

From TP to Web Services:

Using BEA WebLogic Server and HP BridgeWorks to Expose Business Transactions as Web Services

John Apps
Senior Technical Architect
Business Critical Systems
Session 2063



Agenda

■ From

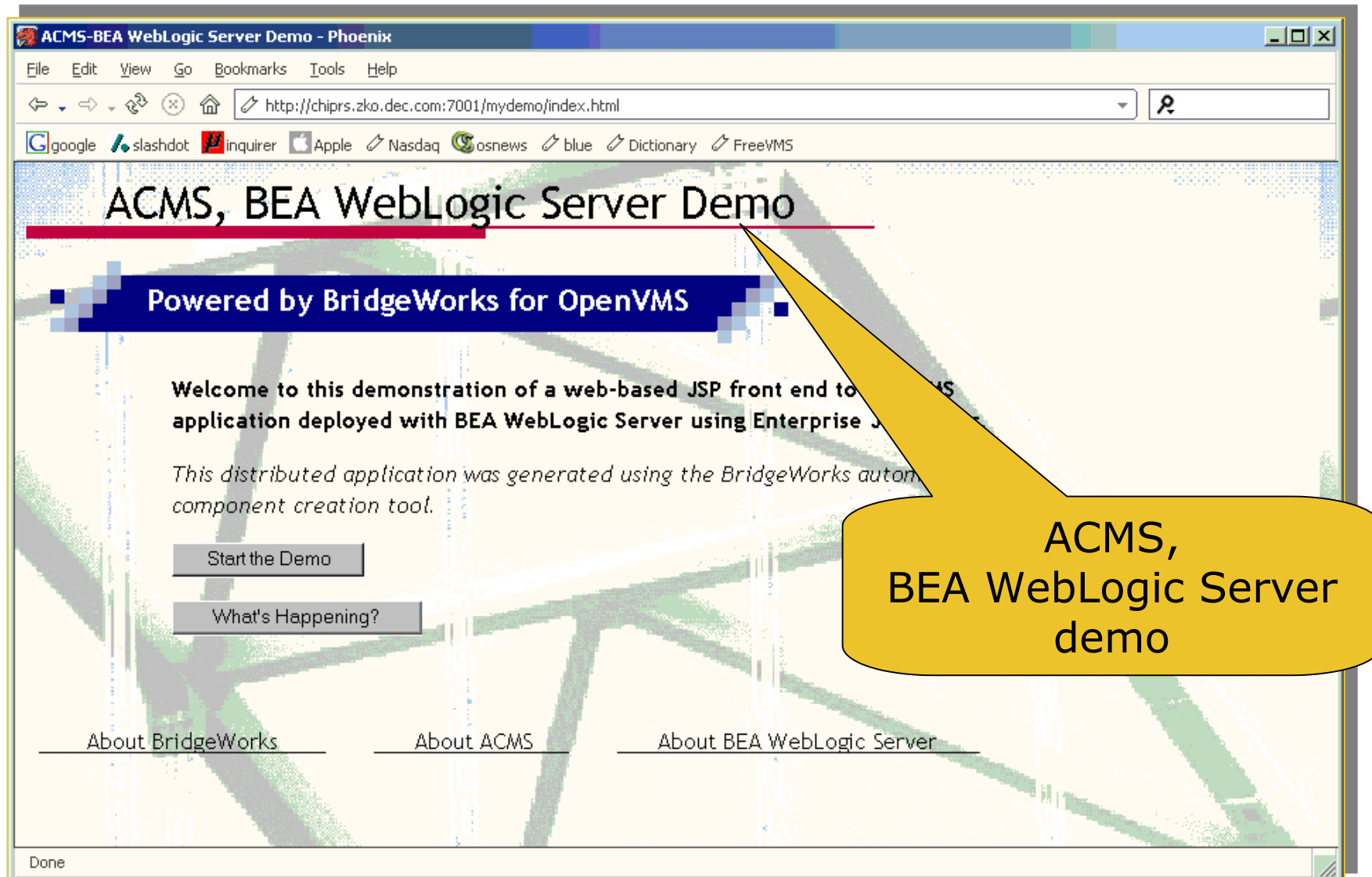
- Transaction Processing
 - ACMS: Application Control and Management System
 - hp BridgeWorks
 - BEA WebLogic Platform
 - BEA WebLogic Workshop
 - BEA WebLogic Server

■ To

- Distributed Computing and
- Web Services

■ Summary

Transaction Processing



Transaction Processing

ACMS, BEA WebLogic Server Demo - Phoenix

File Edit View Go Bookmarks Tools Help

http://chiprs.zko.dec.com:7001/mydemo/add.jsp

google slashdot inquirer Apple Nasdaq osnews blue Dictionary FreeVMS

ACMS, BEA WebLogic Server Demo

Add Employee

Main Menu

Employee ID:

Employee Name:

Address:

City:

Country:

Postal Code:

Submit Reset

Powered by BridgeWorks

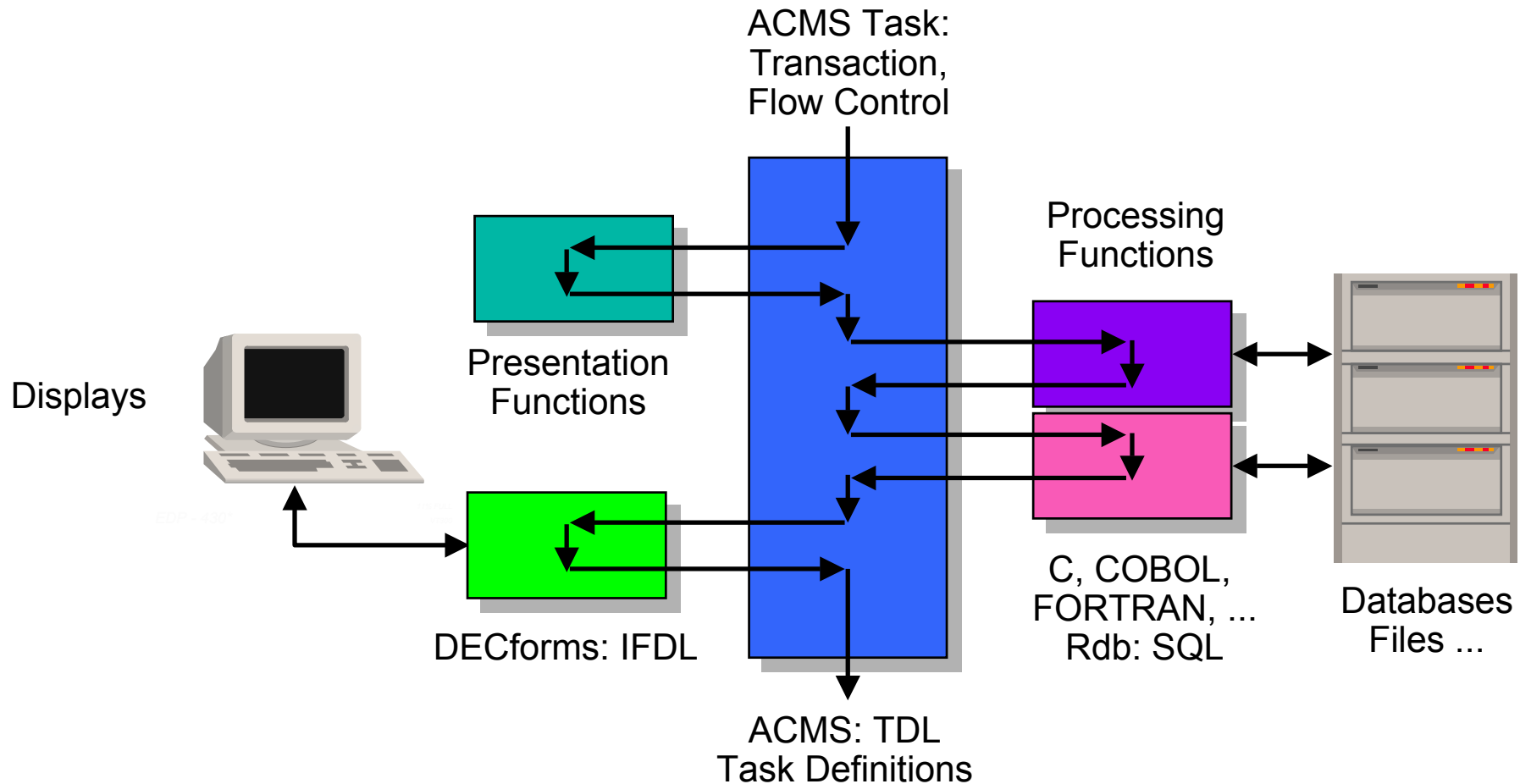
Done

ACMS, BEA WebLogic Server demo

Application Control and Management System

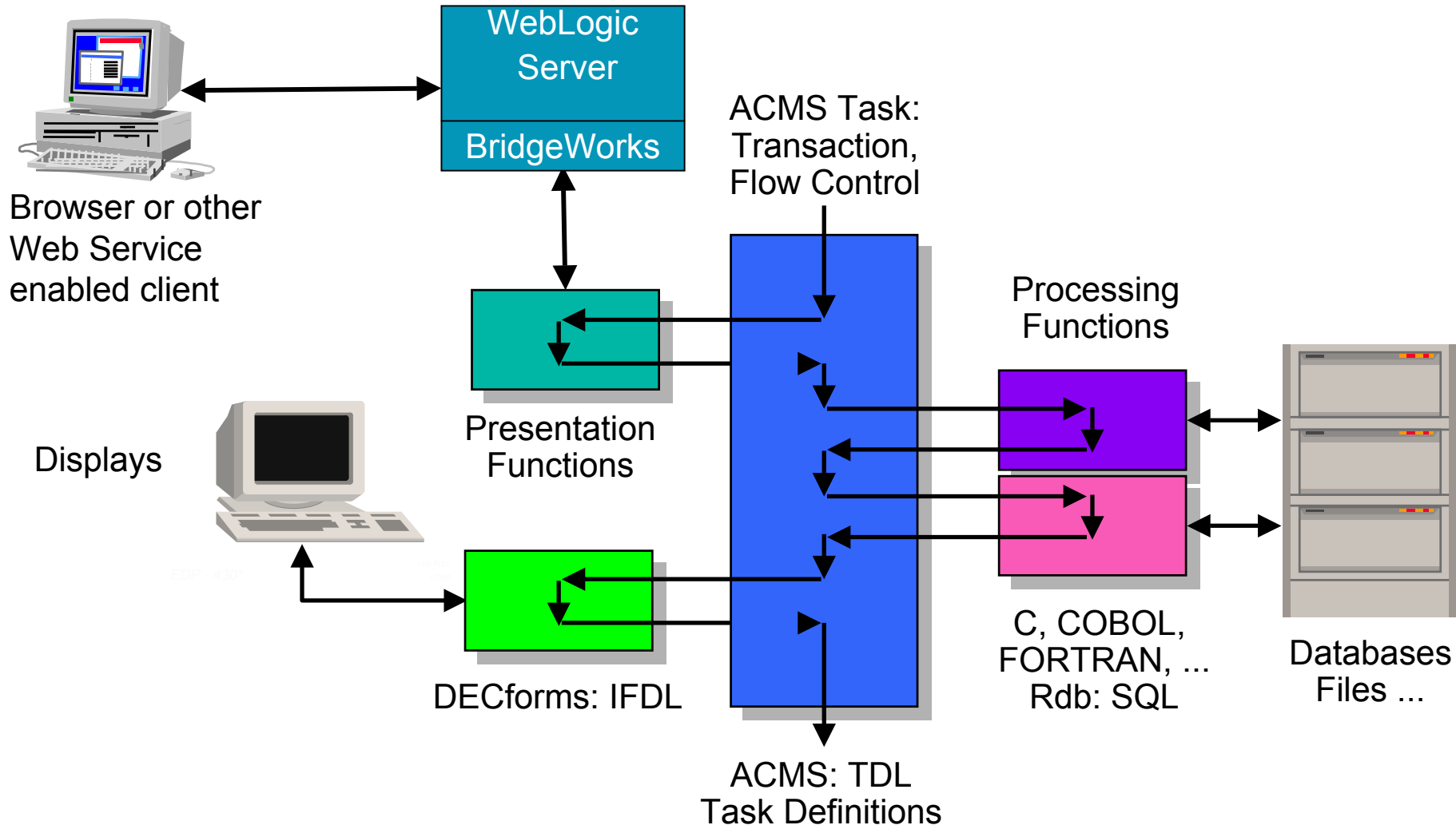
- Designed and written 1981-1984
 - V1.0 – 1984 – First release, after 2 field tests
 - V1.2 – 1985 – 1st client/server release
 - V3.2 – 1991 – Distributed Txn (DECdtm) support
 - V4.0 – 1994 – Port to OpenVMS Alpha
 - V4.3 – 1998 – Remote Manager support
 - V4.4B – 2002 – Current Release
-
- Some 15,000 systems in [mission critical] production world-wide

ACMS Application Model



Modular Applications With a TP-Specific Programming Language

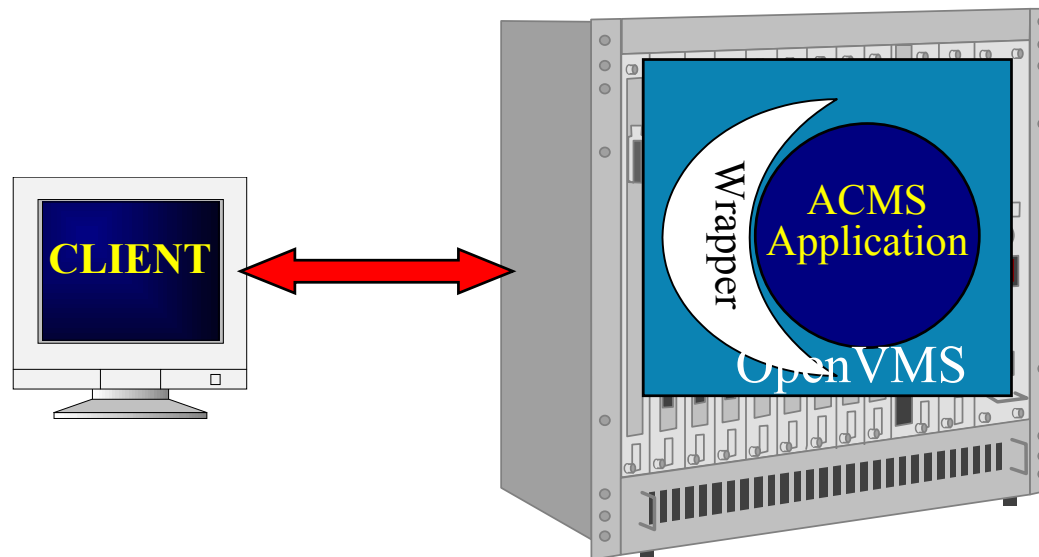
ACMS, hp BridgeWorks and BEA WebLogic Server



Modular Applications With a TP-Specific Programming Language

Hp BridgeWorks

- Provides
 - Wizard-style dialogs to step users through code generation
 - Easy Point & Click building of Components/Interfaces
 - Easy modification of previously generated 'connections'



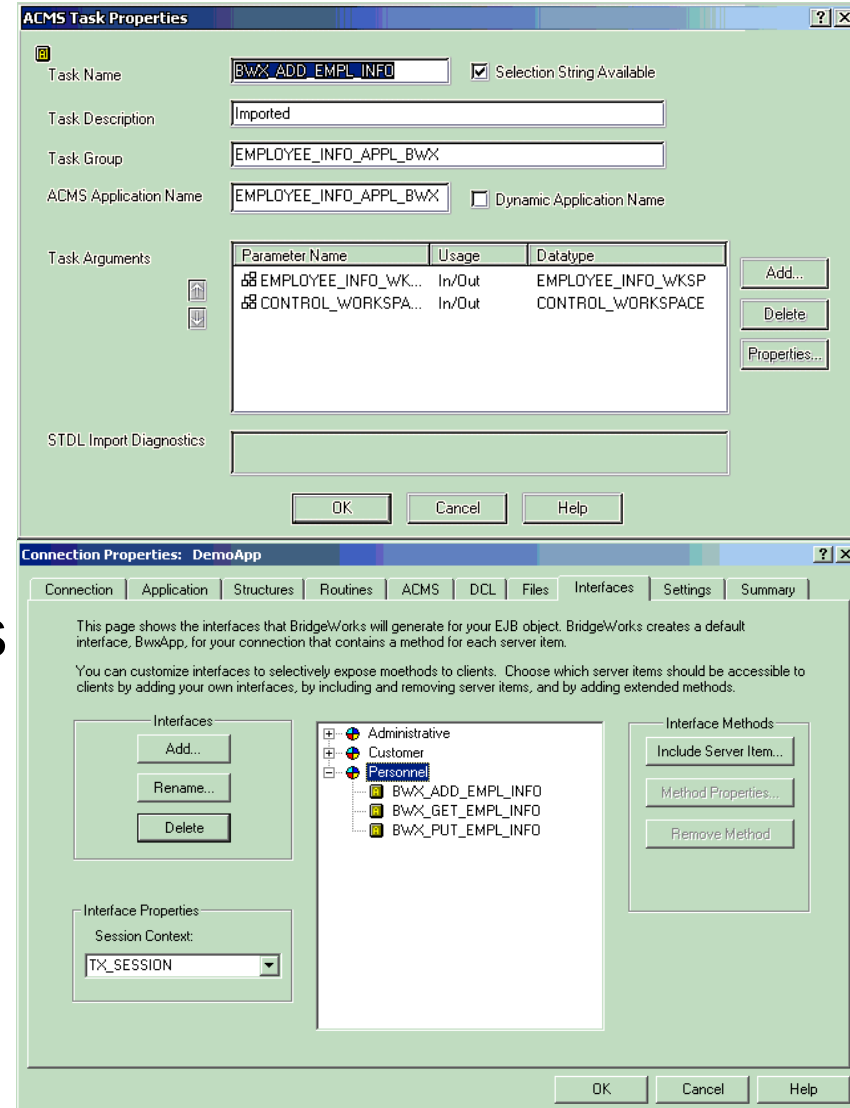
hp BridgeWorks

Generates robust EJB Interfaces from your TP application

- Creates J2EE Stateful SessionBeans
- Creates 1 EJB per interface

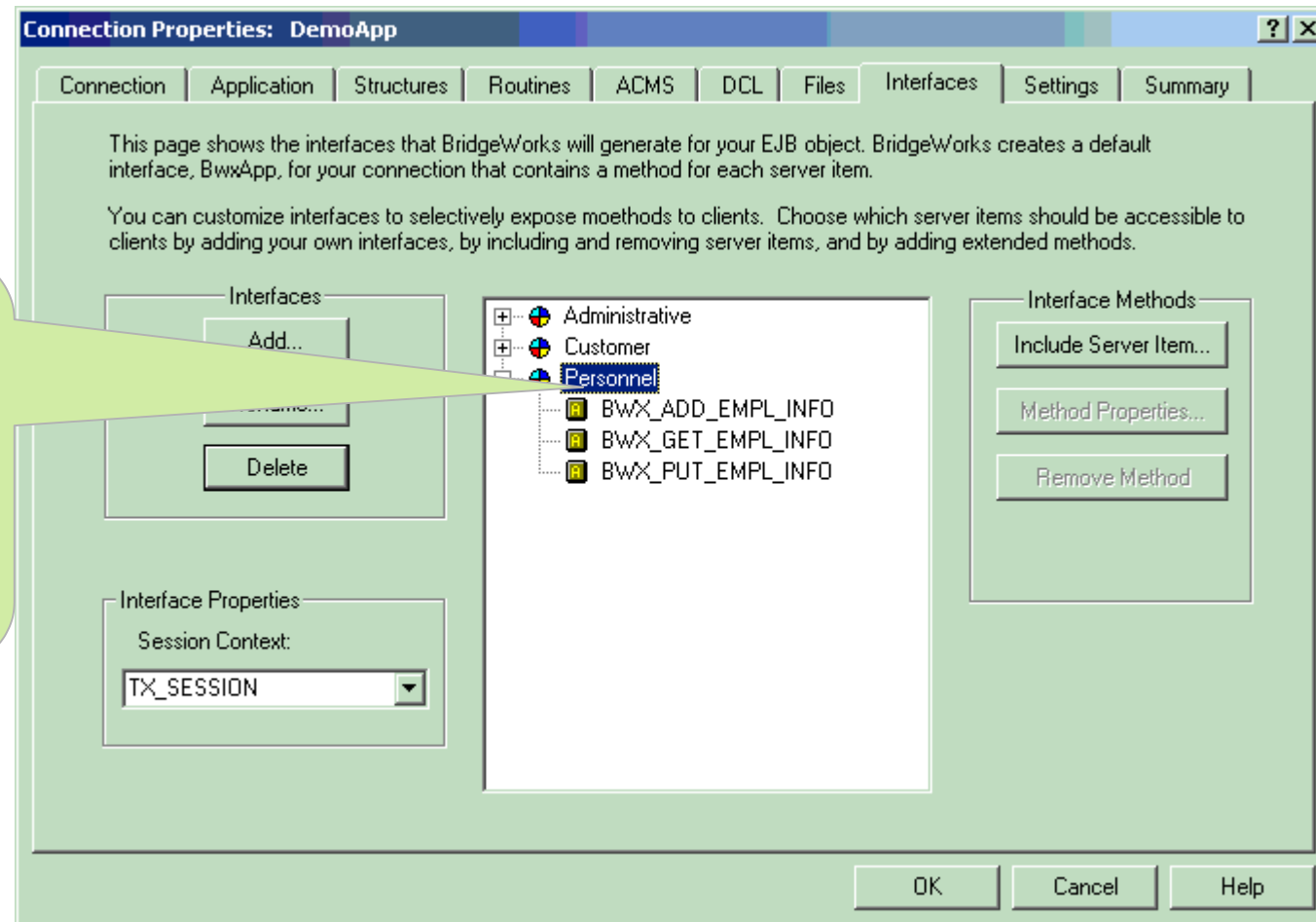
Generates JavaBean Interfaces from your TP application

- Creates a single simple class per interface
- J2EE environment not needed to build or run



hp BridgeWorks

ACMS Task
(Txn.) imported
by BridgeWorks
from ACMS
meta-data
automatically



hp BridgeWorks

ACMS Task
(Txn.) 'attributes'
displayed by
Bridgeworks

ACMS Task Properties

Task Name: ☒ Selection String Available

Task Description:

Task Group:

ACMS Application Name: ☐ Dynamic Application Name

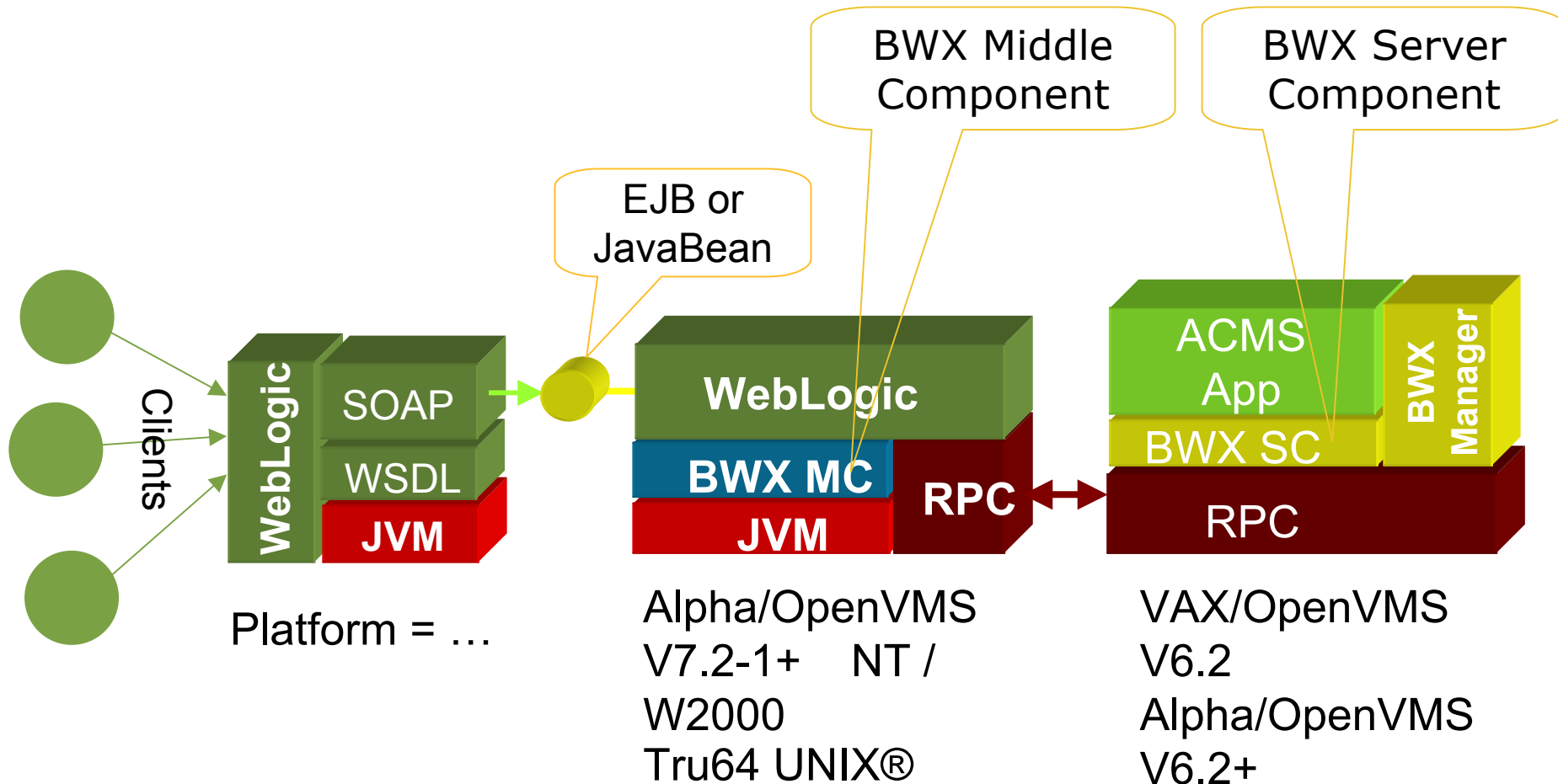
Task Arguments:

Parameter Name	Usage	Datatype
EMPLOYEE_INFO_WK...	In/Out	EMPLOYEE_INFO_WKSP
CONTROL_WORKSPA...	In/Out	CONTROL_WORKSPACE

STDL Import Diagnostics:

Buttons: Add..., Delete, Properties..., OK, Cancel, Help

hp BridgeWorks (BWX) Components



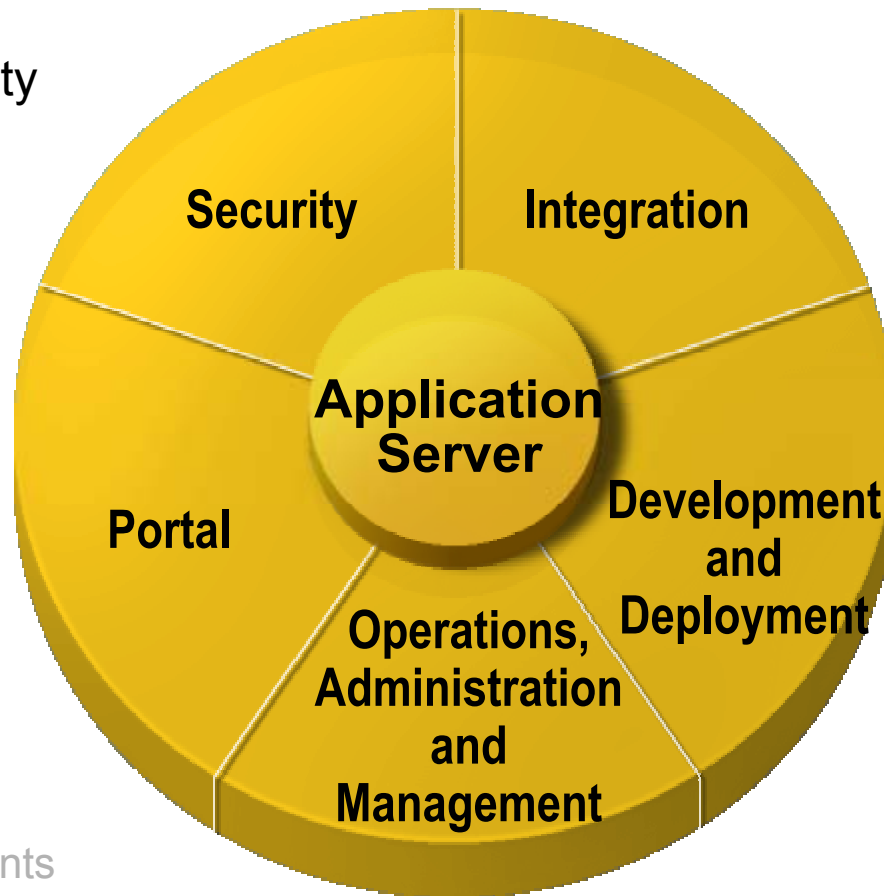
BEA WebLogic Platform

BEA WebLogic Server

- Scalability, Reliability
- Component model, J2EE
- Transactions
- Admin & Security
- Web Services

BEA WebLogic Portal

- Personalization
- E-Commerce
- E-Marketing
- Usability Enhancements
- Web Services



BEA WebLogic Integration

- Business Process Management
- J2EE CA Adapters
- Scalability & Reliability

BEA WebLogic Workshop

- Quickly build Enterprise Applications
- Make J2EE and Web Services accessible to all developers, not just experts

BEA WebLogic Workshop

ACMS Tasks
'imported' from
EJB generated
by hp
Bridgeworks

The screenshot displays the BEA WebLogic Workshop interface for the 'Employee.jws' project. The Project Tree on the left shows the project structure: Project 'eWorld' containing WEB-INF, Employee.jws*, EWCTRLControl.ctrl, and index.html. The Design View shows a diagram with a CLIENT component and an eworld EJB control. The eworld control is highlighted with a red box, and a green callout bubble points to it from the text 'ACMS Tasks 'imported' from EJB generated by hp Bridgeworks'. The Properties - eworld panel on the right shows the Name 'eworld' and the home-jndi-name 'eworld_BwxApp'. The Structure Pane at the bottom left lists the methods of the eworld control: AcmsSignIn(), AcmsSignOut(), Bwx_ADD_EMPL_INFO(), Bwx_GET_EMPL_INFO(), Bwx_PUT_EMPL_INFO(), create(), and getBwxStatus(). The Status Bar at the bottom indicates 'Ready', 'Server Running', 'Ln 1', 'Col 1', and 'INS'.

BEA WebLogic Workshop

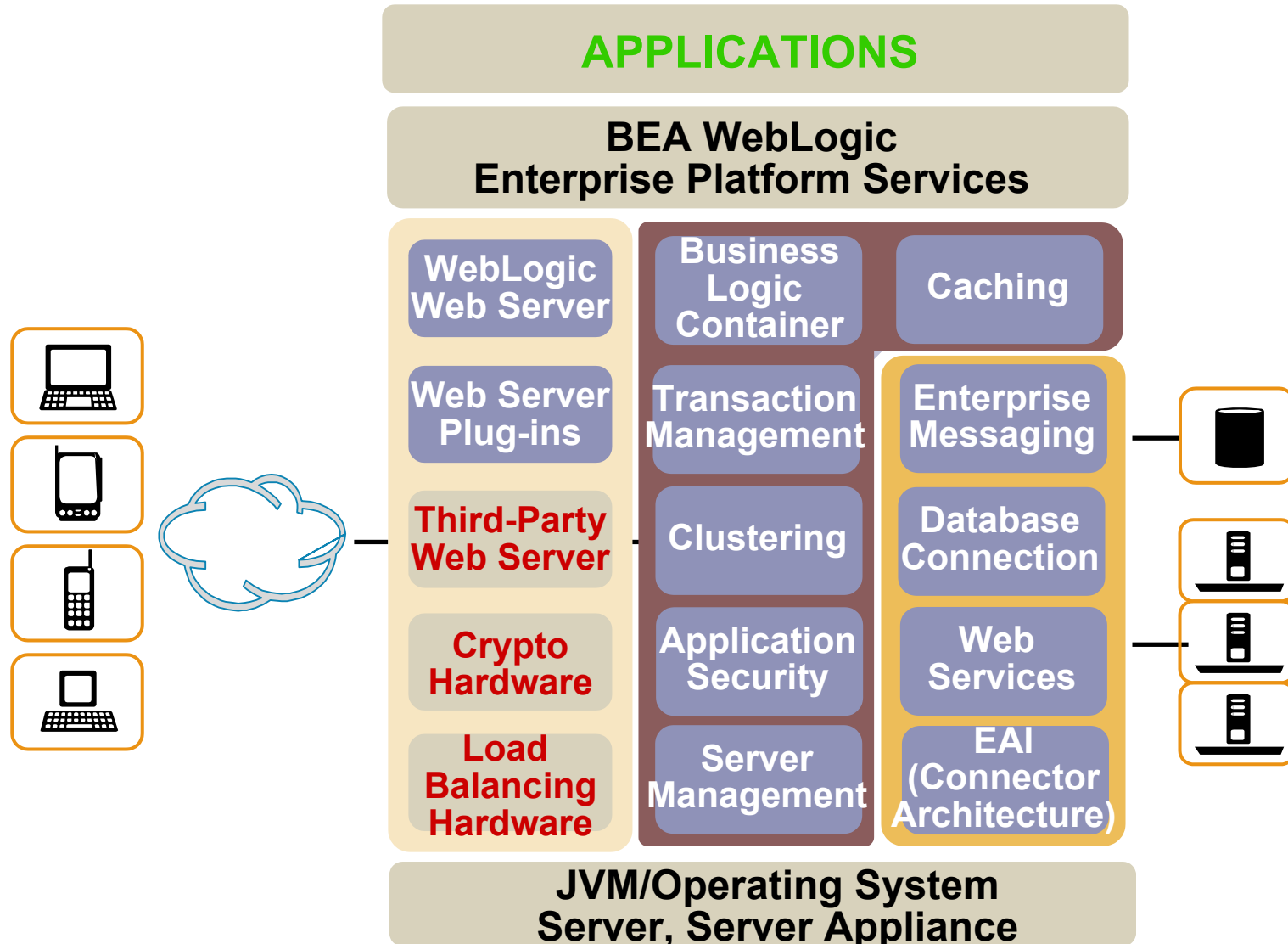
ACMS Task
'methods' for
invocation by
WebLogic
Server

The screenshot displays the BEA WebLogic Workshop interface for the **Employee.jws** service. The **Project Tree** on the left shows the project structure: **Project 'eWorld'** containing **WEB-INF**, **Employee.jws**, **EWCTRLControl.ctrl**, and **index.html**. The **Design View** in the center shows the service design with a **CLIENT** and an **eworld** control. The **CLIENT** has three methods: **ADD**, **GET**, and **PUT**. The **eworld** control has several methods: **AcmsSignIn**, **AcmsSignOut**, **BWX_ADD_EMPL_INFO**, **BWX_GET_EMPL_INFO**, **BWX_PUT_EMPL_INFO**, **create**, and **getBwxStatus**. The **Properties - Employee** pane on the right shows the service properties, including **target-namespace**, **conversation-lifetime**, **protocol**, **wsdl**, and **schema**. The **Description** pane at the bottom provides a description of the service and tasks.

Properties - Employee

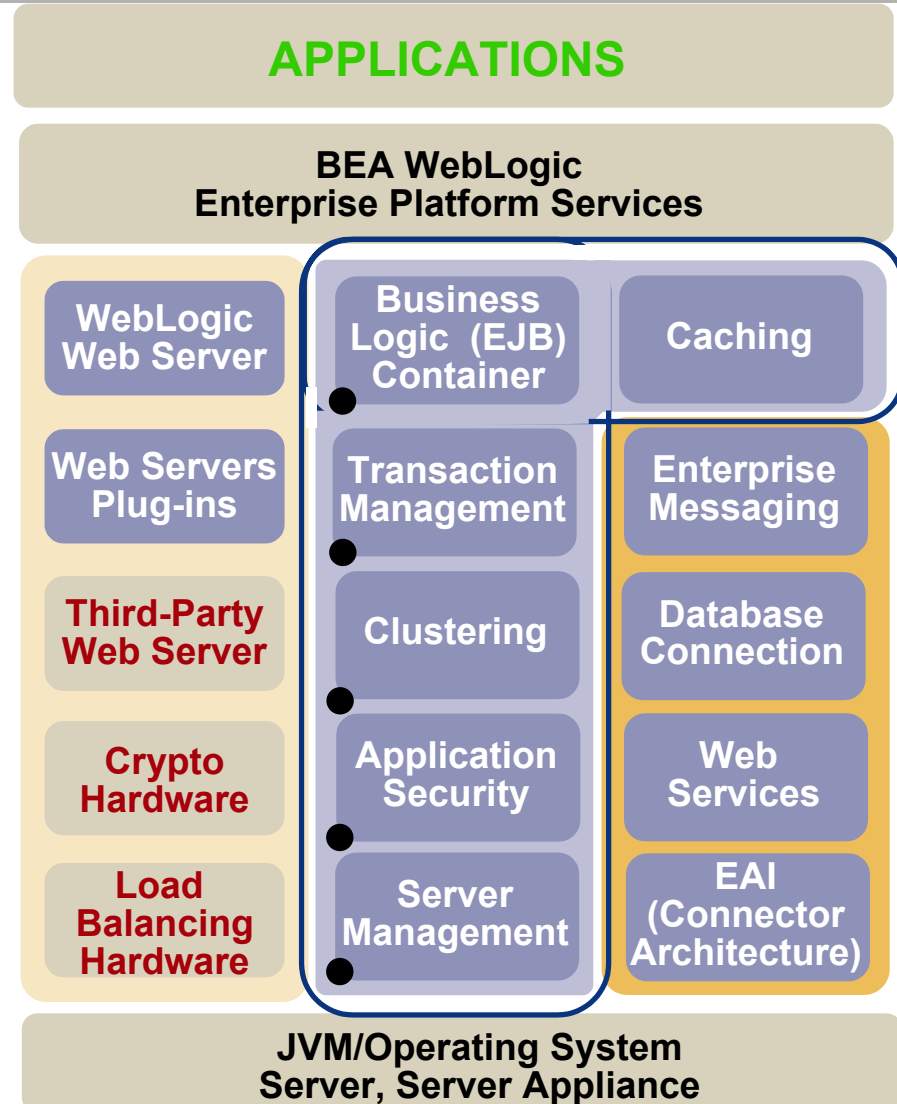
Name Employee	
☐ target-namespace	
namespace	
☐ conversation-lifetime	
max-age	1 day
max-idle-time	0
☐ protocol	
form-post	true
form-get	true
http-soap	true
http-xml	false
jms-soap	false
jms-xml	false
soap-style	document
☐ wsdl	
file	
☐ schema	
file	
Description	
Employee Service	
This is the web service that you are creating. You can add methods to the service to exchange messages with a client. You can also add controls to invoke other web services, connect to a database, use a timer, utilize an Enterprise JavaBean, or exchange messages with a JMS queue or topic.	
Tasks	
◆ Add a new method ◆ Add a new callback	

BEA WebLogic Server Architecture



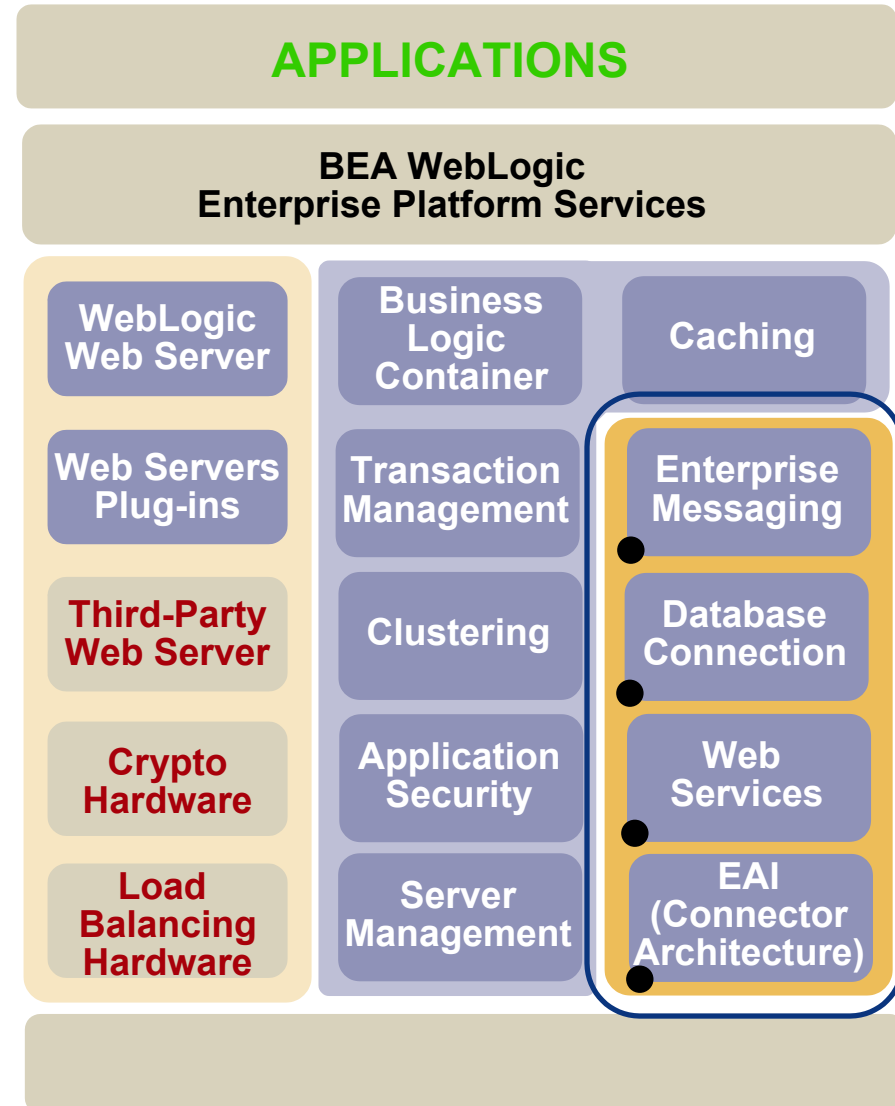
BEA WebLogic Middle tier

- J2EE Services
 - Session, Entity, Message Driven Beans (EJB 2.0)
 - Naming and Directory (JNDI)
 - JMS
 - Security
 - Transactional Support (JTA, two phase commit)
- Caching
 - EJB
- XML (with JAXP)



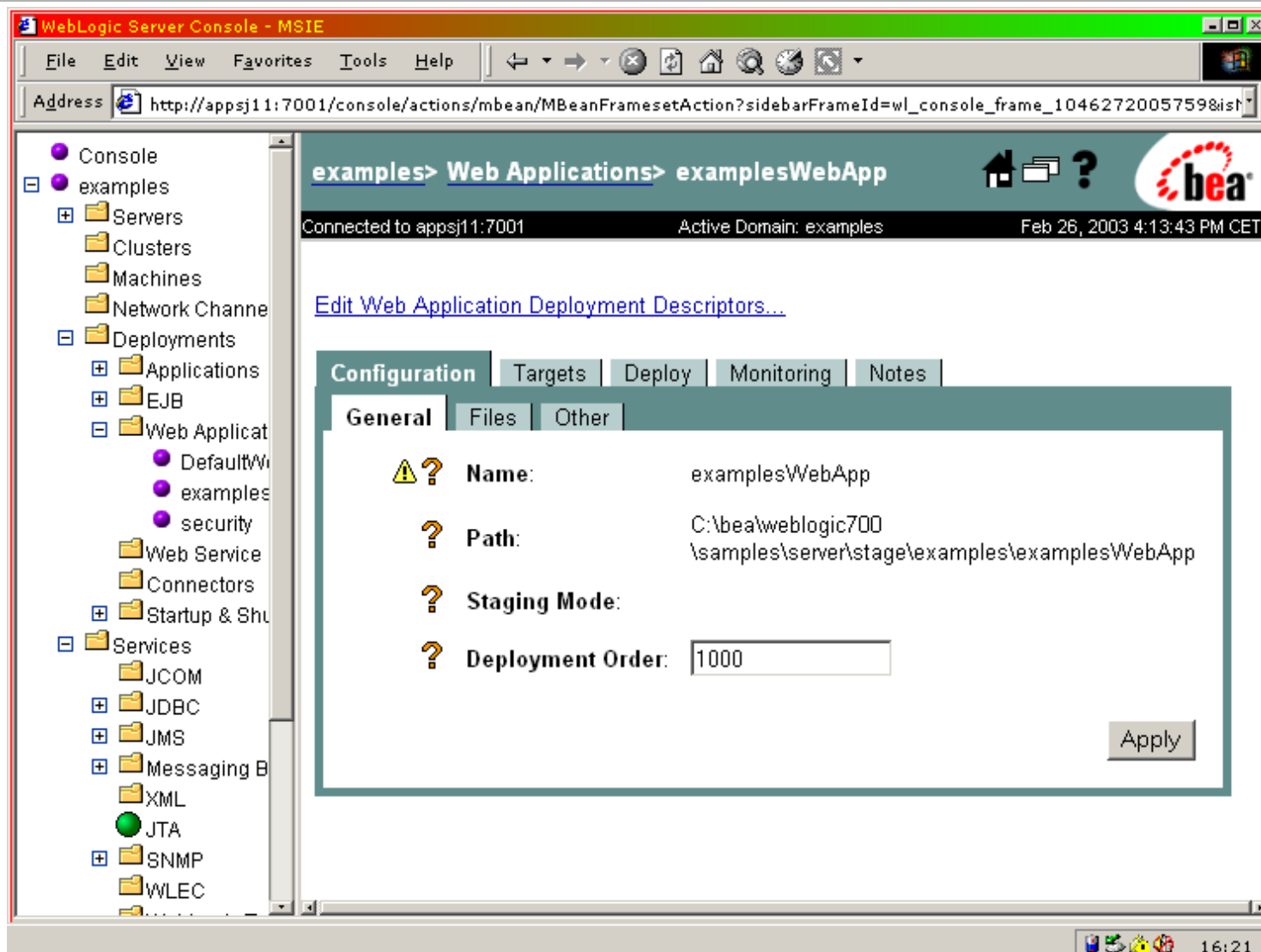
BEA WebLogic Integration Tier

- Web Services Support
 - XML, SOAP, WSDL, UDDI
- J2EE Connector Architecture
- Messaging
 - Built-in JMS
 - Messaging Bridge
- DBMS Access via JDBC, jDrivers
- RMI/IIOP
- WebLogic Tuxedo Connector
- jCom



BEA WebLogic Server

Used to
invoke and
control the
EJB or
JavaBean
methods
generated by
Bridgeworks
from ACMS
meta-data



Distributed Computing: Brief History

Client-Server Paradigms

- Synch – RPC / DCE
- Asynch - Message Queuing
- Database Access
- Distributed Transactions

Distributed Objects

- DCOM
- CORBA - IIOP
- Java - RMI

The Web – Middleware for the masses

- Static
- Dynamic – client/server-side scripts
- Context retention – Cookies, ASPs, JSPs...

Enriched Distributed Objects

- COM+
- CORBA Services
- EJB

Application Servers – Robust Web Applications

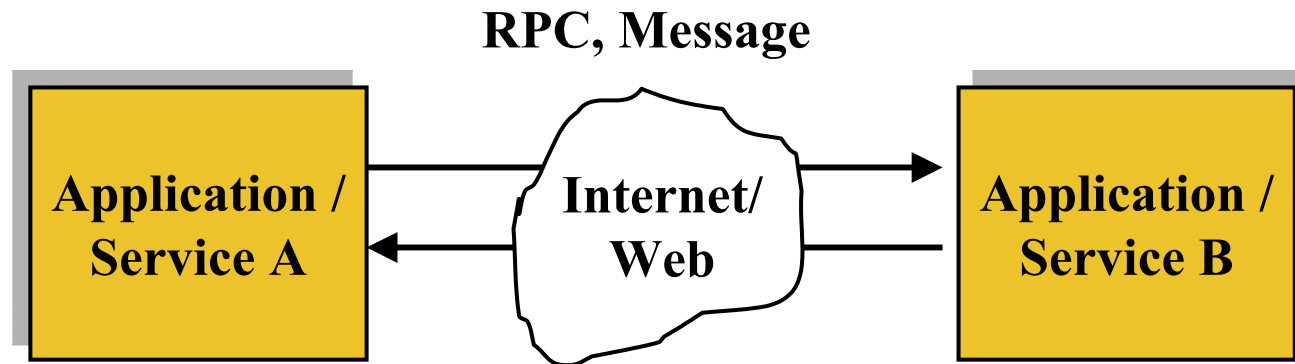
- MS, IBM, BEA, Oracle,...

Web Services

Web Services

- Provide the 'wire protocol' to:
 - Enable the invocation of remote methods
 - Handle marshalling of arguments
 - Allow message or RPC based connections
- Enable the discovery of services through standards-based directories
- Hide the intricacies of the methods being invoked
- Allow invocation of methods regardless of language and platform
- Endorsed by both Java and .NET
- Are prerequisite for Service-oriented Architectures

Web Services – New Paradigm

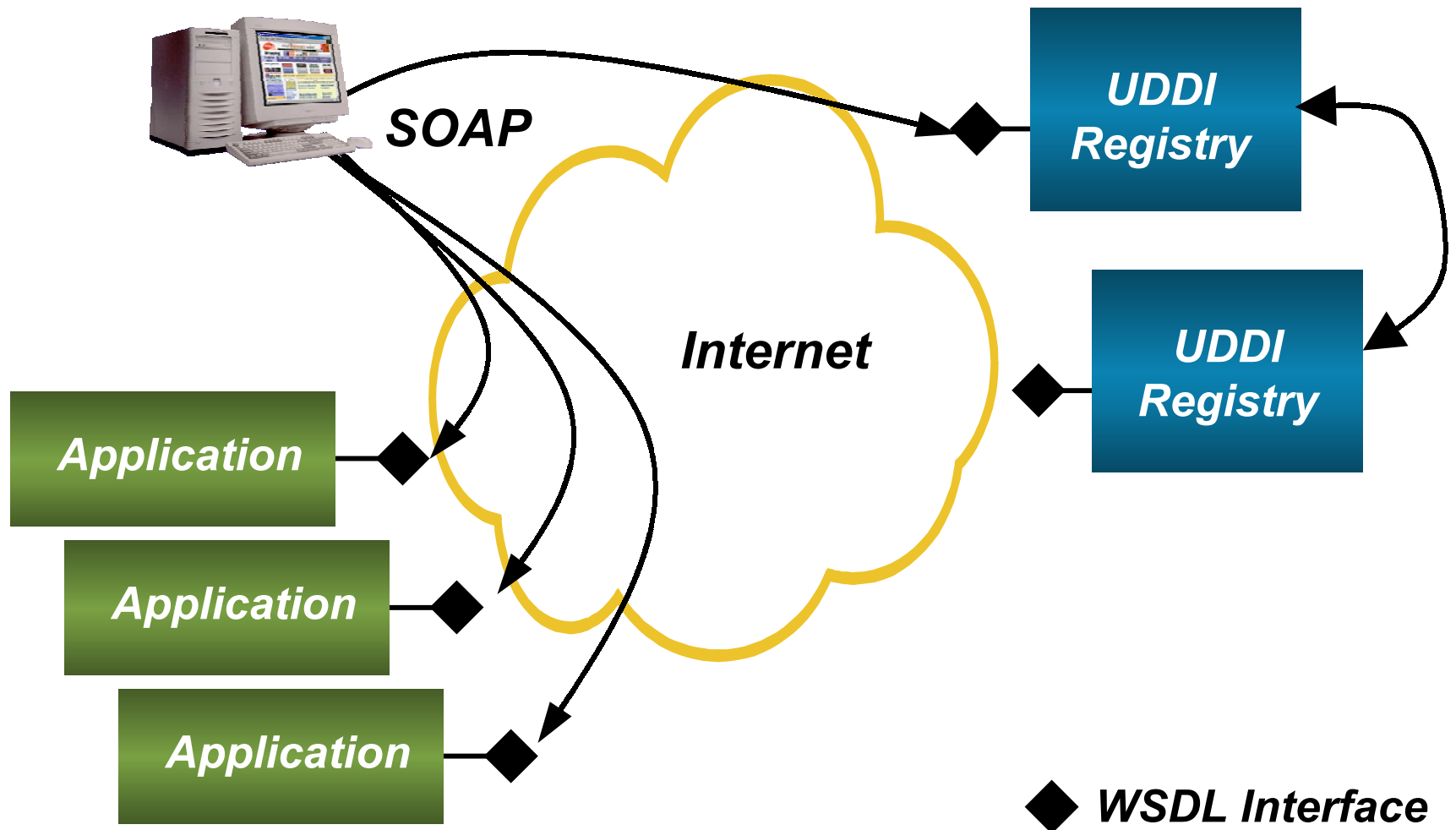


- Build (internet-wide) distributed applications via Web Services:
 - accessible using standard Internet protocols
 - Independent of operating system
 - Independent of programming language
 - represent black-box functionality that can be reused without worrying about how the service is implemented

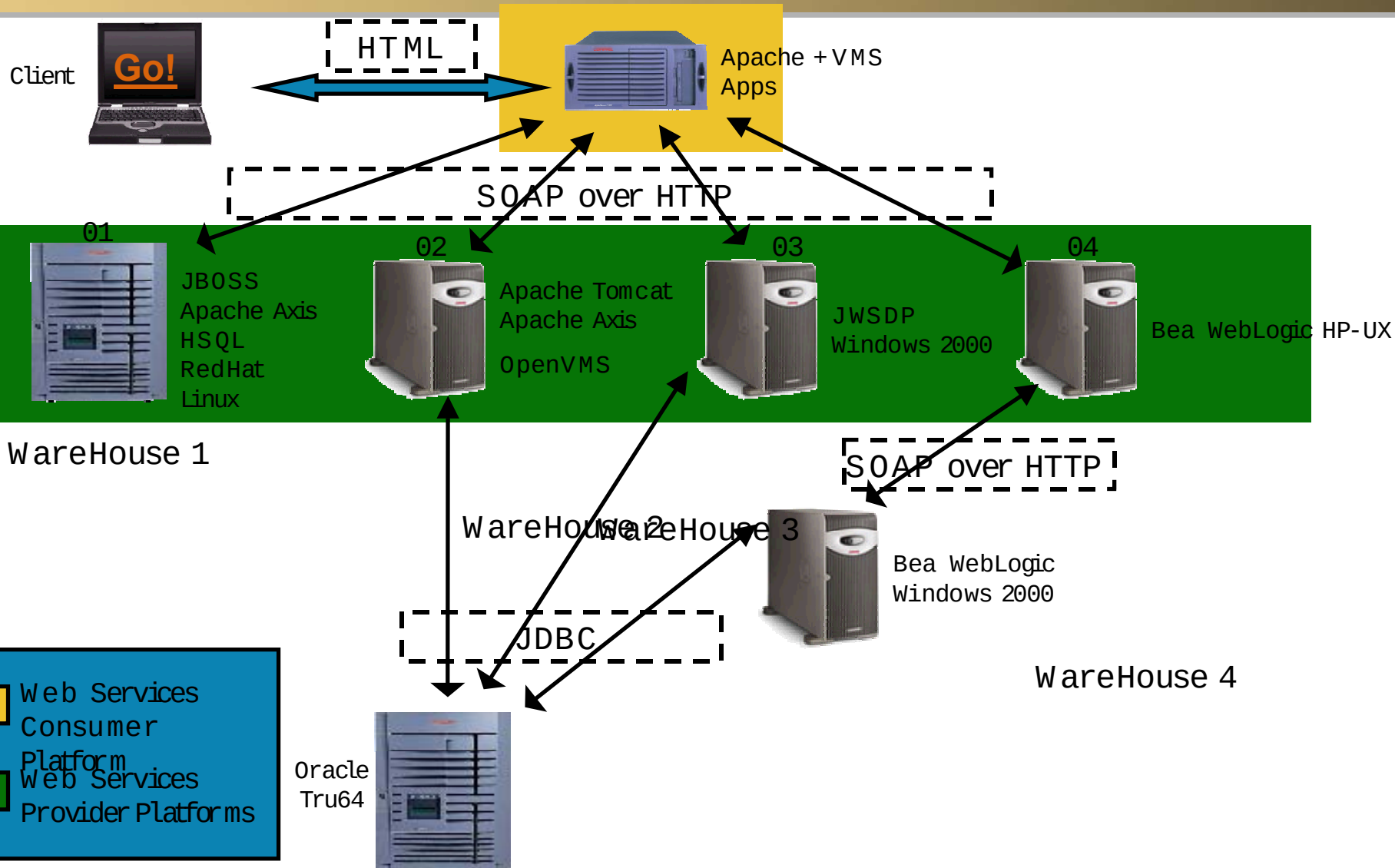
Web Services Technologies

- Simple Object Access Protocol (SOAP)
 - An XML-based way to invoke Web services
- Web Services Description Language (WSDL)
 - An XML-based language for describing Web services
- Universal Description, Discovery, and Integration (UDDI)
 - A registry for Web services
- Global XML Web Services Architecture (GXA)
 - SOAP headers for security and more

Web Services Technologies



Web Services - Typical Architecture



Summary

- Quickly generate all necessary components with little to no manual intervention or changes
- Generate EJB or JavaBean components from same definitions with no changes
- Integrate 'old' transactions using Web Services
- Reuse well-tested, reliable and long-running transaction processing functions
- Increase return on IT investment

Come and see it all in the hp booth!



i n v e n t