <!-- BPEL SOURCE CODE -->
32 <!-- BPEL SOURCE CODE -->
33 <!-- BPEL SOURCE CODE -->
34 <!-- BPEL SOURCE CODE -->
35 <!-- BPEL SOURCE CODE -->
36 <!-- BPEL SOURCE CODE -->
37 <!-- BPEL SOURCE CODE -->
38 <!-- BPEL SOURCE CODE -->
39 <!-- BPEL SOURCE CODE -->
40 <!-- BPEL SOURCE CODE -->
41 <!-- BPEL SOURCE CODE -->
42 <!-- BPEL SOURCE CODE -->
43 <!-- BPEL SOURCE CODE -->
44 <!-- BPEL SOURCE CODE -->
45 <!-- BPEL SOURCE CODE -->
46 <!-- BPEL SOURCE CODE -->
47 <!-- BPEL SOURCE CODE -->
48 <!-- BPEL SOURCE CODE -->
49 <!-- BPEL SOURCE CODE -->
50 <!-- BPEL SOURCE CODE -->
51 <!-- BPEL SOURCE CODE -->
52 <!-- BPEL SOURCE CODE -->
53 <!-- BPEL SOURCE CODE -->
54 <!-- BPEL SOURCE CODE -->
55 <!-- BPEL SOURCE CODE -->
56 <!-- BPEL SOURCE CODE -->
57 <!-- BPEL SOURCE CODE -->
58 <!-- BPEL SOURCE CODE -->
59 <!-- BPEL SOURCE CODE -->
60 <!-- BPEL SOURCE CODE -->
61 <!-- BPEL SOURCE CODE -->
62 <!-- BPEL SOURCE CODE -->
63 <!-- BPEL SOURCE CODE -->
64 <!-- BPEL SOURCE CODE -->
65 <!-- BPEL SOURCE CODE -->
66 <!-- BPEL SOURCE CODE -->
67 <!-- BPEL SOURCE CODE -->
68 <!-- BPEL SOURCE CODE -->
69 <!-- BPEL SOURCE CODE -->
70 <!-- BPEL SOURCE CODE -->
71 <!-- BPEL SOURCE CODE -->
72 <!-- BPEL SOURCE CODE -->
73 <!-- BPEL SOURCE CODE -->
74 <!-- BPEL SOURCE CODE -->
75 <!-- BPEL SOURCE CODE -->
76 <!-- BPEL SOURCE CODE -->
77 <!-- BPEL SOURCE CODE -->
78 <!-- BPEL SOURCE CODE -->
79 <!-- BPEL SOURCE CODE -->
80 <!-- BPEL SOURCE CODE -->
81 <!-- BPEL SOURCE CODE -->
82 <!-- BPEL SOURCE CODE -->
83 <!-- BPEL SOURCE CODE -->
84 <!-- BPEL SOURCE CODE -->
85 <!-- BPEL SOURCE CODE -->
86 <!-- BPEL SOURCE CODE -->
87 <!-- BPEL SOURCE CODE -->
88 <!-- BPEL SOURCE CODE -->
89 <!-- BPEL SOURCE CODE -->
90 <!-- BPEL SOURCE CODE -->
91 <!-- BPEL SOURCE CODE -->
92 <!-- BPEL SOURCE CODE -->
93 <!-- BPEL SOURCE CODE -->
94 <!-- BPEL SOURCE CODE -->
95 <!-- BPEL SOURCE CODE -->
96 <!-- BPEL SOURCE CODE -->
97 <!-- BPEL SOURCE CODE -->
98 <!-- BPEL SOURCE CODE -->
99 <!-- BPEL SOURCE CODE -->
100 <!-- BPEL SOURCE CODE -->

```


<bpelx:exec allows you to embed a Java snippet within a BPEL process

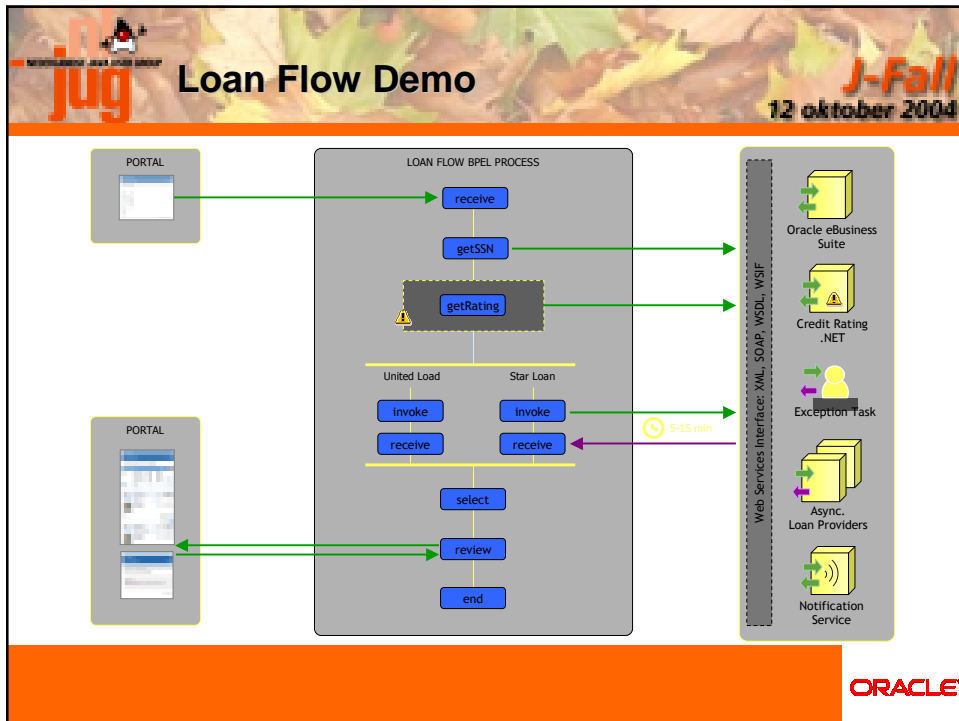
Use getVariableData built-in method to access scope variables. Use XML Façade to simplify DOM manipulation.

Look up entity bean.

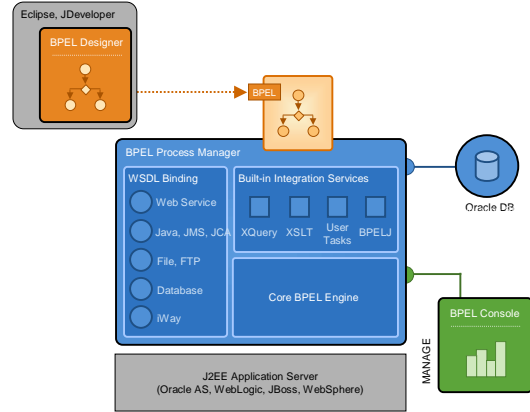
Log SSN to audit trail of this instance

Catch and log any potential exceptions



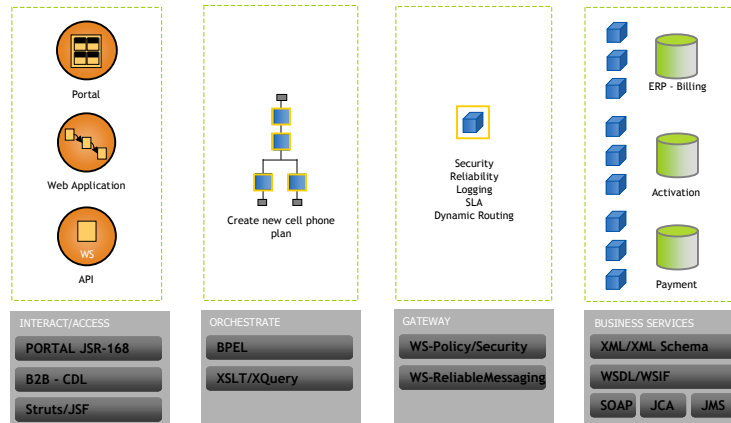


Enterprise-strength infrastructure for designing, deploying and managing BPEL business processes.



- **Comprehensive and native BPEL implementation**
- Easy-to-use modeling tool
- Scalable and reliable engine
- Flexible binding framework
- Rich management and monitoring
- Support for Oracle AS, WebLogic and WebSphere
- Get up and running in less than 15 minutes!

Reduce friction, enhance visibility, thrive on change



jug **Longer Term Issues** Policies, Choreography, QoS and Manageability **J-Fall** 12 oktober 2004

Management

Clients/Remote Portlets

Choreography

Orchestration

Policy

Reliability Transactions Events

Addressing

SOAP, WSDL, UDDI

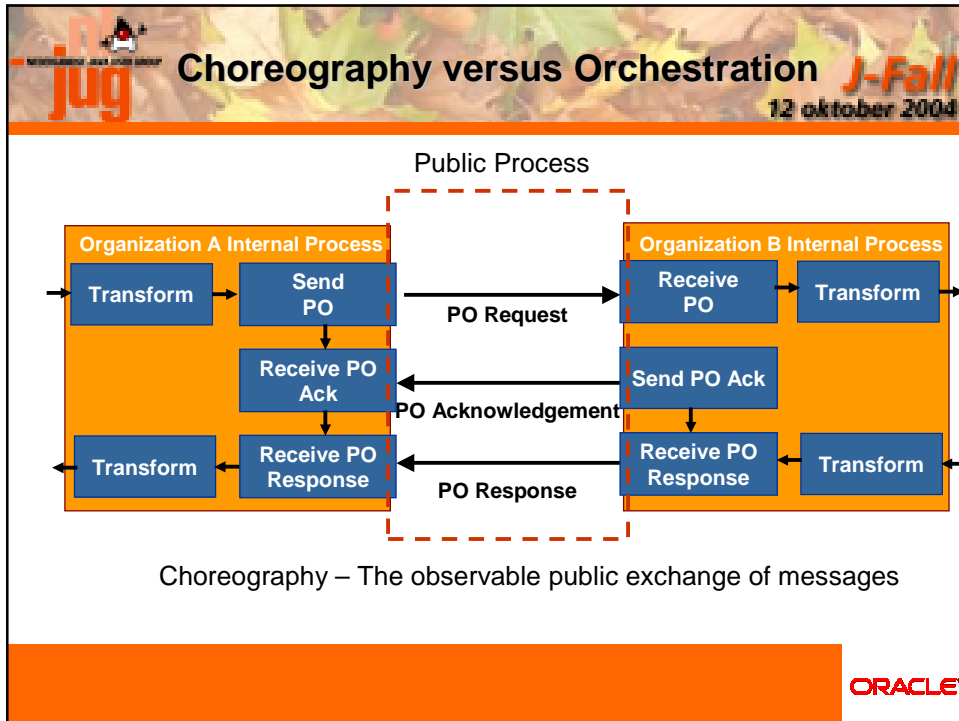
Security

ORACLE

jug **Politics of Web Services** Consensus? Utility? Interoperability? **J-Fall** 12 oktober 2004

- Addressing
 - WS-Addressing
 - WS-MessageDelivery
- Policy
 - WS-Policy
- Reliability
 - WS-Reliability
 - WS-ReliableMessaging
- Transactions
 - WS CAF
 - Context, Transaction, Coordination
 - WS Transactions
 - Atomic, Business Activity, Coordination
- Choreography
 - Relationship to BPEL
- Management
 - WSDM, WSRF

ORACLE



jug **SOA** Core principles **J-Fall**
12 oktober 2004

**Service-enable existing applications and
Build applications that expose services**
(that are prepared to become legacy...)

- Using standards: **J2EE**

Assemble services to build the application...

- Using standards: **BPEL**

Self-describing	Process Centric	Asynchronous
Maturing Standards	Loosely Coupled	Model Driven

ORACLE

jug **Javapolis** **J-Fall**
12 oktober 2004




ORACLE

jug **Javapolis** **J-Fall**
12 oktober 2004



- Metropolis (Antwerp)
- 13-17 December
 - 13-14 December: University
 - 15-16 December: Tech conference
 - 17 December: Business Day
 - 40 B.O.F sessions (6pm til 10pm)
 - Exhibition
 - A party, a Movie, ...

ORACLE

 **Javapolis** **J-Fall**
12 oktober 2004



- Adrian Colyer (AspectJ)
- Rod Johnson (Spring)
- Erich Gamma (Eclipse)
- Joshua Bloch and Neal Gafter (J2SE 5.0)
- Gavin King (Hibernate)
- Linda DeMichiel (EJB 3.0 spec lead)
- Robin Roos (JDO expert group)
- **Edwin Khodabakchian (BPEL, Collaxa/Oracle)**
- Craig McClanahan (Struts + JSF spec lead)
- Rick Ross (javalobby)
- Cedric Beust (Google)
- Jan Baan
- Marc Fleury (Jboss)
- ... many more !!!

ORACLE

 **Javapolis** **J-Fall**
12 oktober 2004



<http://wiki.javapolis.com>

ORACLE