CICS and Enterprise JavaBeans...
"EJB for Dummies"

Matthew Webster
matthew_webster@uk.ibm.com
Acknowledgments

The following are trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, CICS, CICS/ESA, DB2, MQSeries, OS/390, S/390, WebSphere.

Java, JavaBeans, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names and logos may be trademarks or service marks of others.
Why Java?

Enterprise JavaBeans Overview

Why Enterprise JavaBeans??

Comparison to COBOL

Mixing EJBs and COBOL
An Enterprise JavaBean

Provides one or more logically related business services (called methods)

Each method has a defined set of arguments (COMMAREA copybook)

These entry points are declared in something called an *interface*. 
<table>
<thead>
<tr>
<th>EJB</th>
<th>vs</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known by bean name</td>
<td></td>
<td>Known by program name</td>
</tr>
<tr>
<td>Many entry points, each with own argument list</td>
<td>Single entry point with defined COMMAREA</td>
<td></td>
</tr>
<tr>
<td>Caller uses interface</td>
<td></td>
<td>Caller uses copybook</td>
</tr>
</tbody>
</table>
Remote Interface

```java
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface HelloWorld extends EJBObject {
    public String sayHello( String name )
        throws RemoteException;
}
```
Bean Instances

A "bean" is akin to a load module
  * bytecodes <-> machine instructions
"Instances" of the bean are akin to conversational executions
There can be many instances concurrently
  * pseudo-conversations
Pseudo-conversations

AMCFRED
commarea
Screen
<enter>

AMCBILL
commarea
Screen
<enter>

AMCJOE
commarea

Technology • Connections • Results
Pseudo-conversations (ECI)

Technology • Connections • Results
Bean Instances

Conceptually

Application server

Bean A bytecodes

Bean B bytecodes

Physically

Application server

Bean Store (copies of pseudo-conversational commareas)

Bean A bytecodes

Bean B bytecodes
Each bean instance is identified by an IOR
  * analogous to TSQ name

Given an IOR, a client can call the entry points (methods) of the bean instance it represents

Where do IORs (and hence bean instances) come from?
  * Bean instances are created by "Homes"
Every enterprise bean has a Home.
The Home is a special program
  * With its own "well-known" IOR
And entry points that create bean instances
  * Returns the IOR for the newly created instance
  * Analagous to service creating TSQ names
Pseudo-conversations (EJBs)

Home

```
 instance

-IOR-R

"IOR-H", "create", ...

"IOR-R", OK

-HelloBean

instance

"IOR-R", "sayHello", ...

"IOR-R", "Hi!", ...

-HelloBean

instance

"IOR-R", "remove", ...

OK
```

Technology - Connections - Results
<table>
<thead>
<tr>
<th>ECI</th>
<th>EJB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Tran ID</td>
<td>Home IOR</td>
</tr>
<tr>
<td>First program</td>
<td>Home</td>
</tr>
<tr>
<td>TSQ Name</td>
<td>Remote IOR</td>
</tr>
<tr>
<td>TSQ Items</td>
<td>Instance state</td>
</tr>
<tr>
<td>COMMAREA</td>
<td>IIOP message</td>
</tr>
<tr>
<td>Subsequent TRANs</td>
<td>Remote methods</td>
</tr>
<tr>
<td>Final TRAN</td>
<td>Remove method</td>
</tr>
</tbody>
</table>
Homes

Conceptually

Application server

Bean A
bytecodes

Bean B
bytecodes

Physically

Application server

Bean Store
(copies of
pseudo-conversational
commareas)

Bean A
bytecodes

Bean B
bytecodes

Technology • Connections • Results
The well-known IORs of Homes are published externally
* in a directory server (JNDI)

Clients look up the IORs of Homes using the bean name as a key (usually)
import javax.ejb.EJBHome;
import javax.ejb.CreateException;
import java.rmi.RemoteException;

public interface HelloWorldHome extends EJBHome {
    public HelloWorld create() throws RemoteException, CreateException;
}
Where's the Program Logic??

Developer provides two interfaces (Home and Remote)
Developer also provides enterprise bean class that implements methods on the interfaces
EJB "deployment" tools generate additional "programs" that also implement the interfaces
Where's the Program Logic?

IOR_H

Home
+create()

"EXEC CICS LINK"

Remote
+sayHello(...)
import javax.ejb.*;
public class HelloWorldBean implements SessionBean {
    SessionContext sc;
    public String sayHello( String name ) {
        return "hello " + name + "!";
    }
    ...
}
public void ejbCreate( ) { }
public void ejbRemove( ) { }
public void ejbActivate( ) { }
public void ejbPassivate( ) { }
public void setSessionContext( SessionContext sc ) {
    this.sc = sc;
}

Generated Classes

Generated home and remote programs
  * called "classes"

Intercept requests from client
  * on way from client to enterprise bean
  * on return from enterprise bean to client

Perform EJB container management services
EJB Container Services

Transaction management
Security management
Persistence management
Creation of environment in which bean logic runs
The actions to perform are determined from a side-file called a deployment descriptor
Deployment Descriptor

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN" "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>HelloWorld</ejb-name>
      <home>HelloWorldHome</home>
      <remote>HelloWorld</remote>
      <ejb-class>HelloWorldBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
  ...
</ejb-jar>
Deployment Descriptor

...  
<assembly-descriptor>
  <container-transaction>
    <method>
      <description></description>
      <ejb-name>HelloWorld</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Supports</trans-attribute>
  </container-transaction>
</assembly-descriptor>
</ejb-jar>
EJB Architecture - Client View

- Client
  - create method
  - lookup
- namespace
  - enterprise beans
    - banking
    - accounts
    - funds
- container
  - account enterprise bean
  - ejbCreate
- account
  - home
  - remote

methods intercepted to satisfy deployment specification
EJB Architecture - runtime

Java Client

RMI/IIOP
create
find
remove
methods

EJB Server

EJB Container

ejbCreate
ejbRemove
ejbFind

EJB Home

ejCreate

EJB Instance

business
methods

EJB Object

methods

EJB Jar file

Beans

DD

Technology • Connections • Results
EJB Architecture - Deployment

Application Development

Bean provider

EJB

EJB-Jar

EJB-Jar

Bean provider

EJB

Application assembler

Deployment

EJB Server

D-Jar

EJB

EJB

EJB

EJB

mappings

Deployer

Technology • Connections • Results
Types of Bean

Session Beans
Model tasks - represent a conversation (or session) with a user

Entity Beans
Model resources - provide access to persistent data (typically in a relational database)
Meet the Bean Family...

Enterprise Bean

Session Bean

- Stateless
- Stateful

Entity Bean

- CMP
- BMP

- Anonymous
- Single client
- Non-recoverable

- Primary key
- Shared
- Recoverable

- Pseudo-
- Conversational

Technology • Connections • Results
A Client Program

import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import HelloWorld;
import HelloWorldHome;

public class HelloWorldClient {

    public static void main( String[ ] args ) {
        try {
            Context initial = new InitialContext( );
            Object homeObj = initial.lookup("ejbs/HelloWorld");
            ...
        }
    }
}
A Client Program

```java
...  
    HelloWorldHome helloHome = (HelloWorldHome)  
        PortableRemoteObject.narrow( homeObj,  
            HelloWorldHome.class );

    HelloWorld hello = helloHome.create();
    System.out.println( hello.sayHello( "Matthew" ) );
} catch ( Exception ex ) {
    System.out.println( "Failed: " + ex );
}
```
Compile and Run

javac HelloWorldClient.java

java HelloWorldClient

* Hello Matthew!
Why Java?
Enterprise JavaBeans Overview
Why Enterprise JavaBeans??
Comparison to COBOL
Mixing EJBs and COBOL
Why Enterprise JavaBeans?

Server-side Component Model

*Infrastructure provides* transaction, security and persistence support automatically

*EJB programmer* concentrates on business logic

Exploits modern, common, visual AD tools

Specifies infrastructure support required in deployment descriptor

- 1) Forces clear separation of concerns.
  - Enterprise Bean Provider provides the Business Logic. Is an expert in the application domain. Does not require expertise in system infrastructure.
  - Application Assembler composes application out of off-the-shelf EJBs. Is an expert in the specific requirements of the target businesses. Does not require expertise in system infrastructure.
  - Deployer. Is an expert in the operational environment and deploys/administers the Java Beans and containers without needing detailed knowledge of the application domain.

- This approach facilitates reuse and customization.

- 2) Ease of use
  - Same programming model as client side Java Beans Programming model. Multiple client types are supported (Web, RMI applications, CORBA).

- 3) Infrastructure portability
  - Allows possibility of moving an application developed for one container to another without application code change or recompilation. Containers can be built on different operating systems and EJB can exploit the underlying capabilities (e.g. robustness, scalability, security) of the application deployment platform without change.

- 4) CICS/ESA value add
  - CICS/ESA will support EJB with the robustness, availability, scalability and integrity as for other CICS applications.
  - This includes monitoring, statistics, security and full sysplex enablement.
  - EJBs in CICS/ESA will also have access to the CICS services through CICS Java classes and seamless access to existing applications on the platform without the need for gateways, connectors or adapters.
Why Enterprise JavaBeans?

Can exploit existing Transaction Monitor Infrastructure
  * Transactional capability, security, persistence
  * Portability, Scalability
  * Industry Standard
  * Independent of server platform

1) Forces clear separation of concerns.
   - Enterprise Bean Provider provides the Business Logic. Is an expert in the application domain. Does not require
     expertise in system infrastructure.
   - Application Assembler composes application out of off-the-shelf EJBs. Is an expert in the specific requirements of the
     target businesses. Does not require expertise in system infrastructure.
   - Deployer. Is an expert in the operational environment and deploys/administers the Java Beans and containers without
     needing detailed knowledge of the application domain.

   This approach facilitates reuse and customization.

2) Ease of use
   - Same programming model as client side Java Beans Programming model. Multiple client types are supported (Web,
     RMI applications, CORBA).

3) Infrastructure portability
   - Allows possibility of moving an application developed for one container to another without application code change or
     recompilation. Containers can be built on different operating systems and EJB can exploit the underlying capabilities (e.g.
     robustness, scalability, security) of the application deployment platform without change.

4) CICS/ESA value add
   - CICS/ESA will support EJB with the robustness, availability, scalability and integrity as for other CICS applications.
   - This includes monitoring, statistics, security and full sysplex enablement.
   - EJBs in CICS/ESA will also have access to the CICS services through CICS Java classes and seamless access to
     existing applications on the platform without the need for gateways, connectors or adapters.
Enterprise Perspective

Exploit more productive, modern AD tools
Server "components" for scalable business applications
Ease of programming, reusability, visual composition
Write business logic, not system infrastructure
Clear separation of roles
  * application programmer, container provider, deployer
Java Perspective

Exploit existing transaction processing systems
Exploitation of existing high-end server platforms
Evolutionary development & integration of existing IT investments
Java gains robustness, performance / scalability, security, transaction management, systems management, ......
End to End Architecture

Client-side presentation

Middle-tier view & controller

Business logic (Model)

Client:
pure HTML (browser)
aplets
Java applications

Web (Application)
Server:
servlets, JSPs

Enterprise Application
Server:
EJBs

Technology • Connections • Results
Agenda

Why Java?
Enterprise JavaBeans Overview
Why Enterprise JavaBeans??
Comparison to (CICS) COBOL
Mixing EJBs and COBOL
<table>
<thead>
<tr>
<th>COBOL</th>
<th>EJB</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAREA</td>
<td>Method signature</td>
</tr>
<tr>
<td>LINK</td>
<td>Method calls</td>
</tr>
<tr>
<td>XCTL</td>
<td>-</td>
</tr>
<tr>
<td>START</td>
<td>-</td>
</tr>
<tr>
<td>EXEC CICS</td>
<td>JCICS</td>
</tr>
<tr>
<td>SQL</td>
<td>JDBC</td>
</tr>
<tr>
<td>Embedded SQL</td>
<td>SQLJ</td>
</tr>
<tr>
<td>Compile, Link</td>
<td>Compile, Jar</td>
</tr>
</tbody>
</table>
Performance?

+25%
Agenda

Why Java?
Enterprise JavaBeans Overview
Why Enterprise JavaBeans??
Comparison to (CICS) COBOL
Mixing EJBs and COBOL
Example

Create, Read, Update, Delete, Browse Customer Account Records.

From “Designing and Programming CICS Applications”, Horswill et al. O’Reilly

Presentation Logic

3270 interface (NACT01) -> Print (NACT03)

Business Logic

C,R,U,D (NACT02) -> Error handling (NACT04) -> Name Browse (NACT05)

VSAM

Technology • Connections • Results
Example

Presentation Logic

HTTP → servlets → RMI/IIOP

HTTP → HTML → JSPs → XCTL

Business Logic

Account Mgr

C.R.U.D (NACT02)

Error handling (NACT04)

Name Browse (NACT05)

Technology • Connections • Results
public interface AccountMgr extends EJBOBJECT {
    public AccountDetails getAccount(String accountNo)
        throws RemoteException, NoSuchAccountException;
    public String createAccount(AccountDetails accDetails)
        throws RemoteException;
    public void updateAccount(String accountNo,
            AccountDetails accDetails)
        throws RemoteException, NoSuchAccountException,
            InvalidAccountDetailsException;
    public void DeleteAccount(String accountNo)
        throws RemoteException, NoSuchAccountException;
    public Collection findAccountsByName(String name)
        throws RemoteException;
}
Example

Presentation Logic

HTTP

HTML

XCTL

JSPs

RMI/IIOP

Business Logic

Encapsulated knowledge of file format

Account Mgr

Account

Browse Helper

Technology • Connections • Results
Summary

Why Java - portability, productivity, skills
Enterprise JavaBeans - pseudo-conversations
Why Enterprise JavaBeans??
Comparison to (CICS) COBOL
Mixing EJBs and COBOL
Agenda

Why Java?
Enterprise JavaBeans Overview
Why Enterprise JavaBeans??
Comparison to COBOL
Mixing EJBs and COBOL
Why Java?

Productivity
  * language
  * libraries
  * tooling

Portability

Skills Availability
Java vs COBOL*

Portability
Syntax (arguably)
String support (arguably)
Date and time support
Internationalization support
Data structures (arrays, vectors, hashtables, collections)

*"Java for S/390 and AS/400 COBOL Programmers", Coulthard et al., IBM Press
Java vs COBOL

- Graphical User Interface support
- Object orientation support
- Thread support
- Communications support
- User defined functions (called methods in Java)
COBOL vs Java

Performance (esp. database throughput)
Database access support
Batch update support
File sorting support
Access to other languages