

A20

IMS Connector for Java: Developing Web Applications for Accessing IMS Transactions

Haley Fung



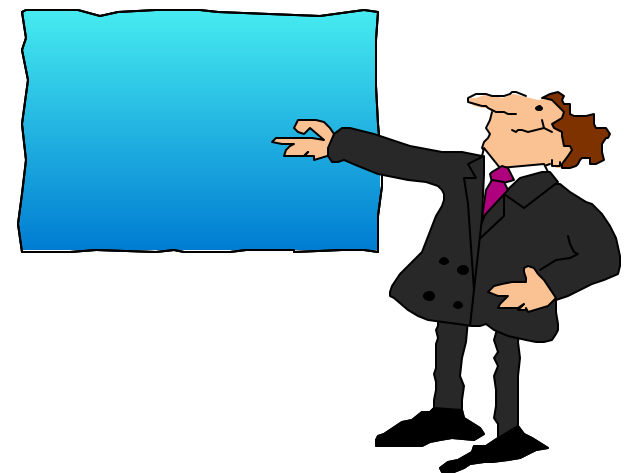
Miami Beach, FL

October 22-25, 2001

Agenda



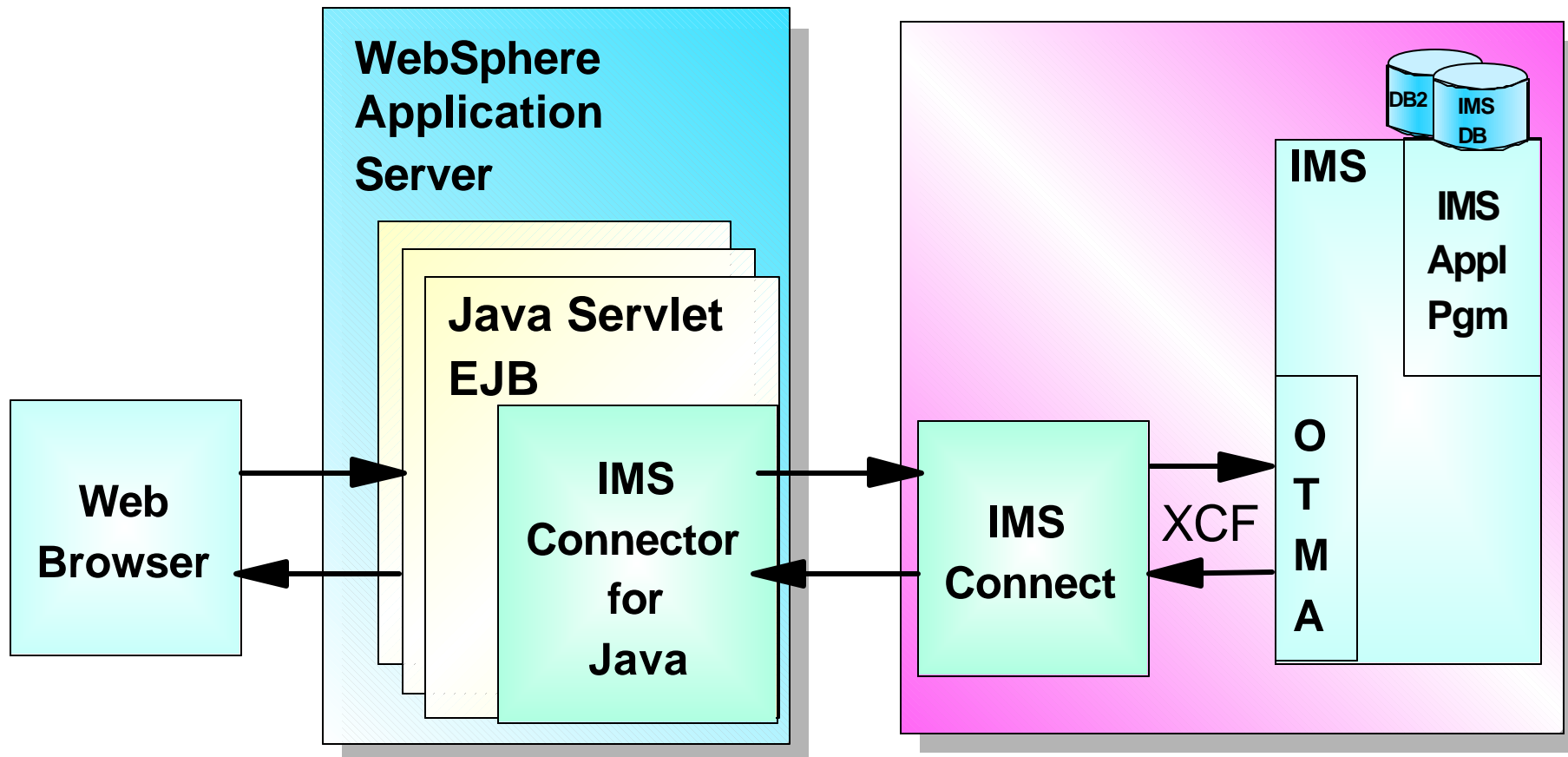
- IMS Connector for Java
- Build, Test and Deploy a J2EE Application to access an IMS Transaction
- Migrating existing CCF applications to J2EE Connector Architecture
- Hints and Tips
- References



IMS Connector for Java



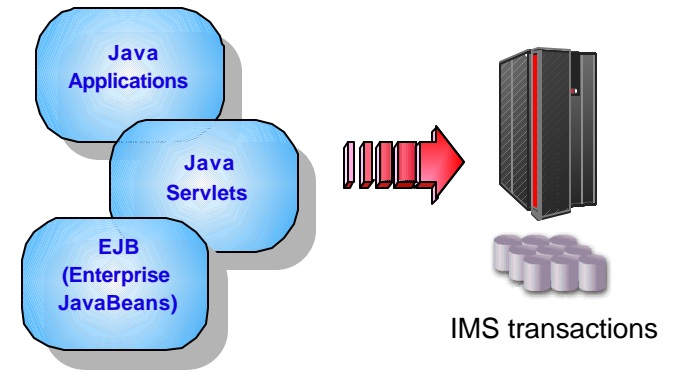
Accessing your IMS transactions from the Web



IMS Connector for Java



Helping IMS Users make the transition to e-business easier



- One of the IBM e-business Connectors (also called WebSphere Adapters)
- Consists of Java components and class libraries which allows Java applications (Java servlets, Enterprise JavaBeans) to submit IMS transactions via IMS Connect
- Implements J2EE Connector Architecture and IBM Common Connector Framework (CCF)

IMS Connector for Java Features



- ★ Implements the J2EE Connector Architecture (IMS Connector is a Resource Adapter) and IBM Common Connector Framework (CCF)
- ★ Allows Java applications to access both IMS non-conversational and conversational transactions from the web
- ★ Supports multi-segment input and output messages
- ★ Communicates with IMS via IMS Connect using TCP/IP or enhanced local connection (Local Option support)
- ★ Provides rapid client application development by integrating with IBM VisualAge for Java and WebSphere Tools

J2EE Platform



Client Tier

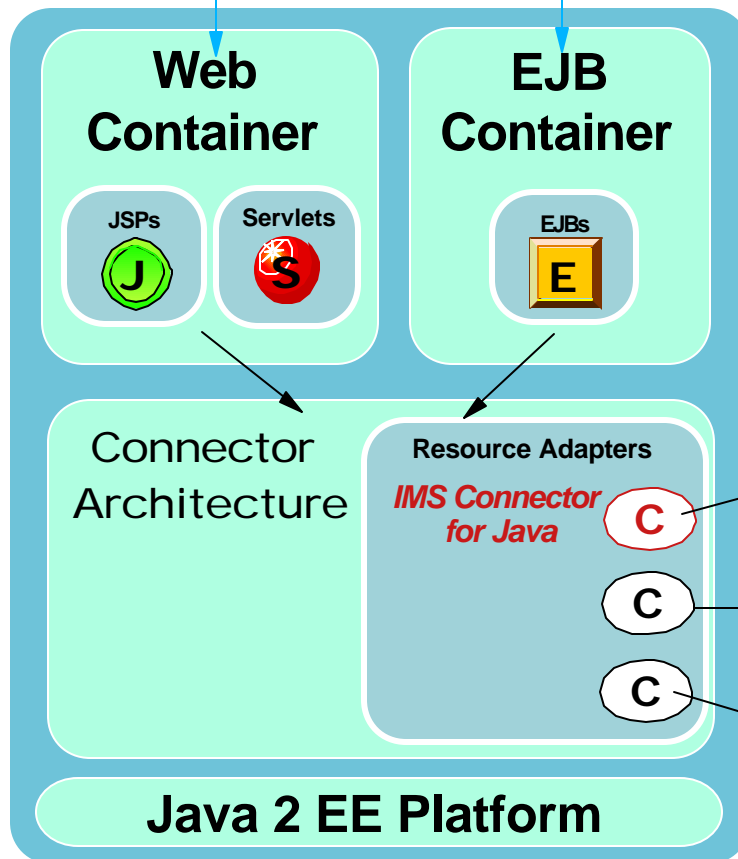
Client-Side Presentation



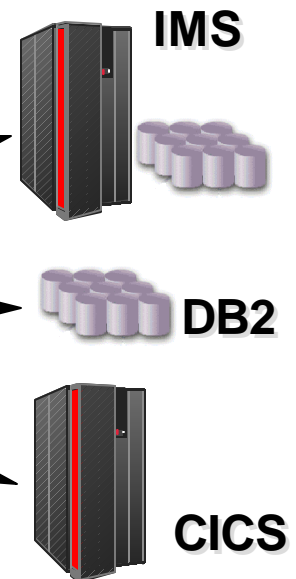
Middle Application Tier

Server-Side Presentation

Server-Side Business Logic



Enterprise Information System Tier

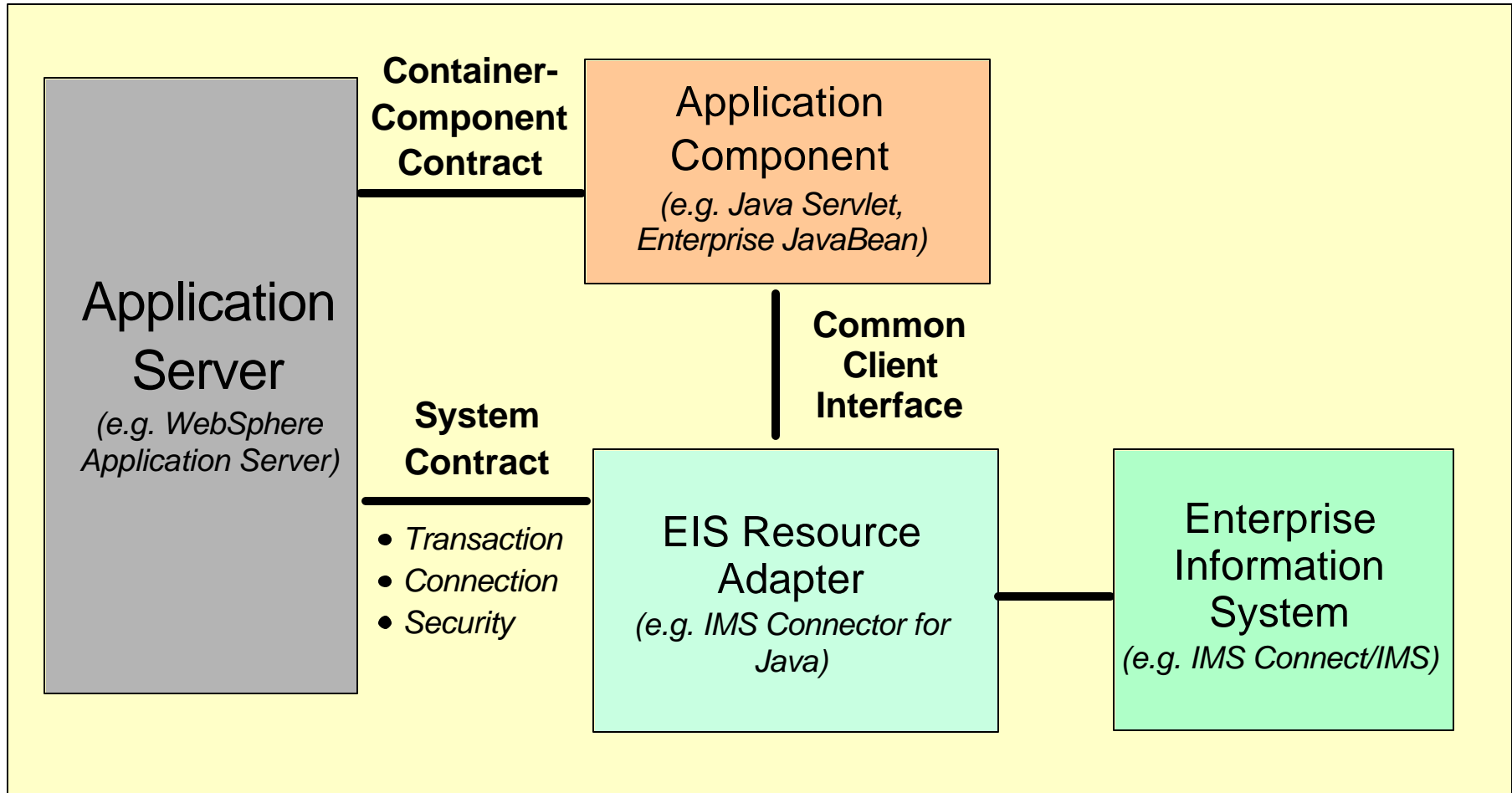


Java 2 EE Platform

WebSphere Application Server



J2EE Connector Architecture



Application Code with CCI



```
// Find ConnectionFactory in JNDI and create Connection
Context initialContext = new InitialContext();
ConnectionFactory connectionFactory =
(ConnectionFactory)initialContext.lookup("java:comp/env/MyIMS");

Connection connection = connectionFactory.getConnection(...);

// Create an interaction and an interaction spec
Interaction interaction = connection.createInteraction();
IMSInteractionSpec interactionSpec = new IMSInteractionSpec();

// set interaction spec specific properties
...

// set input value
inputRec.setInParameter1();
...

// Execute a transaction
interaction.execute(interactionSpec, inputRec, outputRec);

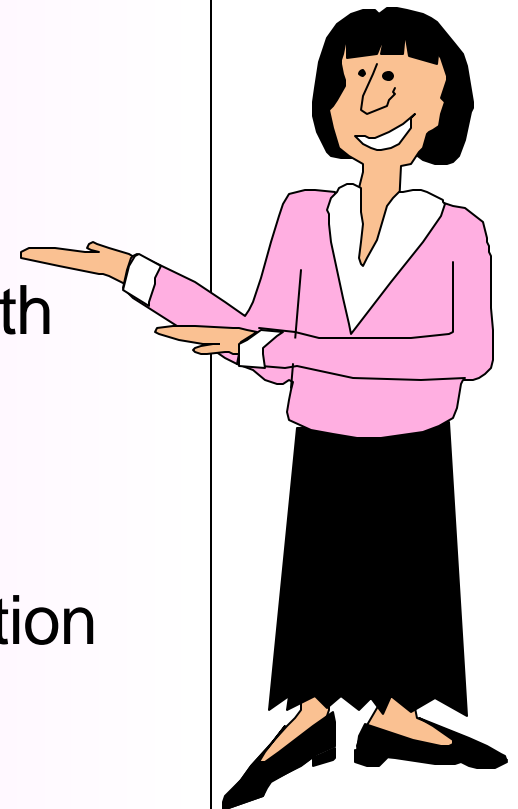
// Close the connection
connection.close();
```


Developing with IMS Connector

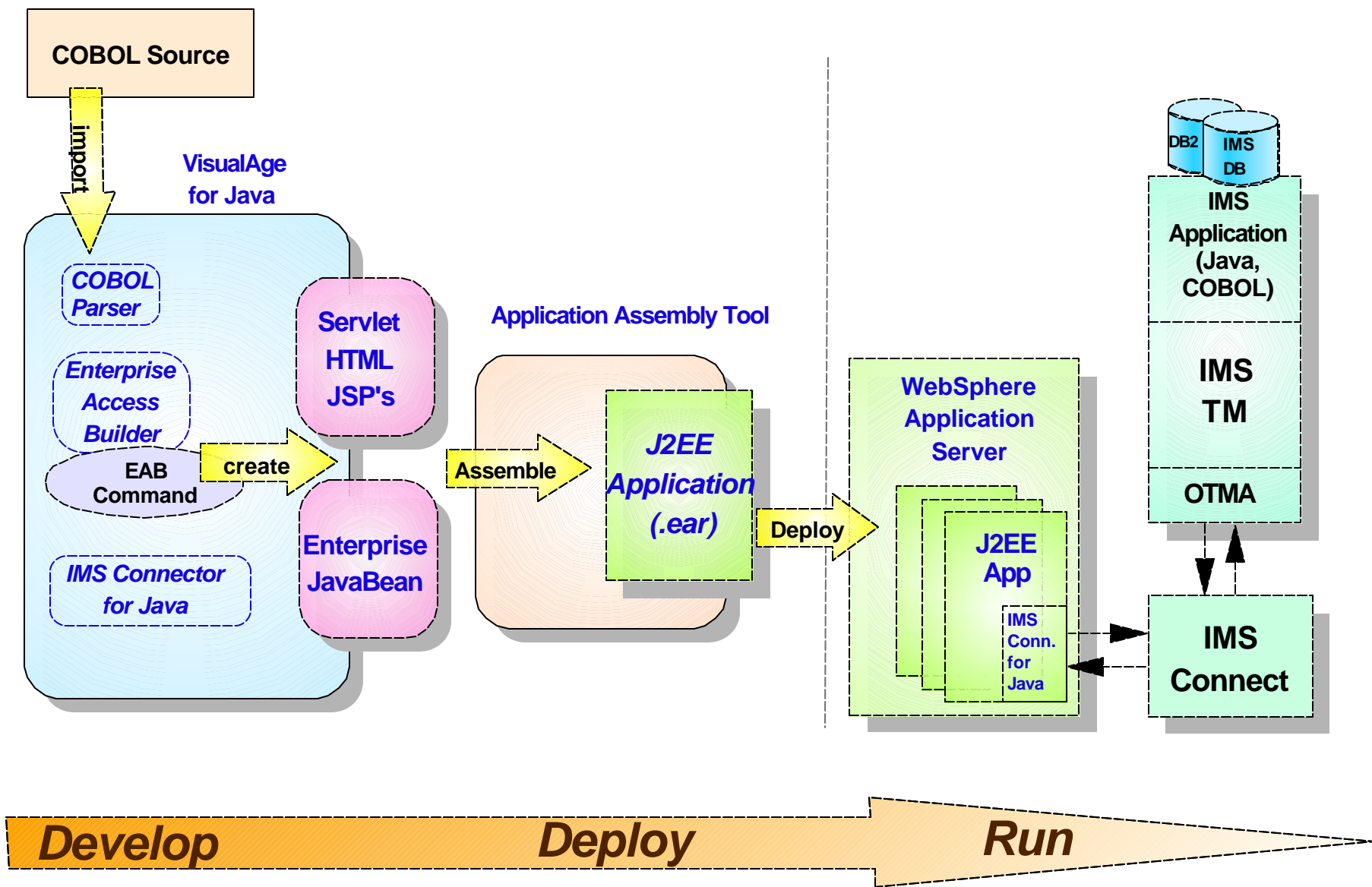


4 Easy Steps

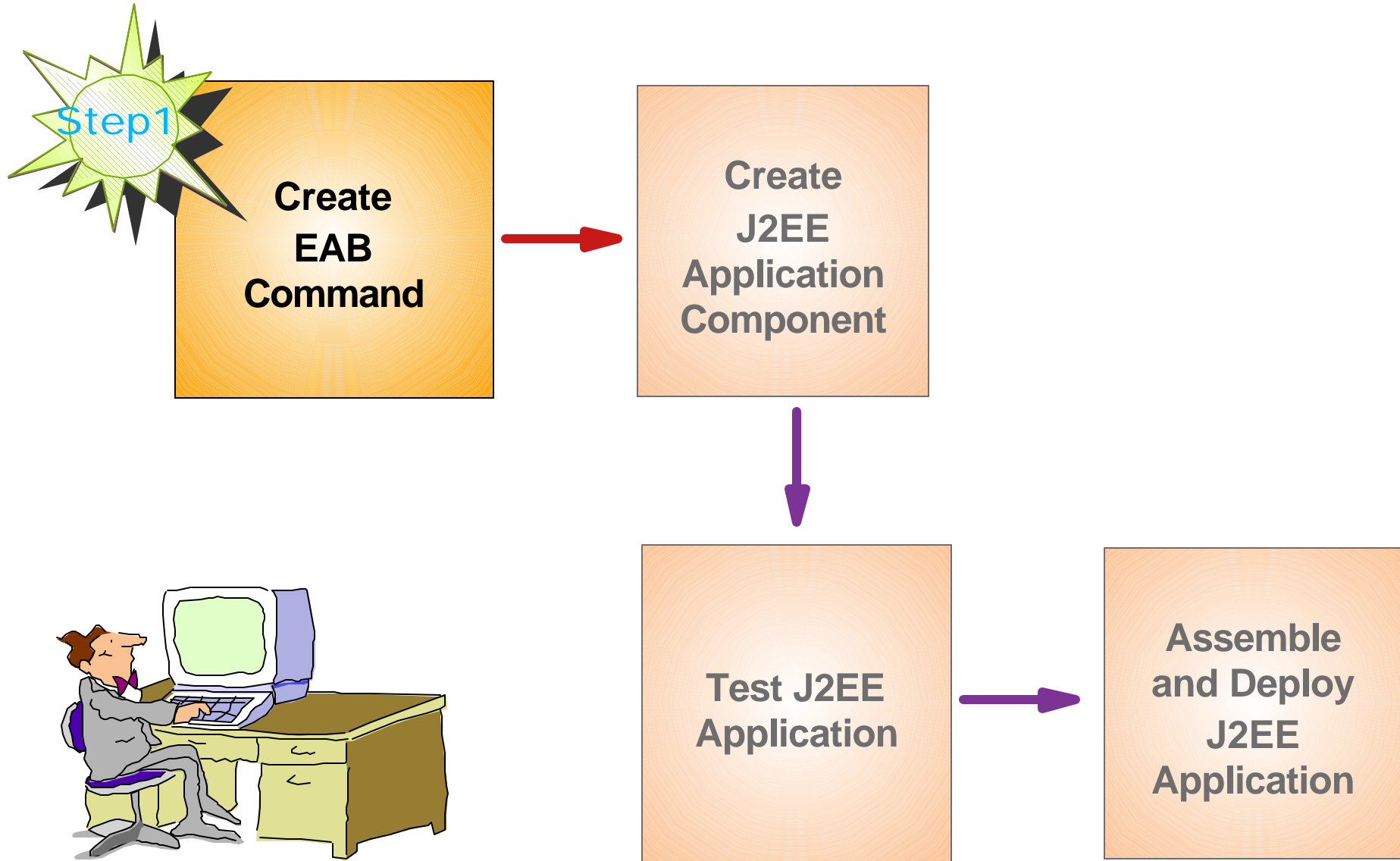
1. Create an EAB command representing the IMS transaction
2. Use the EAB command to create a J2EE application component (e.g. Java servlet, Enterprise JavaBeans)
3. Test and Debug your J2EE application with VisualAge for Java WebSphere Test Environment
4. Assemble and Deploy your J2EE application to your WebSphere Application Server environment



Developing with IMS Connector



Building a J2EE Application

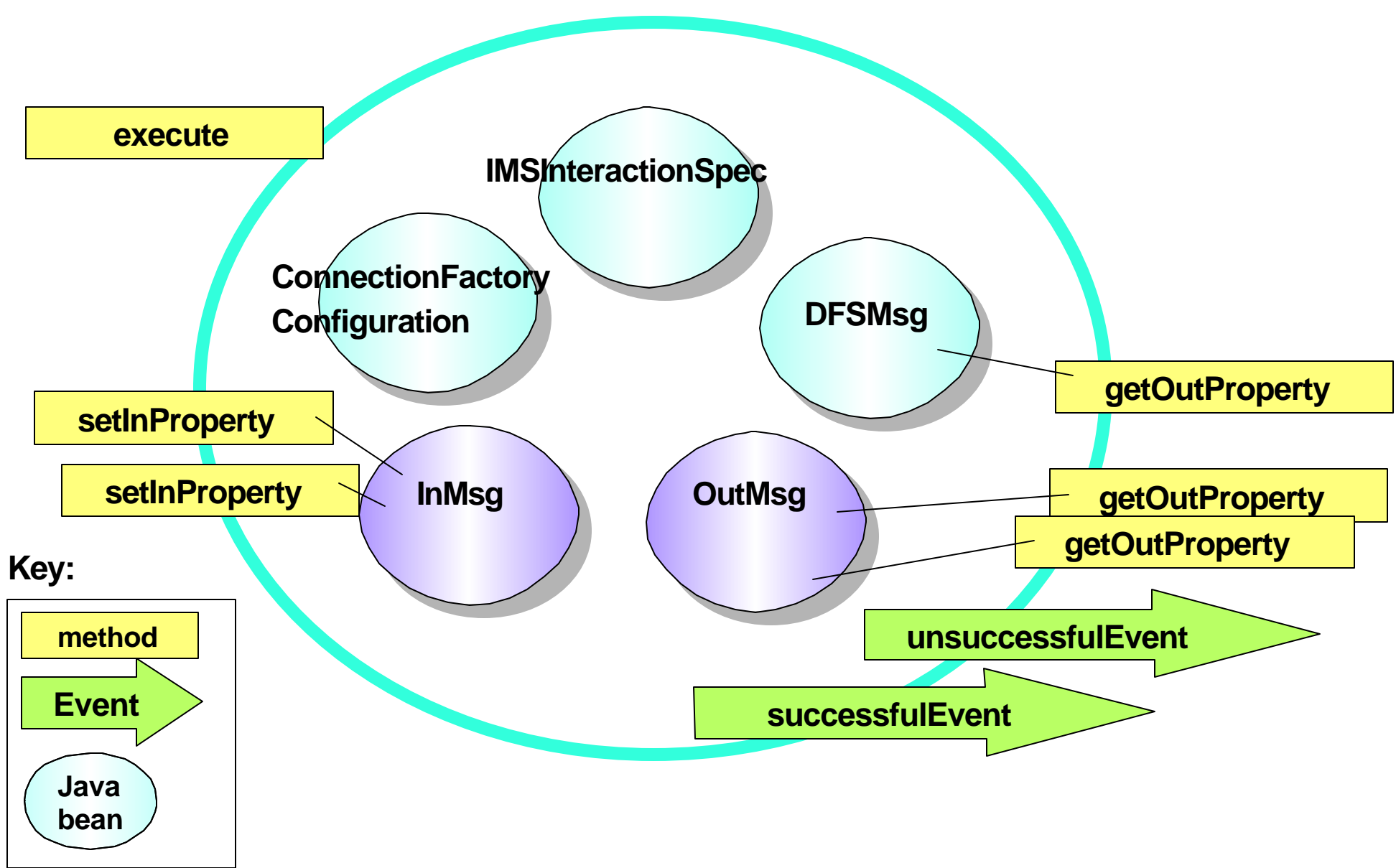


Step 1: Create EAB Command



- An EAB command represents an interaction with a back-end system such as IMS
 - ▶ A typical interaction would be to send transaction input data to an IMS application program and receive the transaction output data back from IMS
 - ▶ Contains the connection and interaction properties and the input and output messages that represent an interaction with the backend system

EAB Command



Application Code with EAB



```
// Instantiate EAB Command
Ex01Command ex01Cmd = new Ex01Command();

// Setup input properties
ex01Cmd.setInProperty1();
...

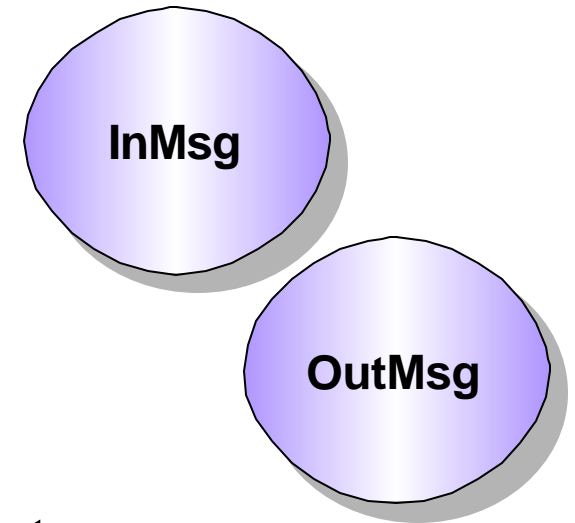
// Execute a transaction
ex01Cmd.execute();

// Get the output data
ex01Cmd.getOutProperty1();
...
```

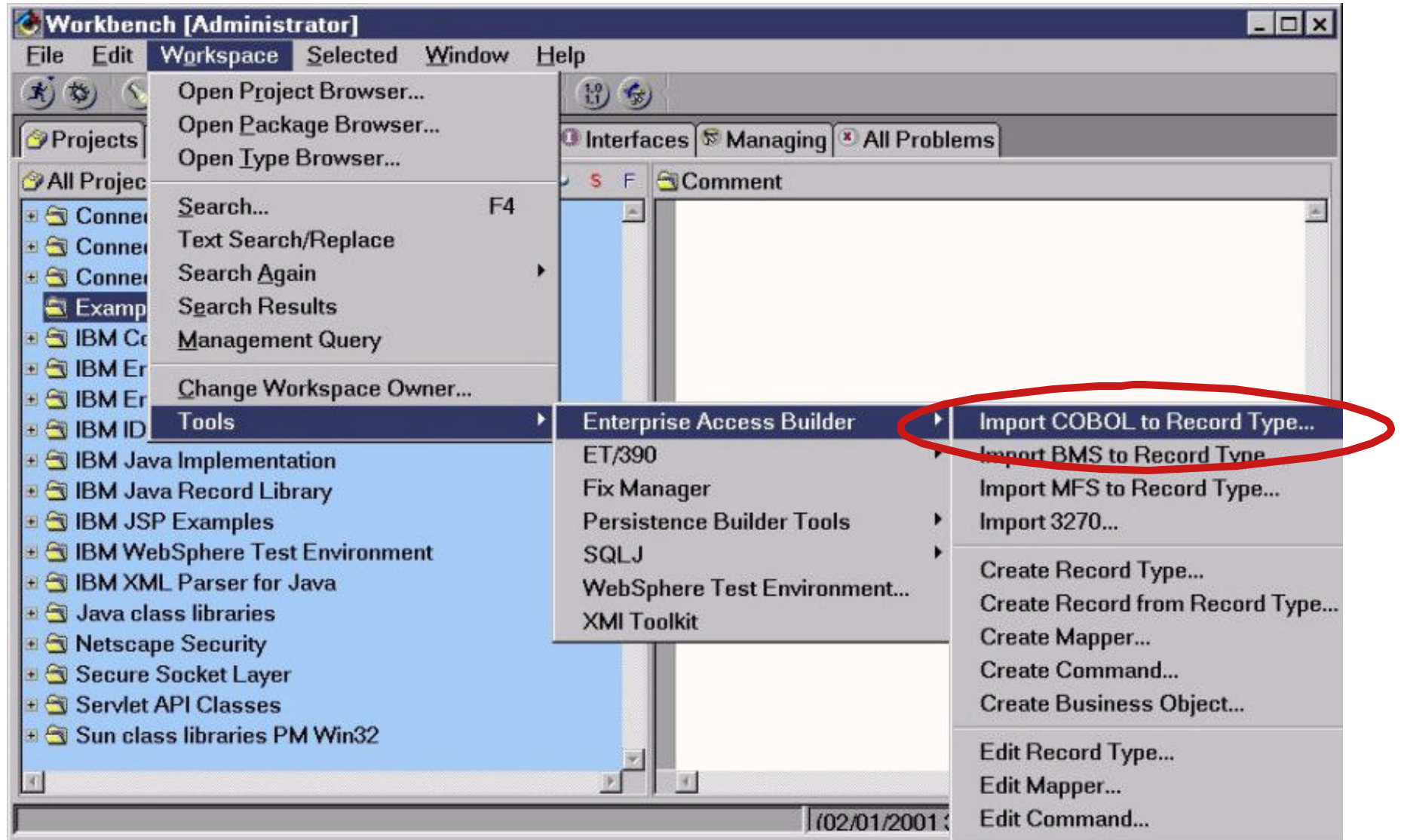
Create IMS Transaction Message Beans



- Create Java Record beans representing the IMS transaction's input message and its output message(s)
- Created using VisualAge for Java's Enterprise Access Builder tool
- Create beans from COBOL data structures
 - ▶ Use the data structures (01 commareas) for the I/O PCB input/output area descriptors



Create IMS Transaction Message Beans



Create IMS Transaction Message Beans



SmartGuide [X]

Import COBOL to Record Type

Enter the COBOL file containing the commareas to import:

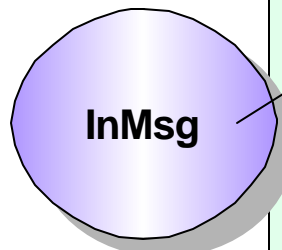
COBOL File:

Code to be imported is:

- A CICS Transaction
- An IMS Application Program
- Generic COBOL code

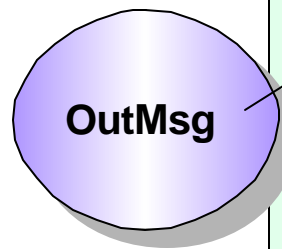


Create IMS Transaction Message Beans



01 INPUT-MSG.

```
02 IN-LL          PICTURE S9(3) COMP.
02 IN-ZZ          PICTURE S9(3) COMP.
02 IN-TRCD        PICTURE X(10).
02 IN-CMD         PICTURE X(8).
02 IN-NAME1       PICTURE X(10).
02 IN-NAME2       PICTURE X(10).
02 IN-EXTN        PICTURE X(10).
02 IN-ZIP         PICTURE X(7).
```



01 OUTPUT-MSG.

```
02 OUT-LL         PICTURE S9(3) COMP VALUE +0.
02 OUT-ZZ         PICTURE S9(3) COMP VALUE +0.
02 OUT-MSG        PICTURE X(40) VALUE SPACES.
02 OUT-CMD        PICTURE X(8) VALUE SPACES.
02 OUT-NAME1      PICTURE X(10) VALUE SPACES.
02 OUT-NAME2      PICTURE X(10) VALUE SPACES.
02 OUT-EXTN       PICTURE X(10) VALUE SPACES.
02 OUT-ZIP        PICTURE X(7) VALUE SPACES.
02 OUT-SEGNO      PICTURE X(4) VALUE SPACES.
```

Create IMS Transaction Message Beans



SmartGuide [X]

Import COBOL to Record Type [Icons]

Select commareas to import.

Available level 01 commareas:

- INPUT-MSG
- OUTPUT-MSG

Selected commareas:

Specify non-level 01 commarea:

Use BigDecimal

Add transaction code field

Field Name:

Length: Value:

< Back Next > Finish Cancel



Create IMS Transaction Message Beans



SmartGuide [X]

Import COBOL to Record Type

Project:

Package:

Class name:

Continue working with newly created record type...

Edit record type

Create record from record type



Create IMS Transaction Message Beans



SmartGuide [X]

Create Record from Record Type

Project:

Package:

Class name:

Select generation options:

Access Method: Direct Hierarchical

Record Style: Dynamic Records Custom Records

Additional Options:

Generate with Notification

Use Inner Classes

Shorten Names

Create IMS Transaction Message Beans



SmartGuide

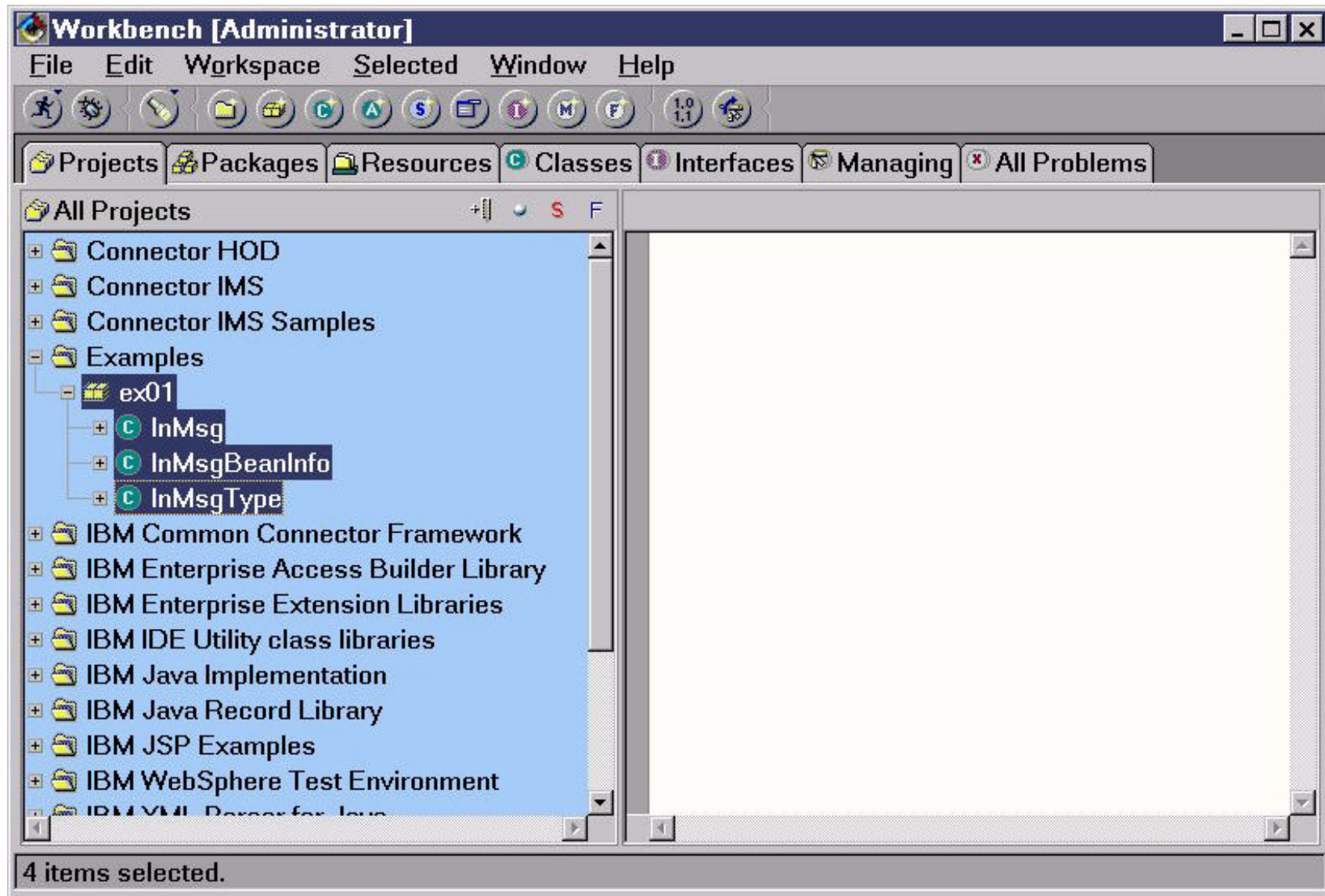
Create Record from Record Type

Change properties of the Record Attributes Bean

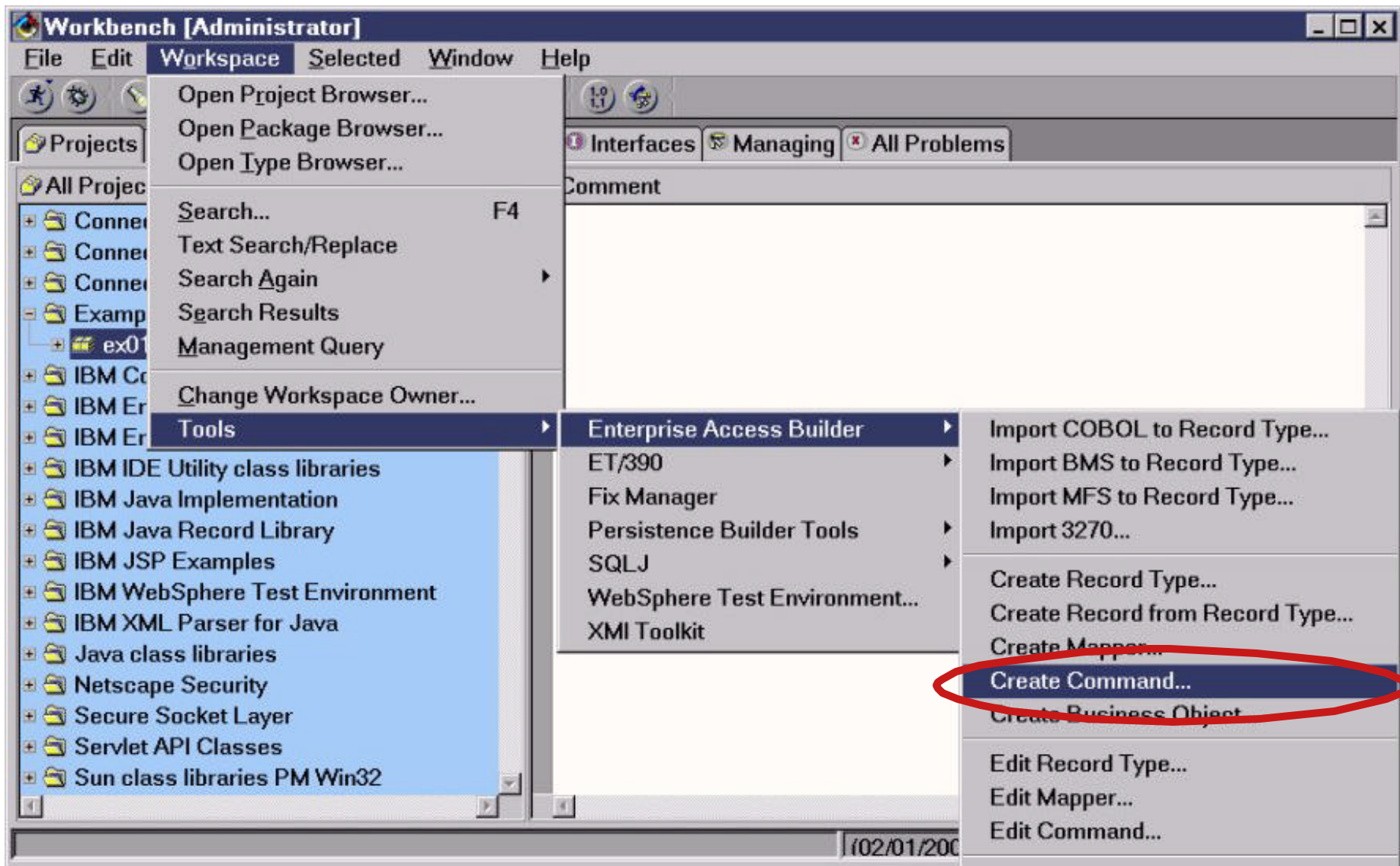
Property	Value
Floating Point Format	IBM
Remote Integer Endian	Big Endian
Endian	Big Endian
Code Page	037
Machine Type	MVS

< Back Next > Finish Cancel

Create IMS Transaction Message Beans



Create EAB Command



Create EAB Command



Connection
Factory
Configuration

IMSInteraction
Spec

SmartGuide

Create Command

Project: Browse...

Package: Browse...

Class name:

Edit when finished

Connection Information:

Class name: Browse...
Edit...

InteractionSpec:

Class name: Browse...
Edit...

< Back Next > Finish Cancel

ConnectionFactoryConfiguration



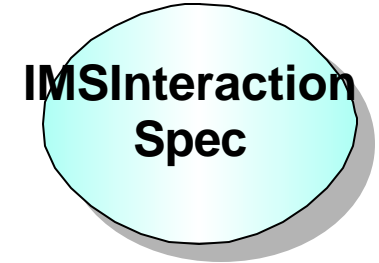
- EAB provides ConnectionFactoryConfiguration class which defines the connection and security properties between the J2EE component and backend EIS (e.g. IMS/IMS Connect)
- In a managed case, specify the JNDI reference name of a ConnectionFactory
- In a non-managed case, specify the properties of the ConnectionFactory (e.g. hostname, port number, etc)

Connection
Factory
Configuration

IMSInteractionSpec



- Defines interaction between J2EE component and IMS accessed via IMS Connect and OTMA
- IMS Connector for Java properties:
 - ▶ InteractionVerb
 - SYNC_SEND_RECEIVE
 - SYNC_END_CONVERSATION
 - ▶ Map Name
 - ▶ LTerm Name



Create EAB Command



InMsg

OutMsg

DFSMsg

SmartGuide

Add Input/Output Beans

Input record bean:

- Implements javax.resource.cci.Record
- Implements ByteBuffer

Class name: Browse...

Mapper class: Browse...

Output record beans:

- Use input bean type as output bean type
- Select output record beans:

Output	Mapper
OutMsg	
DFSMsg	

Buttons: Add... Modify... Delete

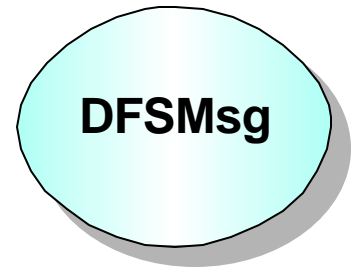
ex01.InMsg

Navigation: < Back Next > Finish Cancel

DFSMsg



- Represents IMS "DFS" messages
- "DFS" messages are always a possibility instead of transaction output
- At runtime, the Enterprise Access Builder populates either the EAB command's DFSMsg bean or (one of) the transaction output bean(s)



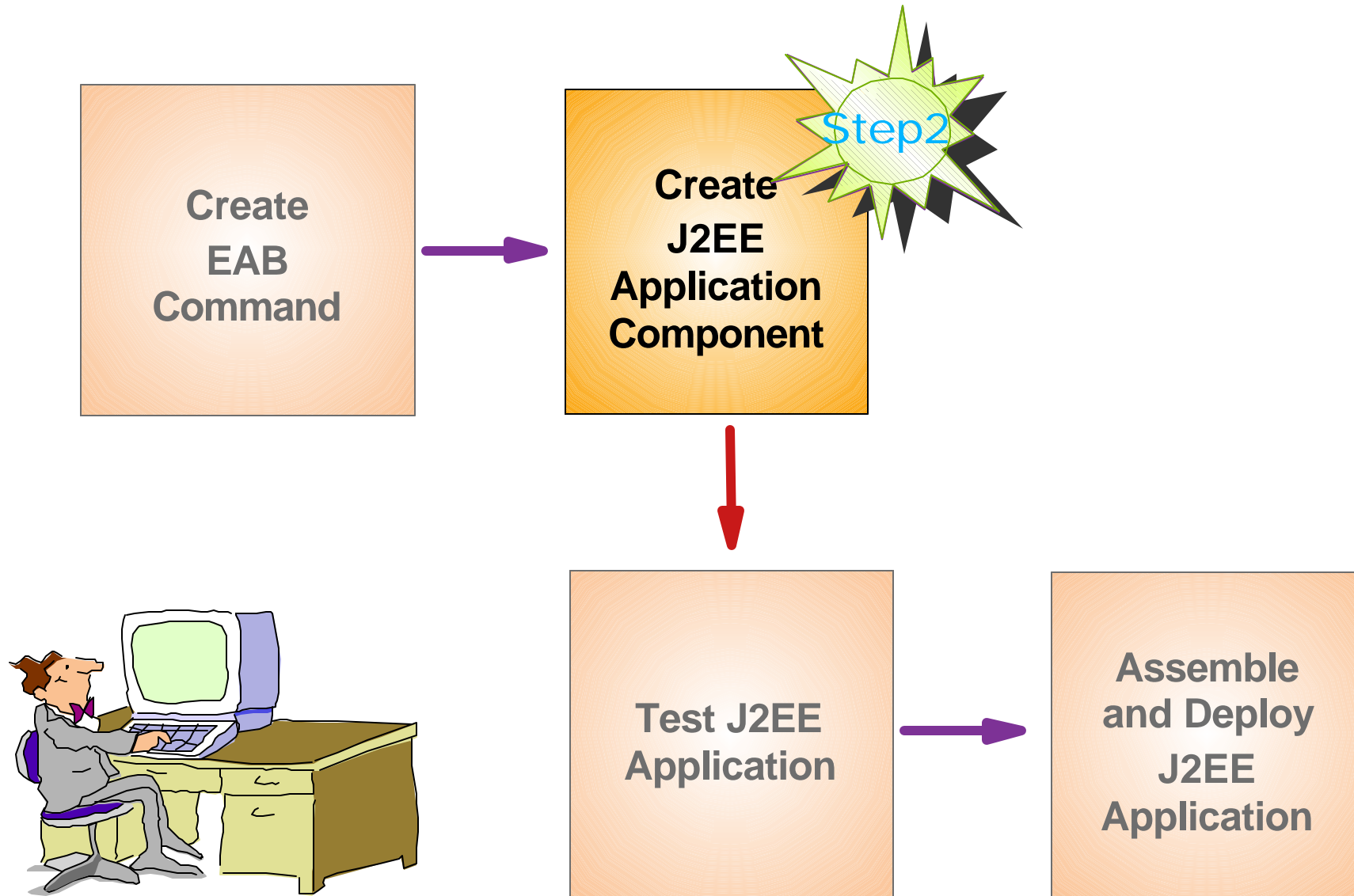
Create EAB Command



The screenshot shows the 'Command Editor' application. The left pane displays a tree view for 'ex01.Ex01Command' with sub-items 'Connector', 'Input', and 'Output'. The right pane shows 'ex01.InMsg'. Below these panes is a table with two columns: 'Property' and 'Value'. The 'Property' column lists various fields like 'IN_NAME1', 'IN_CMD', 'startingOffset', etc. The 'Value' column contains values like 'byte[59]' and 'com.ibm.record.CustomRecordType'. A context menu is open over the 'IN_NAME1' property, with 'Promote Property' highlighted by a red circle.

Property	Value
IN_NAME1	
IN_CMD	
startingOffset	
IN_EXTN	
IN_TRCD	
recordAttributes	ij.eab.record.cobol.CobolRecordAttributes
IN_LL	
bytes	
alignmentOffset	
IN_ZIP	
rawBytes	byte[59]
recordType	com.ibm.record.CustomRecordType

Building a Web Application

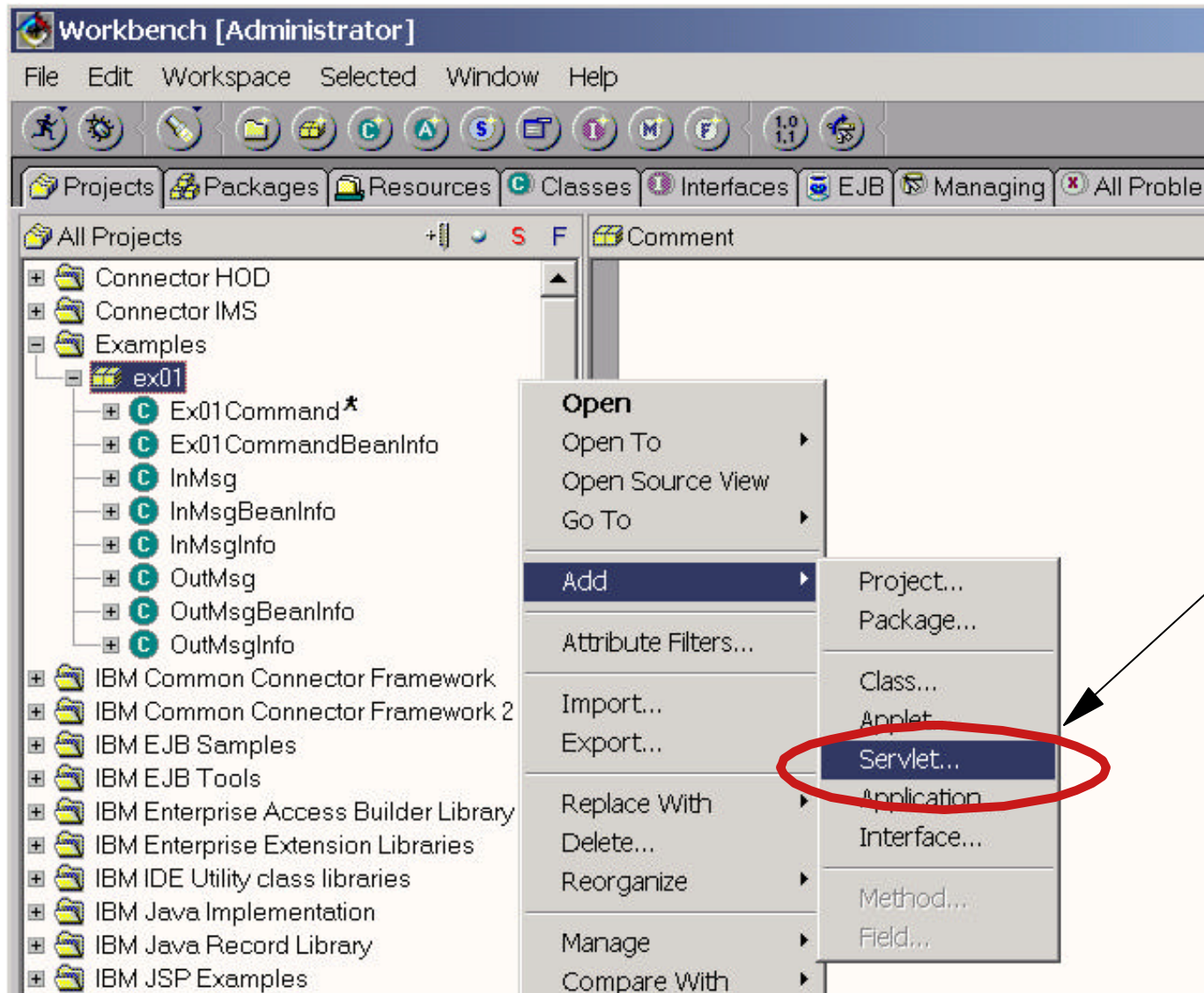


Step 2: Create J2EE Application Component



- Use VisualAge for Java's tool to create the following J2EE application component from an EAB command:
 - ▶ A Web Module
 - A Web Application that normally contains HTML, Java Servlets and JSP
 - ▶ A EJB Module
 - Enterprise JavaBeans

Create a Web module



Create
Servlet
Wizard

Create a Web module



SmartGuide [X]

Create Servlet

Project:

Package:

Class name:

Superclass:

Advanced Options

Inherit from PageListServlet Import Java bean

Use Single Thread Model

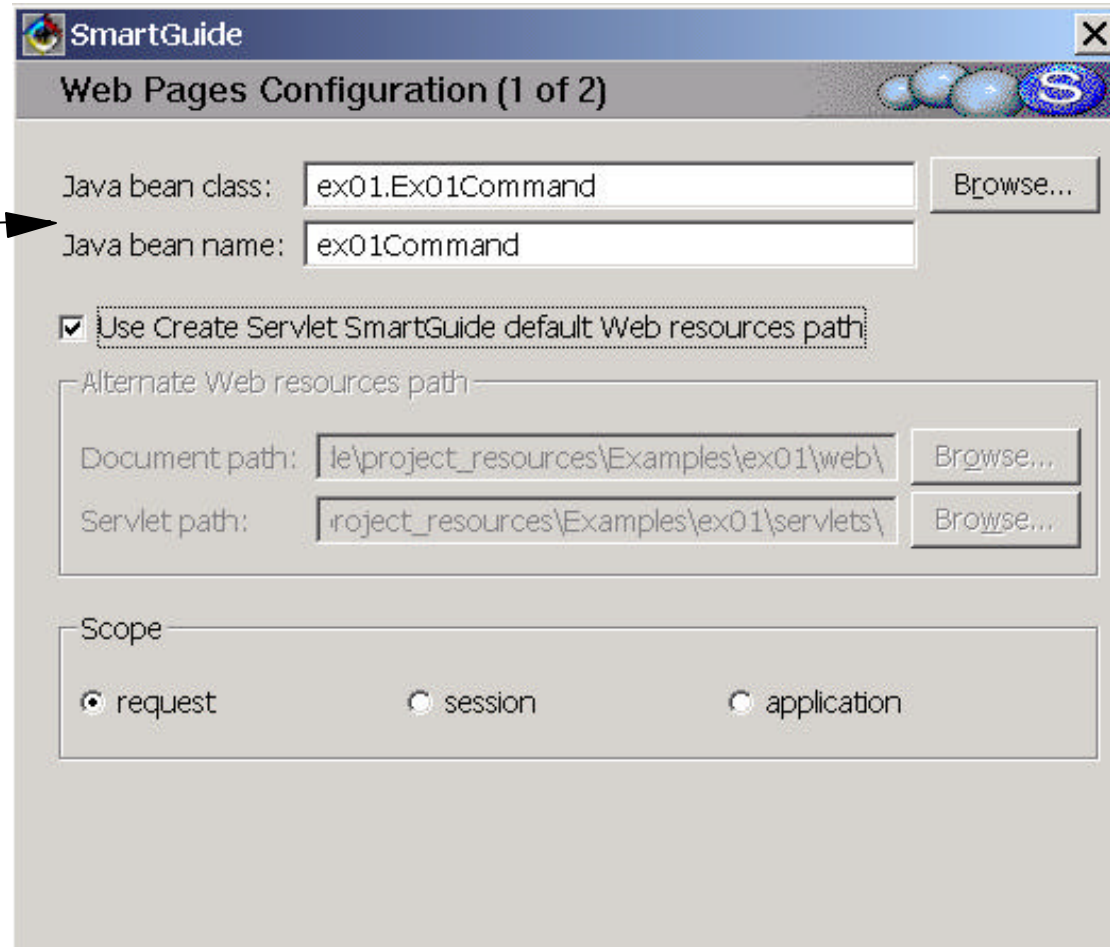
Add import statements:

Which interfaces should this class implement?

Create a Web module



**EAB
Command**



SmartGuide
Web Pages Configuration (1 of 2)

Java bean class: Browse...

Java bean name:

Use Create Servlet SmartGuide default Web resources path

Alternate Web resources path

Document path: Browse...

Servlet path: Browse...

Scope

request session application

Create a Web module



Select the promoted input properties of the EAB command to be displayed on the input HTML page

Select the promoted output properties of the EAB command to be displayed on the output JSP page

Select the method of the EAB command to submit the IMS transaction

SmartGuide

Web Pages Configuration (2 of 2)

Fields displayed on input page:

- (java.lang.String) IN__NAME1
- (java.lang.String) IN__NAME2
- (java.lang.String) IN__TRCD
- (java.lang.String) IN__ZIP
- (java.lang.String) IN__EXTN

Fields displayed on results page:

- (java.lang.String) OUT__MSG
- (java.lang.String) OUT__EXTN
- (java.lang.String) OUT__NAME1
- (java.lang.String) OUT__NAME2
- (java.lang.String) OUT__SEGNO
- (java.lang.String) OUT__ZIP

Action methods to call when the server logic runs:

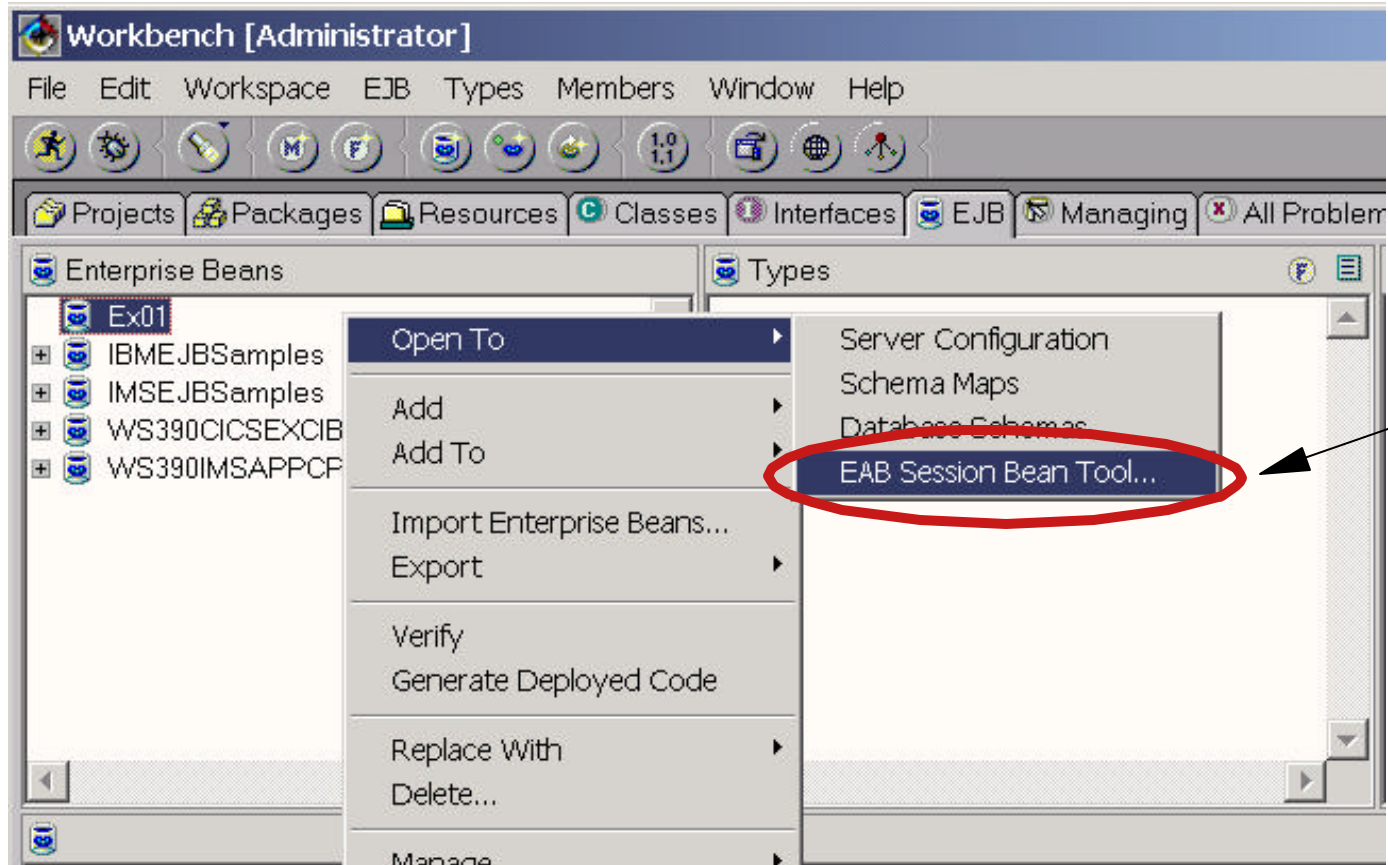
- void execute()

Create a Web module



- Generated Web module files:
 - ▶ Input HTML page (.html) for input of IMS transaction data
 - ▶ Output JavaServer page (.jsp) for dynamic output of IMS transaction data
 - ▶ Output JavaServer page (.jsp) for dynamic output of error data
 - ▶ Java source (.java) for the servlet
 - ▶ Servlet configuration file (.servlet)
 - ▶ Style sheet (Master.css)

Create a EJB Module



EAB
SessionBean
Tool

Create a EJB Module



SmartGuide

Create EAB Session Bean

Project: Browse...

Package: Browse...

Class name:

Edit when finished

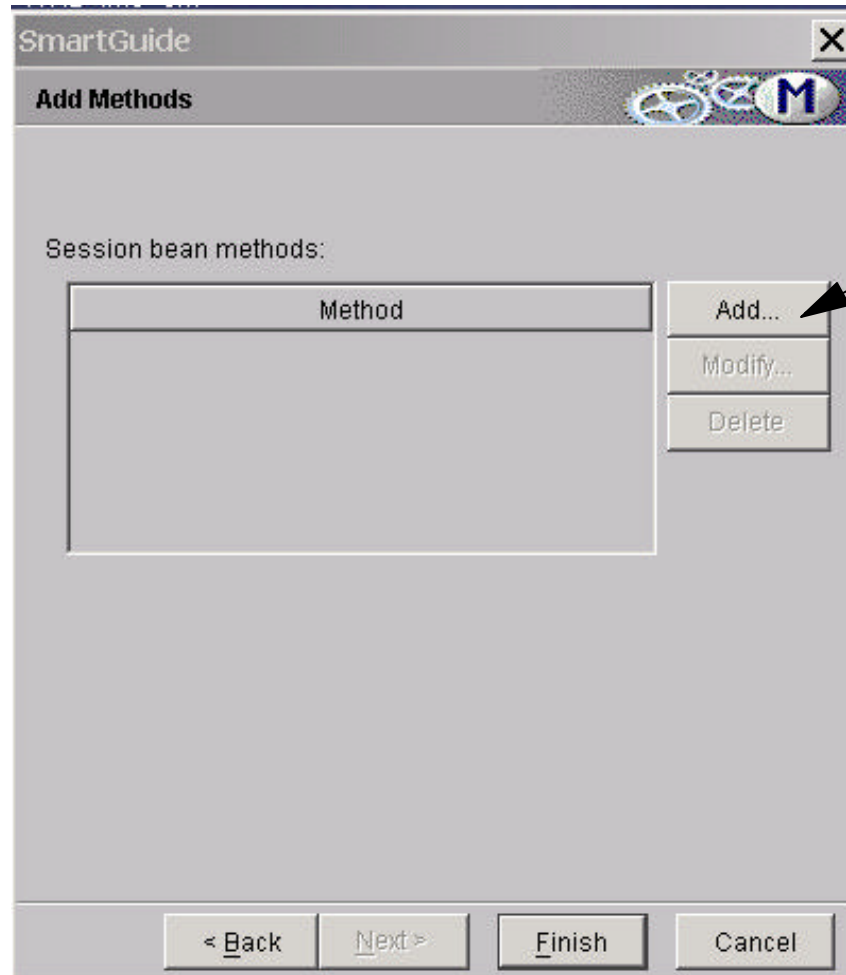
Connection Information:

Class name: Browse...
Edit...

com.ibm.ivj.eab.command.ConnectionFactoryConfiguration

< Back Next > Finish Cancel

Create a EJB Module



Add EJB method

Create a EJB Module



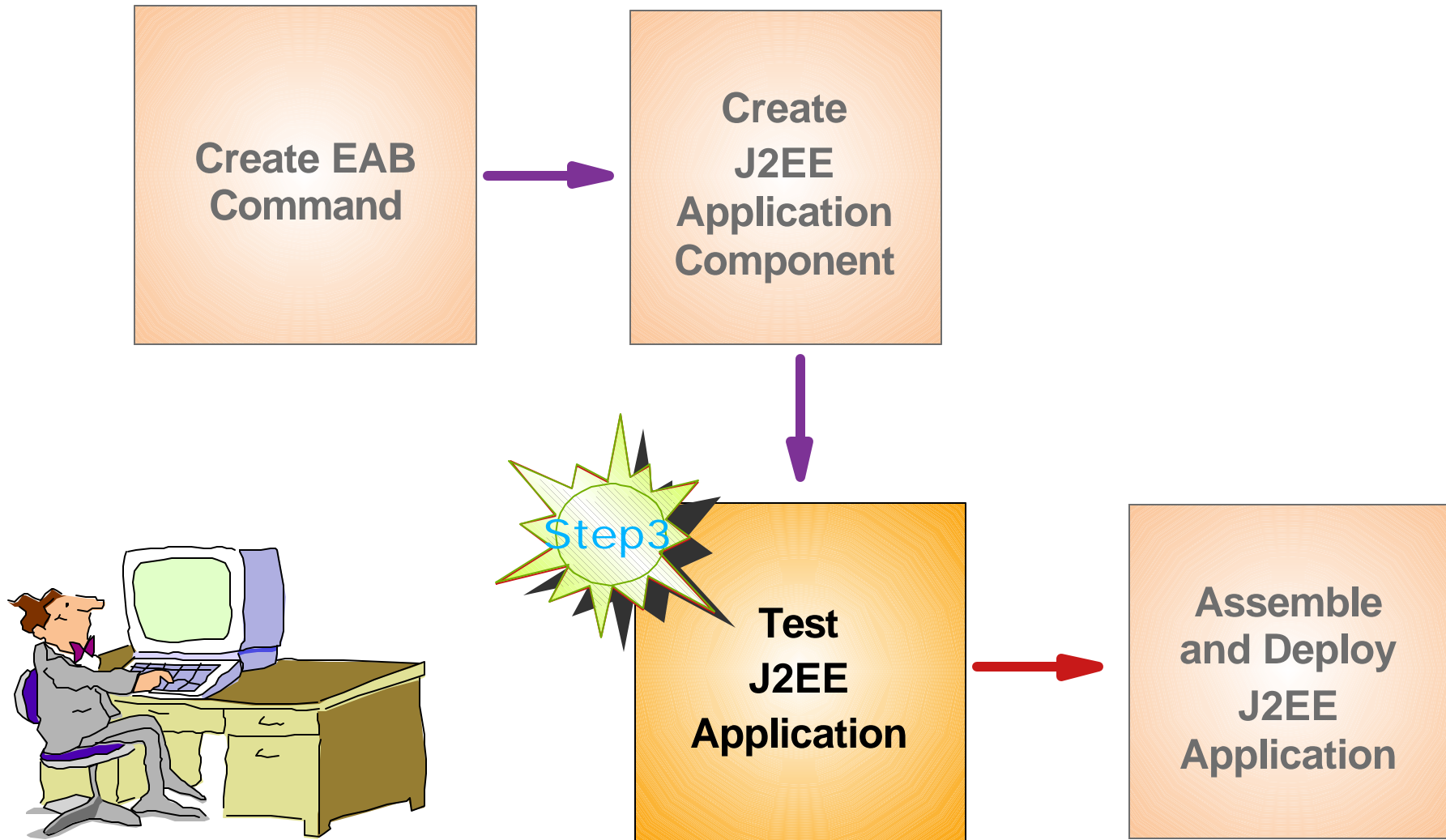
**EAB
Command**

Specify an existing EAB command that represents the method logic
Or you can built the method logic with this panel

The SmartGuide dialog box is titled "SmartGuide" and contains the following fields and options:

- Method name:
- Method implementation:
 - Use existing command:
 - Class name:
 - Specify method logic:
 - InteractionSpec:
 - Class name:
 -
 - Input record bean:
 - Implements javax.resource.cci.Record
 - Implements ByteBuffer
 - Class name:
 -
- Output record beans:
 - Use input bean type as output bean type
 - Select output record beans:
 - | Output |
|--------|
|--------|
 -
 -

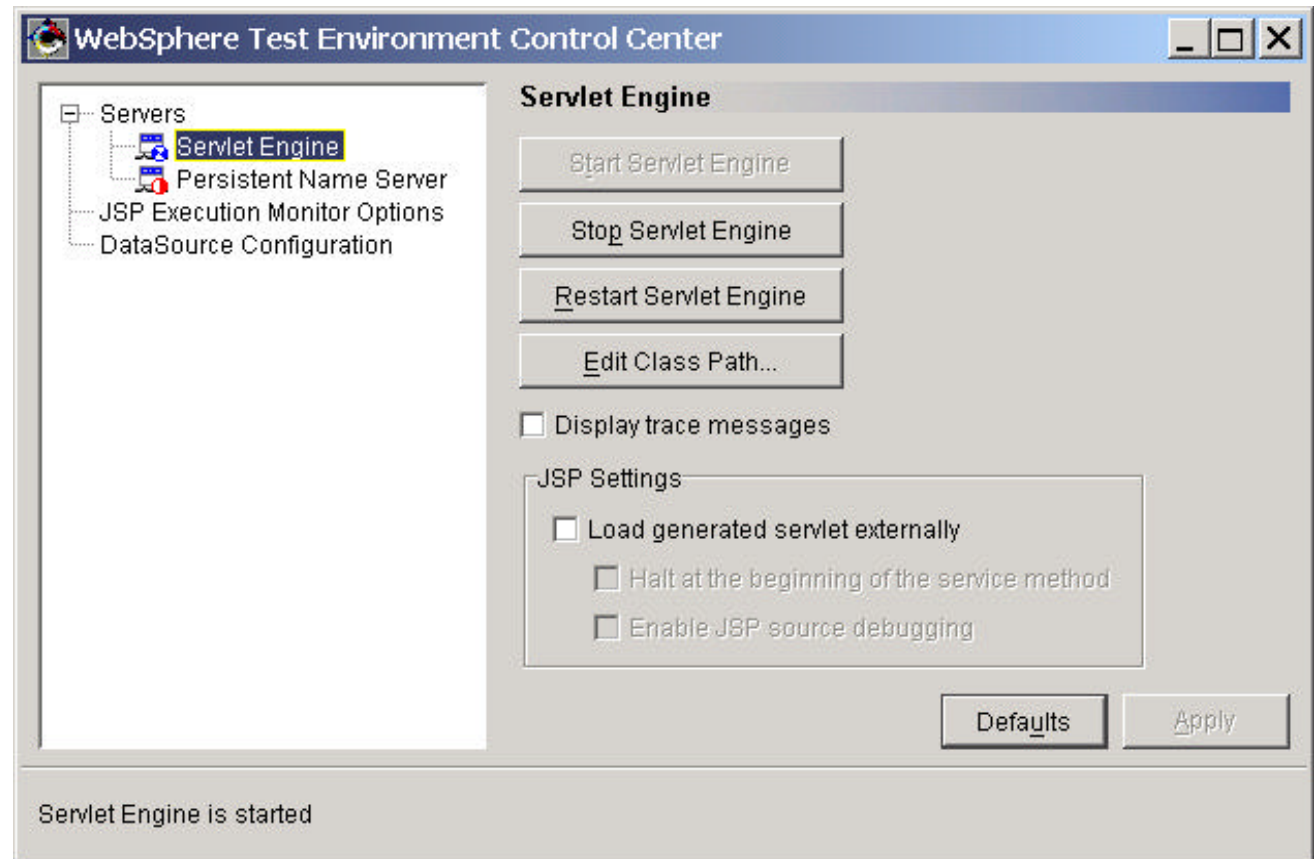
Building a J2EE Application



Step 3: Test J2EE Application



- Test the J2EE application using VisualAge for Java's WebSphere Test Environment
- Enable easy testing and debugging as the application executes



Test J2EE Application



Test/debug
servlets as
they execute

The screenshot shows the IBM Debugger [Administrator] window. The title bar reads "Debugger [Administrator]". The menu bar includes "File", "Edit", "Workspace", "Selected", "Inspector", "Window", and "Help". The toolbar contains various icons for debugging, including a play button, a stop button, and a search icon. Below the toolbar are tabs for "Debug", "Breakpoints", and "Exceptions".

The "All Programs/Threads" pane shows a tree view of threads. The selected thread is "Thread[Server Thread,5,WAS_HTTP_TRANSPORT-]". Below it, the stack of frames is visible, with the current frame being "Ex01Servlet.performTask(HttpServletRequest, HttpS...".

The "Variable" pane shows the following variables and their values:

Variable	Value
this	
request	
response	
isRuntimeContextSet	
runtimeContext	

The "Source" pane shows the following code snippet:

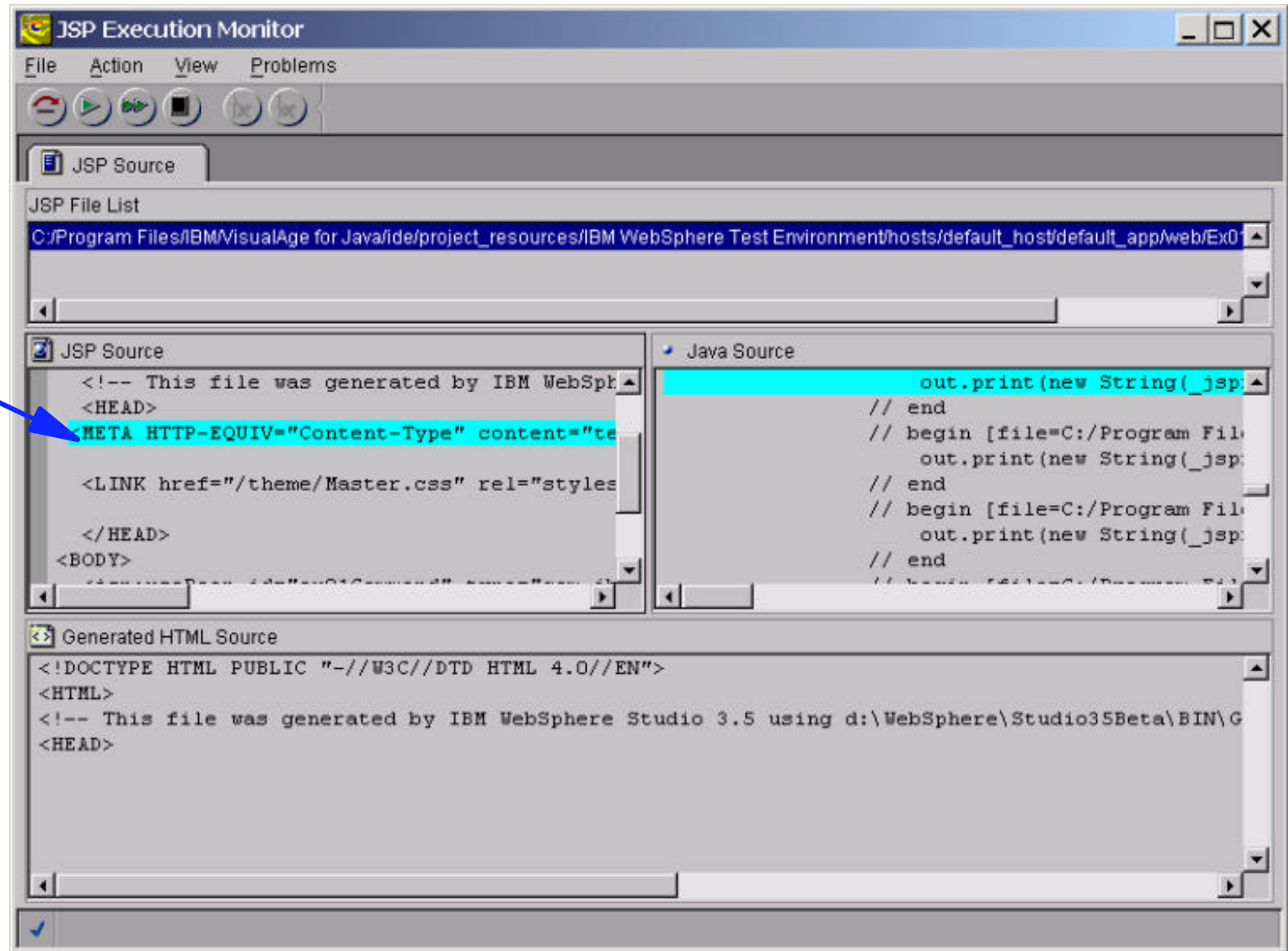
```
try  
{  
    // instantiate the beans and store them so they can be accessed by the ca  
    com.ibm.connector.ims.sample.phonebook.command.Ex01Command ex01Command =  
    setRequestAttribute("ex01Command", ex01Command, request);  
  
    // Setup your IMS Logon Information  
    com.ibm.connector.imstoc.IMSLogonInfoItems logon =  
        new com.ibm.connector.imstoc.IMSLogonInfoItems(  
            RuntimeContext.getComment(), getLogonInfo(), ex01Command.getComment()
```

The status bar at the bottom of the window displays the current file path: "com.ibm.connector.ims.sample.phonebook.servlet.Ex01Servlet.performTask", the date and time: "(1/11/2001 6:58:22 PM)", and the user name: "Administrator".

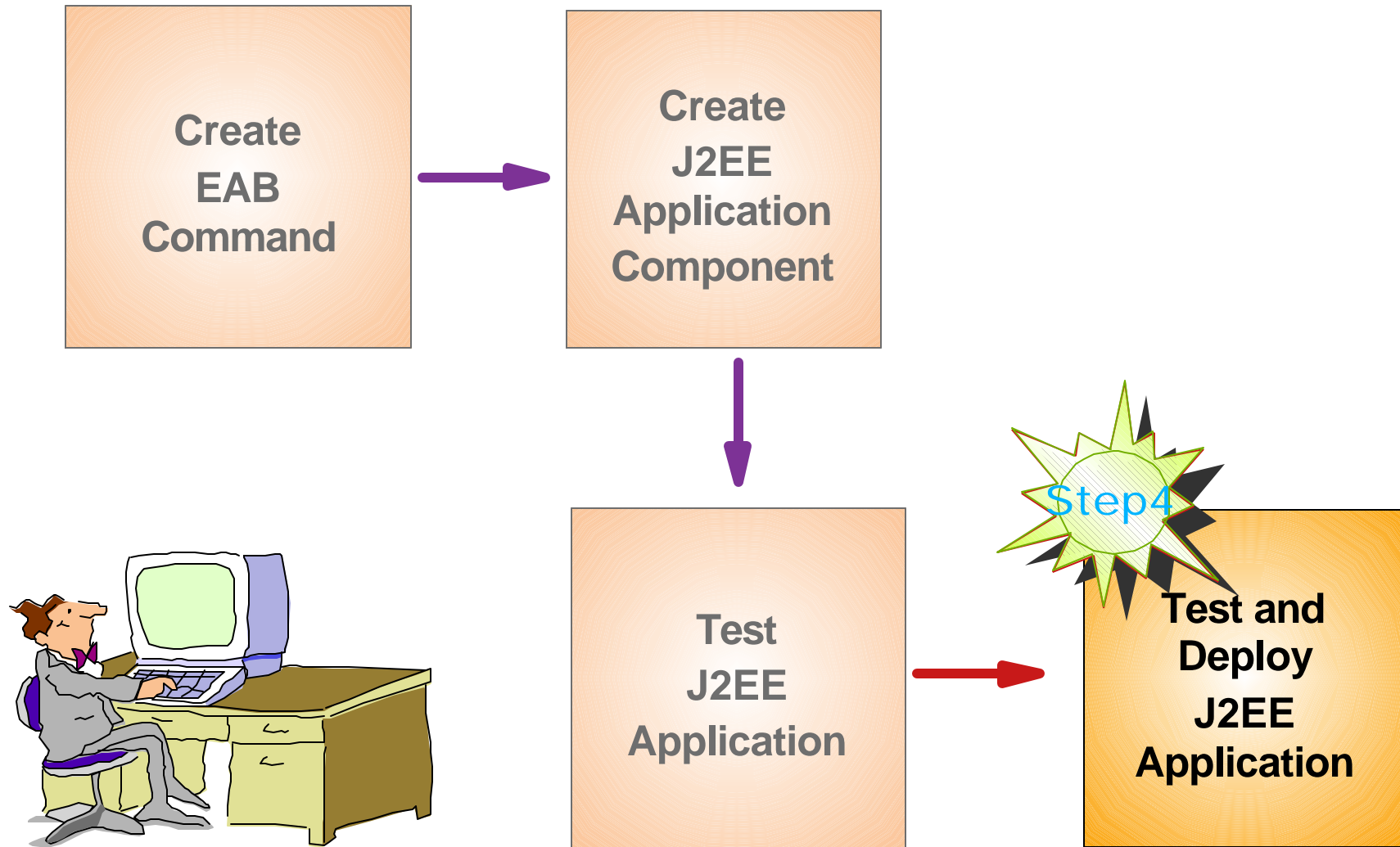
Test J2EE Application



Test/debug
JSP as they
execute



Build a J2EE Application



Step 4: Assemble and Deploy J2EE Application



- Assemble J2EE application components (Web and EJB modules) into a J2EE application using the Application Assembly Tool
- Deploy the J2EE Application to WebSphere Application Server
- Invoke input HTML page from Web browser to run IMS transaction, for example:
 - ▶ <http://<hostname>/myApp/yourInput.html>

Assemble J2EE Application



EJB
Module

Web
Module

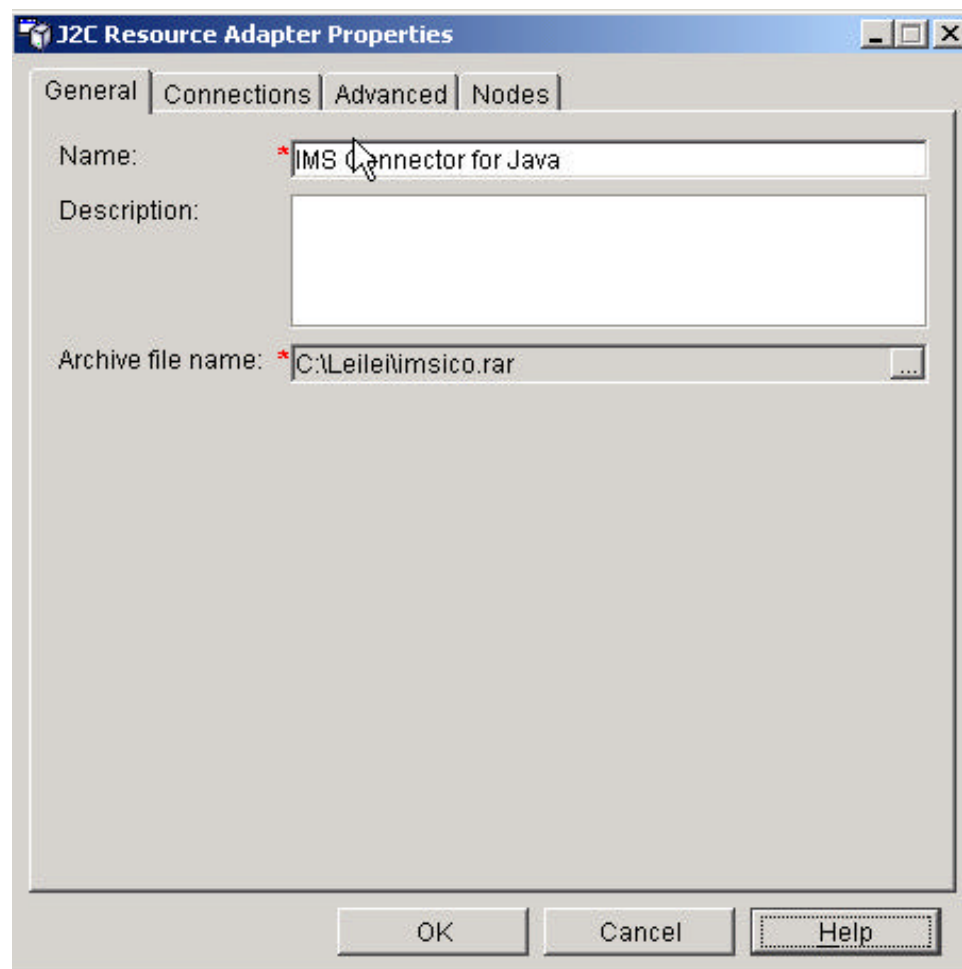
The screenshot shows the Application Assembly Tool interface. The main window is titled "Application Assembler - D:\haley\phonebooksample\IMSPhoneBook.ear". The left pane displays a tree view of the application structure:

- IMSPhoneBookApp
 - EJB Modules
 - phonebookDep.jar
 - Session Beans
 - Entity Beans
 - Security Roles
 - Method Permissions
 - Container Transactions
 - Files
 - Web Modules
 - IMSPhoneBookWAR
 - Web Components
 - Security Constraints
 - Context Parameters
 - EJB References
 - Environment Entries
 - Error Pages
 - MIME Mappings
 - Resource References
 - Security Roles
 - Servlet Mapping
 - Tag Libraries
 - Welcome Files
 - Assembly Property Exter
 - Files
 - Application Clients
 - Security Roles

The right pane shows the "General" tab of the configuration for the application. It contains the following information:

- Name: (empty)
- Application Clients
- EJB Modules
- Files
- Security Roles
- General | Icons | IBM Extensions | Bindings
- Description: An enterprise application consists of one or more Web applications, enterprise beans, or application clients.
- File name: D:\haley\phonebooksample\IMSPhoneBook.ear
- Display name: IMSPhoneBookApp
- Description: (empty text area)
- Buttons: Apply, Reset, Help

Deploy IMS Connector for Java

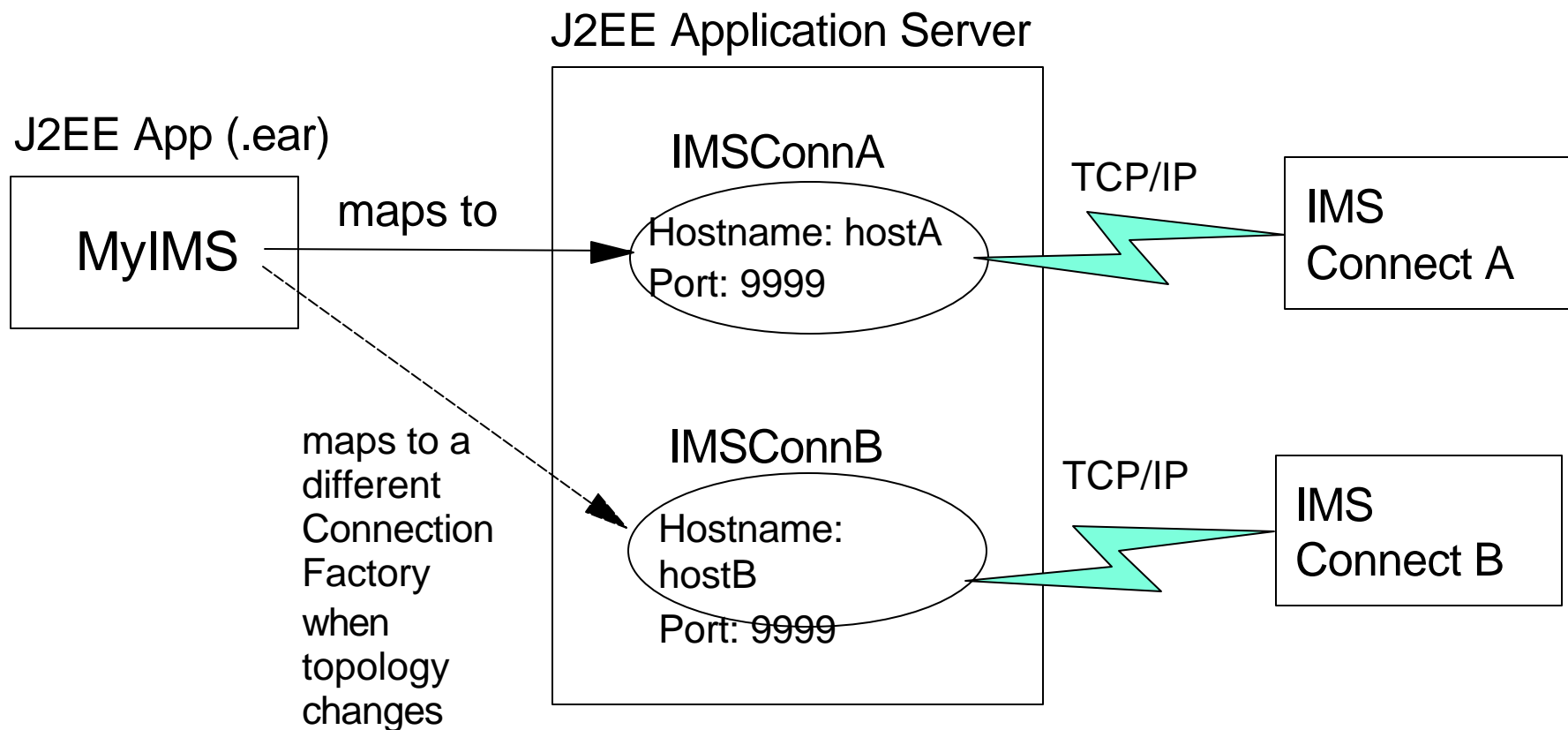


Connection Lookup



- Uses JNDI to lookup a ConnectionFactory and acquire Connection
 - ▶ Looks up a ConnectionFactory in JNDI
 - ▶ Uses the ConnectionFactory to acquire a Connection
 - ▶ e.g. MyIMS is a JNDI reference name that bounds(maps) to a connection factory.
- J2EE Application Server provides tool to define the properties of ConnectionFactory at deployment time

Connection Configuration



- ConnectionFactories are configured at deployment time using Application Server's administrative tool. You maps the application's resource reference (e.g. "MyIMS" in this case) to a particular connection factory
- Also, You can map "MyIMS" to a different ConnectionFactory resource when the system topology changes. No need to modify the J2EE Client Application code.

Configuring Connection Factory



J2C Connection Factory Properties

General | **Advanced** | Connections

Name: *MSConnA

JNDI binding path: eis/MyMSConnA

Description:

J2C resource adapter: *IMS J2C

OK Cancel Help

Configuring Connection Factory



J2C Connection Factory Properties

General | Advanced | Connections

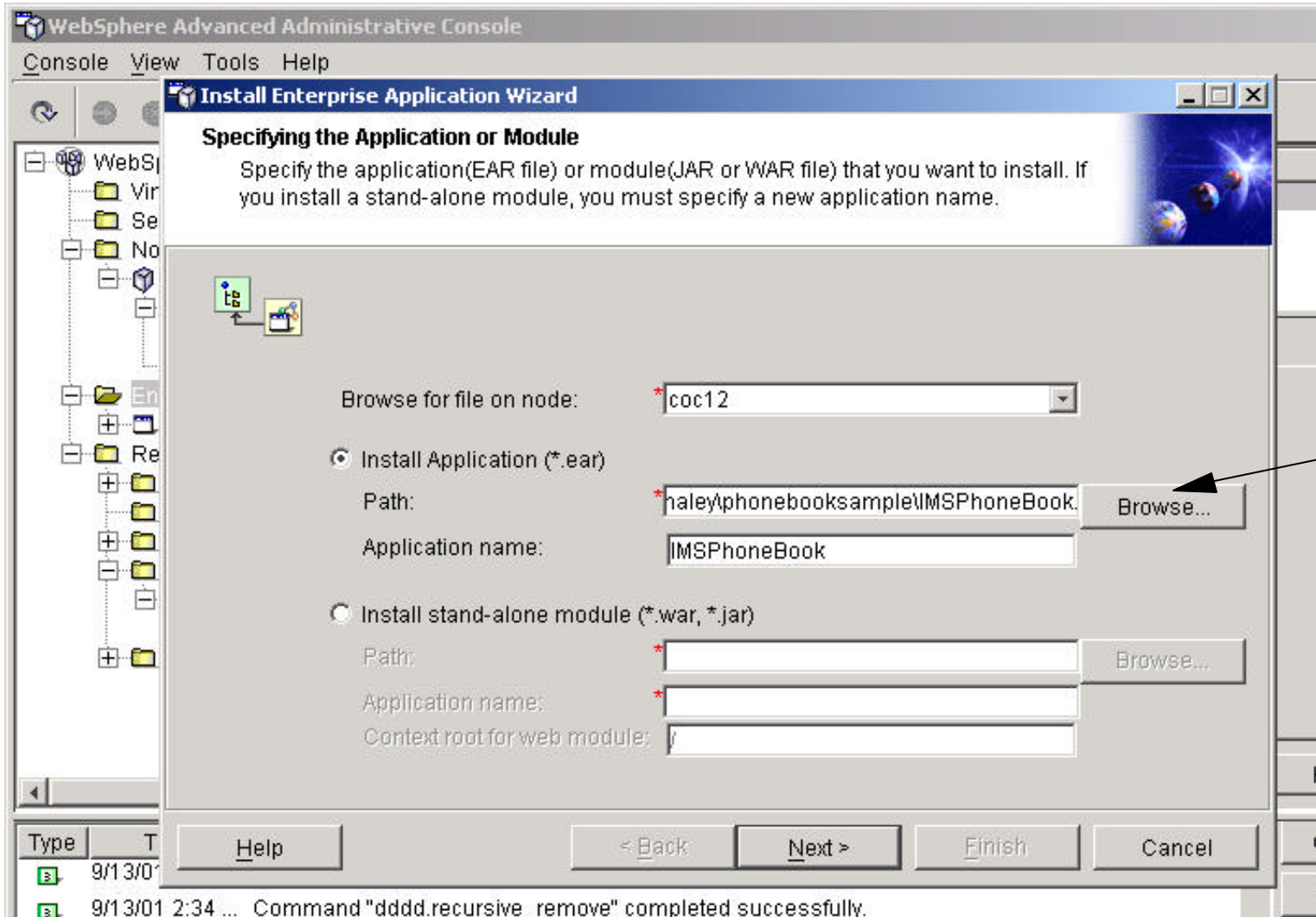
Name	Description	Data Type	Value
PortNumber	Target TCP/IP port numb...	java.lang.Int...	0
Password	Default password of the ...	java.lang.Str...	
HostName	TCP/IP host name of the ...	java.lang.Str...	myHostA
DataStoreN...	Name of the target IMS d...	java.lang.Str...	myDStrNm
TraceLevel	Level of information to be...	java.lang.Int...	1
UserName	Default name of the user	java.lang.Str	

Name: HostName
Description: TCP/IP host name of the target IMS Connect

Data Type: java.lang.String
Value: myHostA

OK Cancel Help

Deploy J2EE Application



J2EE
Application
(.ear)

Migration



- VisualAge for Java supports tooling for migration assistance from CCF to J2EE Connector Architecture
- VisualAge for Java's Enterprise Access Builder provides tooling to migrate existing EAB commands, Navigators and Java Record Beans from CCF to J2EE Connector Architecture
- The following properties of IMSInteractionSpec are not migrated:
 - ▶ Synchronization level (not supported in the initial release)
 - ▶ IMS Datastore name (now becomes a property of the connection factory)
- May need to modify the application code to use the new IMSConnectionSpec class in the EAB Command to provide the Security information

Migration



The screenshot shows the IBM Rational IDE interface. On the left, a project tree is visible with the package `com.ibm.connector.ims.sample.phonebook` expanded, and the class `Ex01Command` selected. A context menu is open over this class, showing the path: **Tools** > **Enterprise Access Builder** > **Migrate Commands...**. The `Migrate Commands...` option is circled in red, and a blue arrow points to it from the text "Migrate CCF EAB commands".

```
import com.ibm.ivj.eab.command.*;

// user code begin {__import_statements_1}
// user code end {__import_statements_1}

public class Ex01Command extends com.ibm.ivj.ea
```

Migrate CCF
EAB
commands

Migration



SmartGuide

Migrate to Connector Architecture

Select the Commands to migrate:

Project: Browse...

Package: Browse...

Class name:

<input type="text" value="com.ibm.connector.ims.sample.phonebook.command.Ex01 Command"/>	Add...
	Delete

< Back Next > Finish Cancel

Hints and Tips



- Generate efficient Java Record Beans
- Use of CCF RuntimeContext and Connection Pooling on different platforms
- MAXSOC and MaxConnections

IMS Connector Documentation



- Visual Age for Java Online Help
 - ▶ User's Guide
 - Online Version
 - Help->Task->Accessing the Enterprise->Accessing transactions with the IMS Connector
 - PDF Version
 - Help->PDF Index->PDF Documents->IMS Connector for Java
 - ▶ Diagnosis Guide
 - Help->Task->Accessing the Enterprise->Building IMS Applications
 - ▶ Javadoc for IMS Connector for Java classes
 - Help->Reference->IBM APIs->Connectors->IMS Connector
- IMS Connector for Java website
 - ▶ <http://www.ibm.com/ims>
 - Hints and Tips, Platform Guide and Additional Samples

Related Web Sites



*e-business powered by
IMS*

- **IMS, IMS Connect, IMS Connector for Java**
 - <http://www.ibm.com/ims>
- **VisualAge for Java**
 - <http://www.ibm.com/software/vajava>
- **WebSphere Application Server**
 - <http://www.ibm.com/software/webservers>