SOA/OpenESB/OpenSSO

by Sang Shin
Java Technology Architect
Sun Microsystems
Goals of the Presentation

• SOA/OpenESB (100 minutes)
  – Learn how to build/deploy/test SOA application (Composite application) using OpenESB technology
  – More emphasis on “How?” over “What is?/Why?”

• OpenSSO (50 minutes)
  – Learn how Single-Sign-On (SSO) works internally and externally
  – Learn how Fedlet works
Agenda of “SOA/OpenESB” Part

- WSDL Tutorial
- BPEL Tutorial
- JBI & OpenESB
- NetBeans Support of JBI & OpenESB
- JBI/OpenESB and GlassFish
- Java EE Service Engine
- Intelligent Event Processor (IEP)
- Enterprise Data Mashup (EDM)
Speaker's qualifications

- Sang Shin is a Technology Architect & lead Evangelist at Sun Microsystems
- Sang Shin used to teach software engineering courses at Brandeis Univ.
- Sang Shin teaches “javapassion.com” online courses
  - Java SE, Java EE Basics, Java EE Advanced, JavaFX, Ajax & Web 2.0, Ruby/JRuby/Rails, Web Services
  - Groovy/Grails, Scala (in future)
  - SOA/OpenESB/OpenSSO (with Passion!) course starts from Jan. 15th, 2010
Brief WSDL Tutorial
Example Business Logic

• Simple service providing stock quotes
• A single operation called GetLastTradePrice
• Deployed using SOAP 1.1 over HTTP
• Request takes a ticker symbol of type string
• Response returns price as a float
WSDL Elements

- Types
- Message
- Operation
- Port Type
- Binding
- Port
- Service
- PartnerLink
WSDL Elements

- **Types**
  - Data type definitions
  - Used to *describe exchanged messages*
  - Uses W3C XML Schema as canonical type system
WSDL Example: Types

<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
    xmlns:tns="http://example.com/stockquote.wsdl"
    xmlns:xsd="http://example.com/stockquote.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
        <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="TradePriceRequest">
                <complexType>
                    <all>
                        <element name="tickerSymbol" type="string"/>
                    </all>
                </complexType>
            </element>
            <element name="TradePrice">
                <complexType>
                    <all>
                        <element name="price" type="float"/>
                    </all>
                </complexType>
            </element>
        </schema>
    </types>
</definitions>
WSDL Elements

• Messages
  – Abstract, typed definitions of data being exchanged

• Operations
  – Abstract description of an action
  – Refers to an input and/or output messages

• Port type
  – Collection of operations
  – Abstract definition of a service
Messages, Operation, Port type

```xml
<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
  <!-- More operations -->
</portType>
```
WSDL Elements

• Binding
  – **Concrete protocol and data format** (encoding) for a particular Port type
    • Protocol examples: SOAP 1.1 over HTTP or SOAP 1.1 over SMTP
    • Encoding examples: SOAP encoding, RDF encoding

• Port
  – **Endpoint address** for binding
  – URL for HTTP, email address for SMTP

• Service
  – Aggregate set of related ports
Binding, Port, Service

```xml
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document">
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input> <soap:body use="literal" /> </input>
    <output> <soap:body use="literal" /> </output>
  </operation>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
```
BPEL Tutorial
Need for Business Process (Orchestration)

• We also need a way to orchestrate operations among participating services and clients
  – WSDL lets you specify only a set of operations
• Example Scenario: Concert ticket purchase Web service has 3 operations
  – Getting a price quote
  – Purchase a ticket
  – Confirmation and cancellation
• Orchestration activities required
  – These operations need to be called in sequential order
  – The 3rd activity depends on the outcome of the 2nd
BPEL: Relationship to Partners

- Orchestrating Process (BPEL)
- Partner Service
- Partner Service
- Partner Service
- Partner Service

WSDL
BPEL Document Structure

```xml
<process>
  <!-- Definition and roles of process participants -->
  <partnerLinks> ... </partnerLinks>
  <!-- Data/state used within the process -->
  <variables> ... </variables>
  <!-- Properties that enable conversations -->
  <correlationSets> ... </correlationSets>
  <!-- Exception handling -->
  <faultHandlers> ... </faultHandlers>
  <!-- Error recovery – undoing actions -->
  <compensationHandlers> ... </compensationHandlers>
  <!-- Concurrent events with process itself -->
  <eventHandlers> ... </eventHandlers>
  <!-- Business process flow -->
  (activities)*
</process>
```
BPEL: Basic Activities

- **<invoke>**
  - To invoke a one-way or request/response operation on a portType offered by a partner

- **<receive>**
  - To do a blocking wait for a matching message to arrive
  - **Instantiator of the business process**

- **<reply>**
  - To send a message in reply to a message that was received through a <receive>
  - The combination of a <receive> and a <reply> forms a request-response operation on the WSDL portType
BPEL: Structured Activities

- `<sequence>`
  - Perform activities in sequential order
- `<flow>`
  - Perform activities in parallel
- `<if>`
  - Conditional choice of activities
Purchase Order Business Process

- Receive <PO>
- Invoke <InventoryService>
- Invoke <CreditService>
- Reply <Invoice>
Purchase Order Business Process

```xml
<sequence>
  <receive partnerLink="customer" portType="lns:purchaseOrderPT"
    operation="sendPurchaseOrder" variable="PO"
    createInstance="yes" />

  <flow>
    <invoke partnerLink="inventoryChecker" portType="lns:inventoryPT"
      operation="checkINV" inputVariable="inventoryRequest"
      outputVariable="inventoryResponse" />

    <invoke partnerLink="creditChecker" portType="lns:creditPT"
      operation="checkCRED" inputVariable="creditRequest"
      outputVariable="creditResponse" />
  </flow>

  ...

  <reply partnerLink="customer" portType="lns:purchaseOrderPT"
    operation="sendPurchaseOrder" variable="invoice"/>
</sequence>
```
Why Do You Care on BPEL?

• In SOA-enabled environment, you are more likely to compose an application
  – By orchestrating various services via BPEL

• You will probably use BPEL design tool to create a BPEL document
  – NetBeans BPEL designer

• The BPEL document is then processed by BPEL engine
  – Highly likely in JBI enabled runtime platform
DEMO

Building Travel Reservation Composite Application
(Demo Scenario in the next slide)
Demo Scenario: Travel Reservation Business Process

• It receives travel reservation request from a client
  – The request contains travel reservation request XML document based on OTA (Open Travel Association)
• It then performs travel reservation business process talking to three partner web services
  – Airline reservation partner web service
  – Hotel reservation partner web service
  – Vehicle reservation partner web service
• The three partner web services are implemented as EJB based web services
Things to watch (in this demo)

• See Travel Reservation business process as a BPEL document
• See WSDL documents of three partner web services and of the BPEL process web service
• Build and deploy the application over JBI runtime over GlassFish
• Test the application with test requests
JBI (Java Business Integration)
What Is JBI?

- JBI provides a standard application integration framework
- JBI to Application integration is what Java EE is to Enterprise application
Why JBI?

• Point-to-point integration model is not scalable and hard to maintain
• Enterprise Application Integrator (EAI) model has its problems
  – Proprietary integration server
  – Vendor lock-in
• There is a need for an open standard framework for application integration
What Is JBI?

• Standard “meta-container” for integrating “service containers”
  – Service containers host service units
  – Service can be located locally or remotely

• Plug-in architecture
  – 3rd-party service containers can be plugged in

• Message driven
JBI Key Concepts

• **Service Engines (SE)**
  – Service container hosting business logic or system service service unit

• **Binding Components (BC)**
  – Pluggable remote connectivity – they are service containers too whose services are remotely located

• **Normalized message router (NMR)**
  – High efficient in-memory message router
JBI Service Engines (SE) & Binding Components (BC)

- **Service Engines**
  - BPEL SE
  - Java EE SE
  - IEP SE
  - EDM SE
  - SQL SE
  - XSLT SE
  - WLM SE
  - ETL SE

- **Binding Components**
  - HTTP BC
  - SMTP BC
  - MQSeries BC
  - HL7 BC
  - SAP BC
  - JMS BC
  - File BC
  - CICS BC
  - DCOM BC
  - CORBA BC
  - ...
Open ESB Components

The following list of components are currently available at the Open JBI Components project. In this page you can find a component by its function, their solution or by their name. Clicking on a component link will provide details on how they are built, what their current capabilities are and the future plans around these components.

Learn More about Open JBI Components Project
View the JBI Community Wiki's components

By Function
Application Mashup
  AOSD

Aspects
  Aspect SE

Communications
These components provide different methods of communicating with other components of the Enterprise Service Bus.
  ADABAS Natural
  CICS BC
  CORBA BC
  DCOM BC
  File BC
  FTP BC
  HTTP BC
  HTTPS BC
  IMAP BC
  JMS BC
  LDAP BC
  MQ Series BC
  MDB BC
  RMI BC
  RPC BC
  SMTP BC
  SNMP BC
  SWIFT BC
  TCP/IP BC
  UDDI BC
  XMLRPC BC

By Solution
Healthcare
These components provide solutions for the healthcare industry.
  HL7 BC

Mainframe
These components provide mainframe solutions to integration.
  CICS BC
  IMS BC

Telecommunications
These components provide solutions for the telecommunications industry.
  CORBA BC
  SIP BC
  SNMP BC
  XMPD BC

By Name
Alphabetical Listing
If you know the name of the project you are looking for, find it here.

Applications
  Mural

Binding Components
  CICS BC
  CORBA BC
  DCOM BC
  File BC
  FTP BC
  HTTP BC
  HTTPS BC
  JMS BC
  LDAP BC
  MQ Series BC
  RMI BC
  RPC BC
  SMTP BC
  SNMP BC
  SWIFT BC
  TCP/IP BC
  UDDI BC
  XMLRPC BC

Data Conversion
Translating different data formats among systems with varying requirements.
  ETL SE
  XMLT SE
Remote through HTTP BC

- BPEL
- NMR
- HTTP BC
- *Marshall DOM to* `<xml>`
- SOAP/HTTP
- Network layer
Local through NMR

- BPEL ➔ NMR ➔ HTTP BC ➔ Marshall DOM to <xml> ➔ SOAP/HTTP ➔ Network layer ➔ SOAP/HTTP ➔ JAXWS ➔ Unmarshal <xml> ➔ Create DOM ➔ JAXB ➔ WS.helloWorld(name)

Advantages:
> Performance
> Transaction propagation
> Security context propagation
Service Provider Self-Description

Service containers register the services they provide

- Orchestration (BPEL)
- Transformation (XSLT)
- J2EE Platform
- Normalized Message Router
- WS-I Basic SOAP
- AS2
- JMS
- WSDL
- J2EE Platform

JBI Core Services
System Management
OpenESB
What is Project OpenESB?

- Project OpenESB implements an Enterprise Service Bus (ESB) runtime using Java Business Integration (JBI) as the foundation
- Various SE's and BC's are being developed
- It also provides various tools for the development, deployment, and management of composite applications
OpenESB Architecture
NetBeans Support of JBI & OpenESB
Types of SOA “NetBeans” Projects

• When creating a composite application, you typically use the following types of SOA “NetBeans” projects:
  – BPEL Module project
  – IEP (Intelligent Event Processor) Module project
  – EDM (Enterprise Data Mashup) Module project
  – XSLT Module project
  – SQL Module project
  – Composite Application project
  – And more (WLM, ETL, etc.)
Types of SOA “NetBeans” Projects

![NetBeans New Project Wizard](image)

Description:

Creates an empty Composite Application project, which may include multiple BPEL Modules and other types of Java Business Integration (JBI) modules.

Note: For more information, press the Help button and see the online help section on About Composite Application Projects.
Components for v2.1

These are components that can be installed on top of GlassFish ESB v2.1. Some components are already part of the GlassFish ESB v2.1 installer. Newer development-builds (nightly builds) of these components can be found below. These builds may contain the latest bug fixes, but may also contain regressions because they are not fully tested and certified. Of the other components, some are in active development, meaning your mileage may vary.

<table>
<thead>
<tr>
<th>Component</th>
<th>Released (Stable, tested, supported)</th>
<th>Nightly build (Unstable, partially tested, unsupported)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL SE</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>EIP SE</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>POJOells SE</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>WLM SE</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>eMail BC</td>
<td></td>
<td>download nightly build</td>
</tr>
<tr>
<td>File BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>FTP BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>HTTP BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>JMS BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>Scheduler BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>Database BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>Data integrator / ETL SE</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>Data mashup</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>XSLT SE</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>Exec BC</td>
<td></td>
<td>download nightly build</td>
</tr>
<tr>
<td>HL7 BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>LDAP BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>RSS BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>SIP BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>SNMP BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>TCP/IP BC</td>
<td></td>
<td>download nightly build</td>
</tr>
<tr>
<td>UDDI BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>XMPP BC</td>
<td>download v2.1*</td>
<td>download nightly build</td>
</tr>
<tr>
<td>Jini4jics (CICS BC)</td>
<td></td>
<td>download nightly build</td>
</tr>
<tr>
<td>CORBA BC</td>
<td>download 0.7***</td>
<td>download nightly build</td>
</tr>
<tr>
<td>DCOM BC</td>
<td></td>
<td>download nightly build</td>
</tr>
<tr>
<td>EJABC</td>
<td></td>
<td>download nightly build</td>
</tr>
</tbody>
</table>
OpenESB Package Ships Many SE's/BC's
What is a Composite Application?

• Contains a service assembly
  – Each service is captured in the form of a JBI module (BPEL, XSLT, IEP, EDM, etc.)

• A service assembly is deployment unit
  – Individual JBI module cannot deployed by itself
  – Deployed over JBI server

• Test cases can be created

• Service assembly can be configured through CASA
  – Service access protocol (SOAP 1.1, SOAP 1.2, JMS, SMTP, and others) can be configured for each service
CASA Editor

- Composite Application Service Assembly (CASA)
- Graphical tool for configuring service assembly
BPEL Module Project

- BPEL Module project is a group of source files
  - BPEL file
  - WSDL file
  - XML Schema (*.xsd) files
- Will be added to a Composite application as a JBI module
DEMO

Building HelloWorld Composite Application Step by Step

www.javapassion.com/handsonlabs/wscompositeapps/
“Hello World” Composite Application

• A very simple composite application
  – Receives a simple “Hello World” message from its client
    and echoes it back to the same client

• A simple XML schema
  – Same XML schema for describing the XML document
    syntax of both request and response messages

• A simple WSDL document (reflecting the BPEL process to its client)
  – A simple synchronous request/response

• A simple BPEL process
  – Receive->Assign->Reply activities
Steps for Building a Composite App.

1. Create a “BPEL Module” NetBeans project
2. Create XML Schema (as part of BPEL module)
3. Create WSDL document (as part of BPEL module)
4. Create BPEL process (as part of BPEL module)
5. Create a “Composite Application” NetBeans project
   ➢ Add BPEL module
6. Deploy the Composite application
7. Test the Composite application
8. Debug the Composite application
JBI/OpenESB and GlassFish
(GlassFish ESB)
JBI/OpenESB Support in GlassFish

• A JBI runtime has been integrated with GlassFish V2 and after
  – GlassFish admin console now supports JBI

• JBI runtime has been enhanced to adhere to the appserver clustering architecture
  – Each instance in the app server cluster will also have a JBI runtime in it

• GlassFish ESB is product version of OpenESB
  – Comes with pre-configured SE's and BC's
JBI in GlassFish Admin Console
Java EE Service Engine
JavaEE SE

- Ideal place to execute complex business logic
- Bridge between JavaEE container and JBI container
- Provides support for
  - Transactions
  - Resource Pooling
  - Security
- Code re-use – Invoke your EJBs/web applications from OpenESB components (BPEL SE)
Java EE S.E. Use Cases

- BPEL Service Engine calling an Enterprise Java Bean web service
- Message Driven Bean or Servlet calling a BPEL Process
- Enterprise Java Bean web service called through a JMS transport using the JMS Binding Component.
- Java EE components making web service calls through SMTP transport using the SMTP Binding Component
Java EE Use case Scenario
(Loan Processing)
Use case Scenario: Loan Processing

• Loan Requestor Service:
  > LoanRequestProcess
  > WS-I BP
  > BPEL Orchestration
  > LoanProcessor
    > Java EE
  > TransformReport
    > XSLT
  > LoanReportStore
    > Business Partner thru FTP
  > LoanReportMailer
    > Legacy thru JMS
Architecture Refactoring

BPEL Loan Request Service

XSLT Transform Report

JavaEE Loan Processor EJB

NMR

WS-I BP LoanRS WS

JMS ReportMail

File ReportStore

Sun

SAN
DEMO

Building Loan Processing Composite Application

www.javapassion.com/handsonlabs/wscompositeapps/#Exercise_3
www.javapassion.com/handsonlabs/wscompositeapps/#Exercise_4
Demo Scenario: “Loan Processing”

- EJB Web service provides business logic of “approving” or “denying” loan applicant
  - Age should be over 18 years old
  - Yearly income has to be over $20,000
- BPEL process invokes EJB Web service
Intelligent Event Processor (IEP)
Service Engine
What is IEP?

• Models event-driven business process
  – Pretty much all business applications are event-driven
• IEP applications continuously perform
  – Collate and process events from various sources
  – Detect pre-defined patterns & conditions
  – Trigger another event or invoke other internal or external services
Business Processes are Event-driven

- “Automobile accident insurance claim handling business process” adjusting to various events
Business Service Monitoring

- Monitor business services through pattern matching and condition check
  - Track events by user - “User Behavior”
  - Detect spikes in service usage - “System Behavior”
- Automatic response
  - Alerts, Dashboard updates, service triggers, emails, etc.
Real-life Use Cases of IEP

- How many times did Fred login as root for the last 24 hours?
- How many times is “purchase order” business process triggered for the last hour? And what is the average execution time?
- Is a credit card charged for gasoline twice within last one hour?
- Raise an alert when a stock price jump more than 10% relative to its 1 minute moving average price.
- Is the number of JMS messages in the broker increasing over time? What changed?
IEP Designer – Creates IEP

Event process validates successfully.
DEMO

Insider Trading Detection IEP Application

www.javapassion.com/handsonlabs/openesb_iep/#Exercise_2
Demo Scenario: “Insider Trading Detection”

- Monitor a stream of stock transactions, and detects potential insider trades
- Detect "suspicious" stock trades, that is, transactions whose price stands out relative to the stream
  - Compute 24-hour moving average of a stock price
  - If the stock price is 10% above or below the average, place it into the suspicious stock trade list
- Check parties to suspicious transactions against a table of previously identified persons-of-interest
Enterprise Data Mashup (EDM)
Service Engine
What is EDM?

- Provides a unified view of data from heterogeneous sources
- The sources of data can come from
  - Relational databases, flat files, spreadsheets, Web services, RSS/Atom, XQuery RowSet
- Can perform
  - Join'ing/aggregation, cleansing
- Can use external services
  - XLST
Demo Scenario: “Join'ing tables from multiple databases”

- Join data from multiple databases from a single database server (or multiple database servers)
- Modify Join conditions
Building a simple EDM Composite Application

http://www.javapassion.com/handsonlabs/openesb_edm/#Exercise_1
OpenSSO
Agenda

• What is OpenSSO?
• OpenSSO features
  – SSO and Access Control (within an enterprise)
  – Federated Single Sign On (across enterprise boundary)
  – Web Services Security
  – Identity Services
Enterprise SSO Challenges

• Within an organization - We need Single Sign-On (SSO) within an organization
  – “Every application wants me to log in!”
  – “I have too many passwords – my monitor is covered in Post-its!”
  – “We're implementing Sarbanes-Oxley – we need to control access to applications!”

• Outside of an organization - We need Federated SSO across organizations
  – “We need to access outsourced functions!”
  – “Our partners need to access our applications!”
Enterprise Security Use Cases – Within an Enterprise

- A customer logs into a company’s web site and looks for a product in their online catalog
- A manager retrieves employee salary histories to determine an individual’s merit raise
- An employee retrieves his/her own salary information
Enterprise Security Use Cases – Outside of an Organization

- A vendor submits an invoice to the company’s accounting department.
- A corporate human resources administrator accesses an outsourced benefits application.
- An administrative assistant adds a new hire to the corporate database, triggering the company’s health insurance provider to add the new hire to its enrollment.
What is OpenSSO?

• OpenSSO (http://opensso.org/) is an open source project providing core identity functionality such as
  – Single sign-on (SSO) and Access Control
  – Federated SSO
  – Web services security
  – Identity Web services

• Sun OpenSSO Enterprise 8.0, the currently shipping commercial product, is built from OpenSSO
Identity Management Suite

Identity Manager
- Automated Provisioning
- Password Management
- Identity Synchronization
- Identity Auditing

OpenSSO Enterprise
- Single Sign-on/Log-out
- Federation services
- Authorization policies
- Authentication modules

Role Manager
- Role Engineering
- Role Maintenance
- Role Certification
- Identity Compliance

Directory Server
- Directory services
- Virtual directory services
- Security/failover services
- Data distribution services

OpenSSO
- Open Sourced
- Product codebase for Sun OpenSSO Enterprise

OpenDS
- Open Sourced
- Next Generation
- Product codebase for Sun OpenDS SE

3+ Billion Identities Under Management
OpenSSO Architectural Roles

• OpenSSO Server
  – Provides services like Authentication, Authorization, Federation etc.
  – Is contacted by the Policy Agent for these services
  – Comes in a form of a single deploy'able Web application (opensso.war)

• Policy Agent
  – Sits on the application/web server hosting the application that needs to be protected
  – Intercepts requests to protected resource and redirects them to OpenSSO server
SSO & Access Control
(Within an Enterprise)
How SSO Works (Within an Enterprise)

- “Policy agents” are installed to protect web resources (web sites or web-based applications).
- “Policy agents” interact with OpenSSO “policy server” to handle authentication, single sign-on, and authorization requests.
SSO - Initial Login Process

1. Browser sends access request to a protected resource the first time - no SSO-Token is present

2. Agent intercepts the request, and redirects it to OpenSSO server for Authentication

3. OpenSSO server performs authentication and then sends back SSO-Token

4. Agent validates SSO-token and allows access
SSO - Subsequent Access

1. Page request (with SSO-token) to a 2nd protected resource
2. Agent validates the token - no login required
How SSO Works (Within an Enterprise) Again
Federated Single-Sign On (SSO)
Service Outsourcing **Without** Federation (Multi-Login problem)
Service Outsourcing With Federation (Single Sign-On)

Diagram:
- Outsourced Human Resources Services
  - Add Child to Health Plan
- Verify Authentication and Obtain Attributes
- Example Chocolates
- Log In
- Access Web Application
- Employee
- Outsourced Payroll Services
  - Change Tax Status
Important Concepts in Federation

- **Identity Provider (IDP)**
  - Performs authentication, access control

- **Service Provider (SP)**
  - Provides services, resources

- **Circle of Trust (CoT)**
  - A trust relationship exists between its members (IDP's, SP's)
  - Must include at least one IDP

- **Metadata**
  - SAML specifications describing the entities in a standard way
Use Case #1 of Federation

- University now uses Google gmail as their primary mail system
  - Students don't have to carry two email accounts
  - University saves time and resource
- University still maintains the identity information, performs authentication, authorization
  - It plays the role of IDP
  - Google plays the role of SP
- University might use external student loan processing service for their students/alumni
  - Forms a CoT
Use Cases of Federation

- Business organization let its employees to use Google App, SalesForce.com
- Business organization let its employees to manage their 401K through 3rd-party management company
- Business organization let its employees to manage their healthcare through 3rd-party HMO's
Service provider sends SAML authentication request to identity provider via HTTP redirect.

Web User

User is redirected to identity provider. User logs in.

Identity Provider

User is authenticated.

Service Provider
Federated SSO Example (2 of 2)

4. HTML page with a form containing a SAMLAssertion and Javascript to post it to the service provider

5. Form with SAML Assertion is posted to the service provider
Fedlet
What is Fedlet?

• A lightweight Service Provider (SP) implementation which provide quick enablement of service providers

• Support minimal SSO-related needs in business scenarios without the need for a full fledged Federation product deployment
  – Two guys working in a garage “Two-guy-ringtone” providing ring tones to the Telecom company

• Admin at IDP (Identity Provider) can use the OpenSSO console to create a Fedlet zip file
  – Telecom company as a IDP create a fedlet and give it to the “Two-guy-ringtone” company
Fedlet: SP-Initiated SSO
Fedlet: IDP-Initiated SSO
DEMO

Building and deploying a Fedlet

www.javapassion.com/handsonlabs/opensso basics/
Demo Setup

- Installation and configuration of OpenSSO server
  - Single war file - *opensso.war*
  - Simple configuration - only thing you have to provide is admin and agent passwords
  - Embedded DS (Directory Server) is used - no need to configure DS

- Creation of IDP (Identity Provider) in a new CoT
  - IDP performs authentication and access control policy check
  - IDP maintains the user credentials in the embedded DS

- Creation of Fedlet
  - Functions as a front-end SP (Service Provider)
Demo Scenario

- A user access a resource in a SP (Service Provider), which is implemented as a Fedlet
- The SP redirects the request to the IDP for authentication
- A user logs into IDP
- IDP authentications and redirects to SP
- SP allows access
Summary

- OpenESB provides real-life SOA solutions based on standards and open source
- OpenSSO provides real-life SSO solutions for both internal and external Single Sign On needs
Thanks for your attention!

http://www.javapassion.com/soaprogramming
https://open-esb.dev.java.net/
http://www.opensso.org