Swordfish
The Eclipse SOA Runtime Framework
SOA

Architecture and Features

Usage Scenarios

Past, present, and future

Your questions
SOA
Service Oriented Architecture
Application landscapes in large enterprises tend to evolve in uncontrolled ways.
Ad-hoc integration increases complexity and leads to maintenance nightmares.
Hub-and-Spoke approaches often fail to deliver significant business value.
EAI = Octopus
SOA is a structured approach to defining a sound Enterprise Architecture.
Services encapsulate business functionality and share common characteristics

<table>
<thead>
<tr>
<th>Standardized Contract</th>
<th>Common contract design standards within the same service inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loosely coupled</td>
<td>Contracts impose low consumer coupling requirements</td>
</tr>
<tr>
<td>Abstract</td>
<td>Information about services is limited to what is published in service contracts</td>
</tr>
<tr>
<td>Reusable</td>
<td>Service logic is agnostic and reusable</td>
</tr>
<tr>
<td>Autonomous</td>
<td>Services exercise a high level of control over their underlying runtime execution environment.</td>
</tr>
<tr>
<td>Stateless</td>
<td>Services defer the management of state information when necessary.</td>
</tr>
<tr>
<td>Discoverable</td>
<td>Services are supplemented with meta data by which they can be discovered and interpreted.</td>
</tr>
<tr>
<td>Composable</td>
<td>Services can be composed into more complex services</td>
</tr>
</tbody>
</table>
Architecture and Features
**OSGi services alone are not sufficient for enterprise-scale SOA**

<table>
<thead>
<tr>
<th>Four tenets of SOA*</th>
<th>OSGi/Equinox</th>
<th>SOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service boundaries are explicit</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
functionality can only be accessed through the service boundary |
| Services are autonomous | ✓ | | ✓ |
no hard dependencies on other services |
| Services expose schema and contract, not class and type | | | ✓ |
interface is defined in a platform and language neutral way |
| Services negotiate using policy | | | ✓ |
capabilities and requirements must match, negotiation is dynamic |

* Source: http://msdn.microsoft.com/msdnmag/issues/04/01/Indigo/default.aspx
Swordfish is based on all three relevant standards in the SOA space.

- **SCA**: Programming model, Assembly description format
- **JBI**: Messaging abstraction, Message routing
- **OSGi**: Component model, Module deployment mechanism, Classloading
Swordfish is built on top of a bundelized JBI runtime environment

Framework plug-ins

- Service Discovery
- Message Processor
- Advanced Mgmt.
- Remote Config.

Framework core

- Framework Controller

Equinox

- JBI SE
- JAX-WS
- Java
- BPEL
- JBI BC
- JMS
- HTTP
- NMR
- ServiceMix

JBI components deployed as OSGi bundles
Dynamic service discovery

Logical service name + policy

physical endpoint address

SOA Registry/Repository

Service Discovery

JBI SE Java

JBI BC HTTP

Provider 1

Provider 2
Policy-driven message processing

- Policy
  - Policy planner
    - current runtime configuration
      - create
    - Processing chain
      - Processor 1
      - Processor 2
      - ... Processor n
      - processed Message

Message processed

create
The SOA Tools Platform Project (STP) will provide extensive tool support for Swordfish

Create Service definitions following a contract-first (WSDL-first) approach

Create Services based on annotated Java code (JAX-WS)

Create WS-Policy documents to specify non-functional requirements and capabilities of a service

Deploy Service implementations into Swordfish runtime

Create SCA composite applications both top-down and bottom-up
Usage scenarios
**Scenario 1: RCP application acting as a web service consumer**

Bundles that make up your RCP application

Your consumer implementation lives here!

Web Service

SOAP over HTTP

Swordfish bundles
Scenario 2: Swordfish as a server-side runtime hosting a service
Scenario 3: Swordfish as a server-side runtime hosting a composite service

Your BPEL process lives here!

Your Java service implementations live here!

Web Service consumer

external Web Service

SOAP over HTTP

HTTP Service

JBI BC HTTP

JBI NMR

JBI SE BPEL

JBI SE Java

OSGi Registry

register

Swordfish bundles

Equinox
Past, present, and future
Bottom line: We won’t make it for Ganymede, but we’ll be part of the simultaneous release in 2009.
Where we are now:
The groundwork is done
What’s next: Designing the framework

Milestone 1, mid of Q2 2008

First draft of message processing framework
Hosting of JAX-WS services

Milestone 2, mid of Q3 2008

Message processing done
Draft of other framework aspects (Service Discovery, Management Integration, Remote Configuration)

Release 1.0, end of Q4 2008

Framework and exemplary implementation done
Support for deployment from STP/WTP
What’s on the horizon: Adding the SCA layer on top

Service Units deployed as bundles

STP Intermediary Model?
Swordfish has a diverse community of committers and (future) contributors

**Committers**
- Oliver Wolf
- Dietmar Wolz
- Gerald Preissler
- Klaus Kiehne
- James Strachan
- Guillaume Nodet
- Jonas Lim
- Adrian Co

**Contributors**
- Andrey Kopachevsky

**Future Contributors**
- Heiko Seeberger

Maybe you???
Your questions
Legal notices

The photos used in this presentation have been made available by the copyright holders under a Creative Commons License.

Photo 1 (Barrio in Caracas) by http://www.flickr.com/people/rooshv/,
Photo 2 (Production plant) by
Photo 3 (Cables) by http://www.flickr.com/photos/ryangreenberg/
Photo 4 (Octopus) by http://www.flickr.com/people/ericasimone/
Photo 5 (Brasilia) by http://www.flickr.com/photos/car_tav/
Photo 6 (Lego blocks) by http://www.flickr.com/people/ppdigital/