IBM Software Group

WebSphere. software

IBM

University of Toronto

J2EE Runtime for Business Applications

Dale A. Sue Ping
IBM Canada Ltd.

ON DEMAND BUSINESS

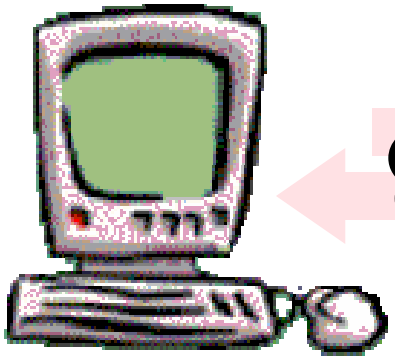
Agenda

- Why J2EE ?
- What is J2EE ?
- Specifications
- Architecture
- Tiers, Containers, Components
- Web Services
- New SOA technologies
- Applications Servers
  - Commercial, Open Source
- MiddlewareTrends
- Application Server Platform

Life was simple in the beginning…

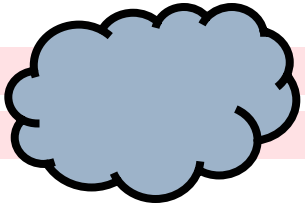
Then came the Internet …

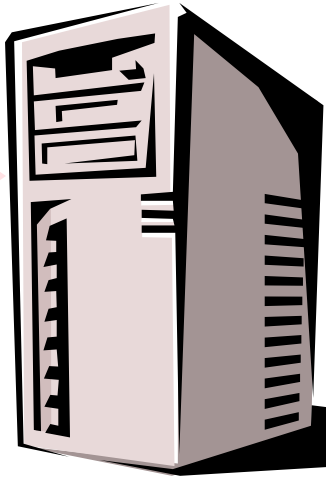
Web User



Web Browser

Internet Connection



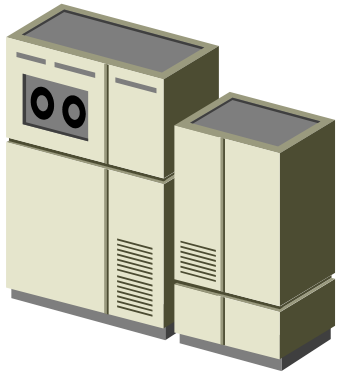


Web Server

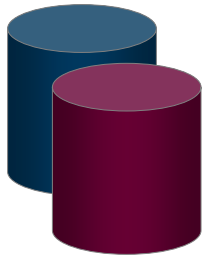
… then things got out of control!

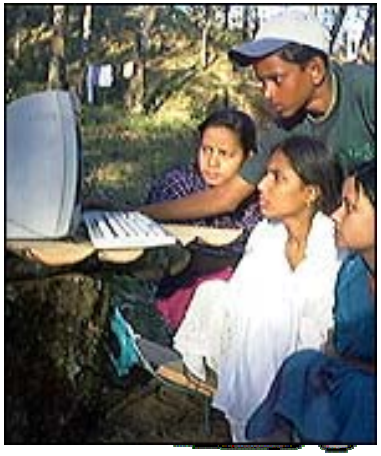
New devices,

Client

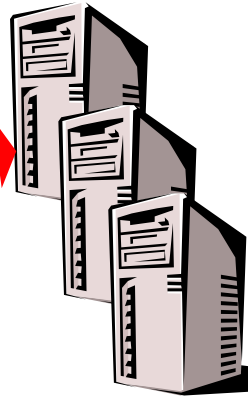


Legacy Systems

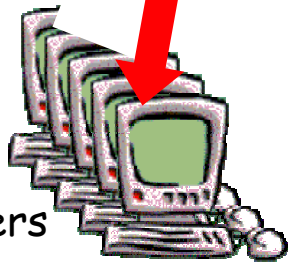




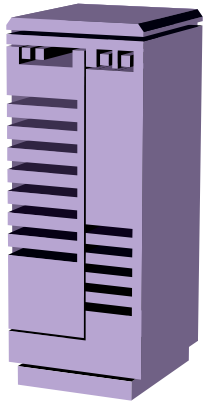
Internet



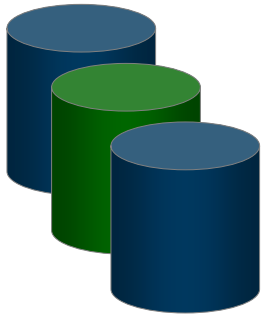
Web servers



Many, many web users all over



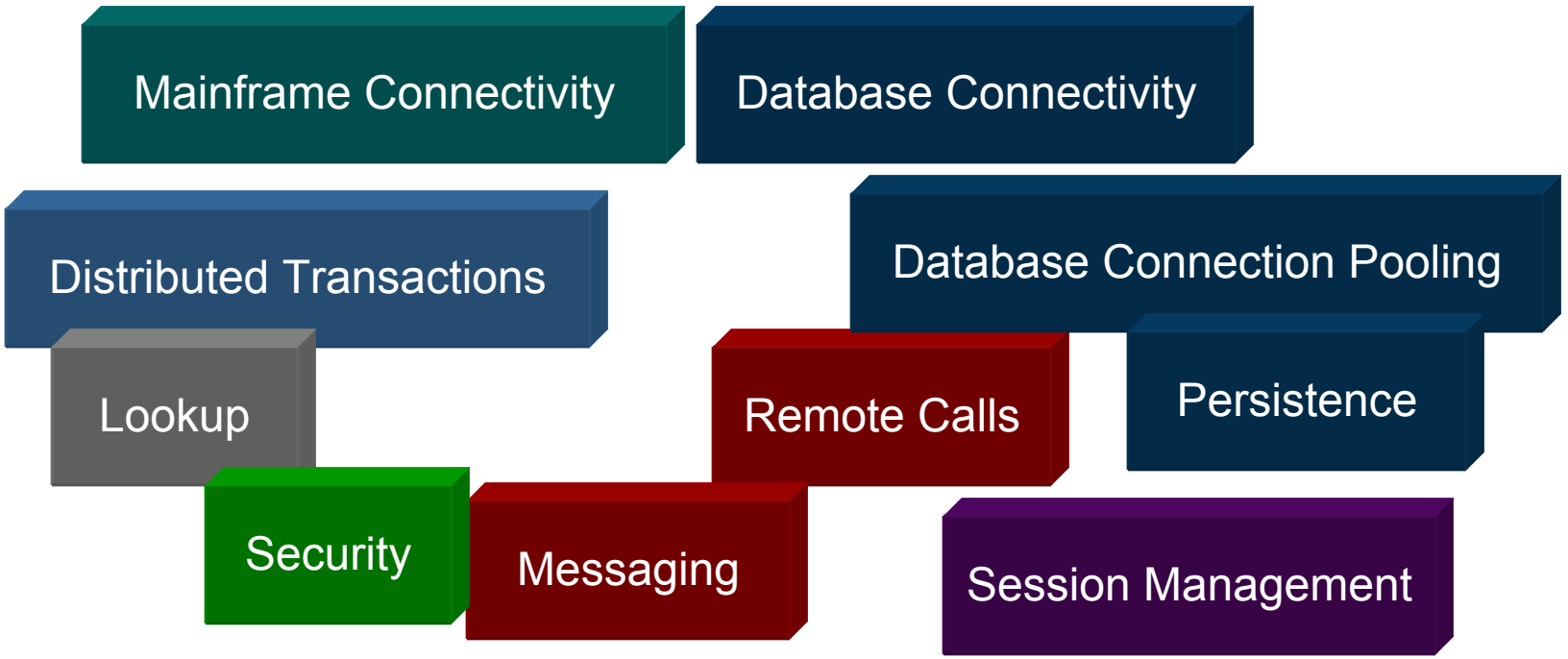
Application Servers



Many DBs, distributed and from different vendors

Challenge: Complexity

- Instead of writing application logic, enterprise project teams were spending up to 80% of the time writing system logic and utilities

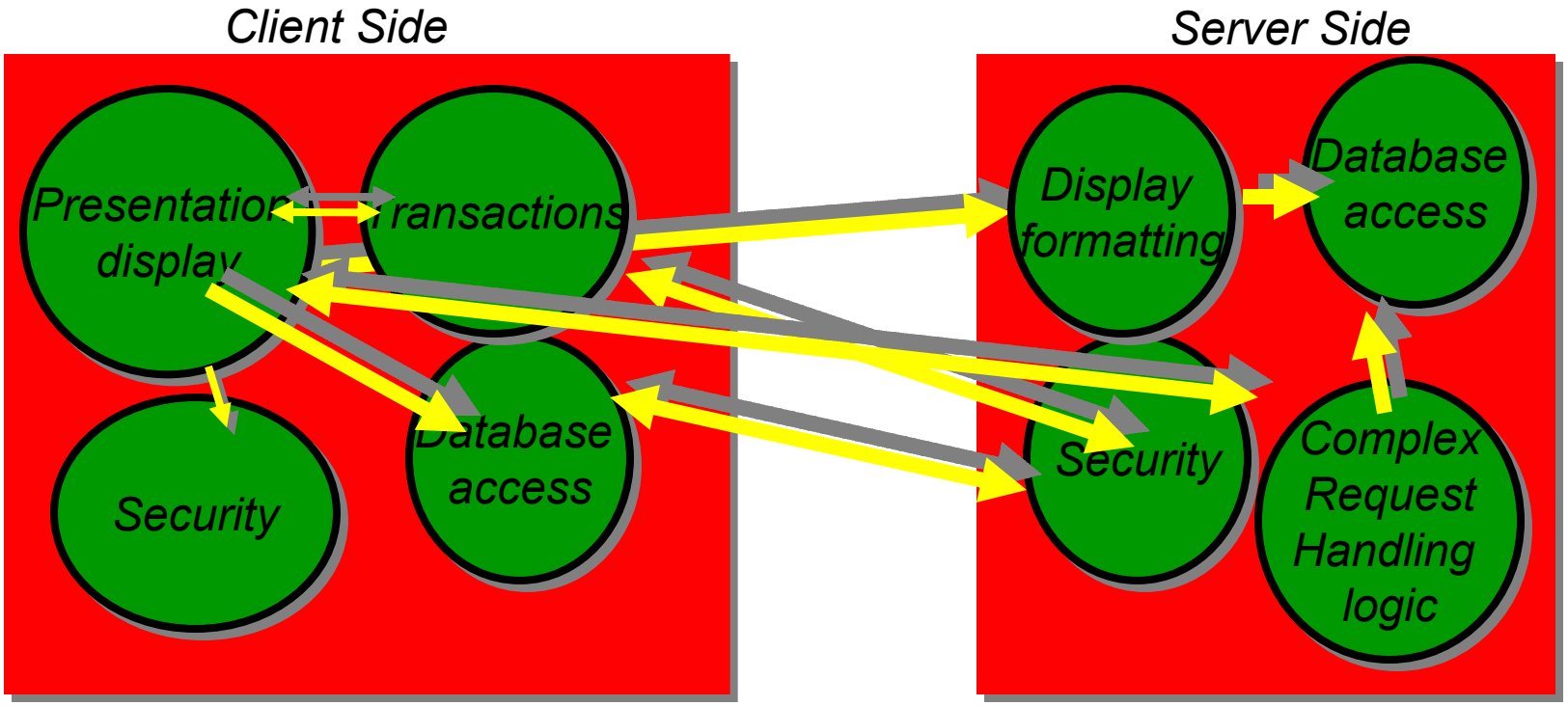


Challenge: Complexity

- For example, database connection pooling
  - Establishing and cleaning up connections to a database is expensive in terms of time and resource for an application
  - Best practice is to ‘pool’ a database connect for re-use by the same or another application using the same database
  - Writing the code to manager this pool must take into account many factors
    - When to connect?
    - How many?
    - Who has one?
    - Who needs one next?
    - When to disconnect?
    - Error conditions
- This is not application specific or business logic code – why spend development time writing it?

Challenge: Intertwined responsibilities

- Infrastructure, presentation, business logic, data access mixed together

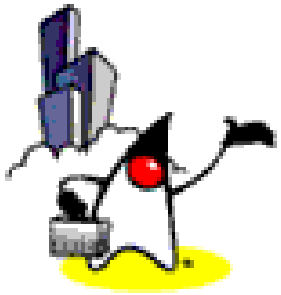


Solution: The Java 2 Platform

J2EE - Enterprise	J2SE – Standard	J2ME – Micro
▪enterprise-wide distributed applications ▪Web applications, etc.	▪client/server applications ▪desktop or distributed applications	▪embedded applications ▪from smart cards to set-top boxes

Java 2 Platform, Enterprise Edition (J2EE)

- Defined various standards for enterprise application development:
  - Architecture for enterprise application development
    - SDK (Enterprise Edition)
    - API
    - Tools
    - Documentation
    - Sun BluePrints Design Guidelines for J2EE
  - Contains a standard set of Java technologies for Java based enterprise applications
  - Released in Dec 1999
  - Current release is J2EE 1.4.
  - J2EE 5 (note the versioning name change) due out this year



History of J2EE

- J2EE 1.2 – Dec 1999
- J2EE 1.3 –Jul 2001
- J2EE 1.4 – Nov 2003
- Java EE 5 – soon (renamed from J2EE 1.5)

- Developed through the Java Community Process (JCP)
  - JSR Review, Early Draft Review, Public Review, Proposed Final, Final Release

- Executive Community members (Java EE 5):
  - Apache Software Foundation
  - Apple Computer Inc.
  - BEA Systems
  - Borland Software Corporation
  - Fujitsu Limited
  - Google Inc.
  - Hewlett-Packard
  - IBM
  - IONA Technologies PLC
  - Intel Corporation
  - JBOSS Inc.
  - Nortel
  - Oracle
  - SAP
  - Sun Microsystems, Inc.

The Benefits of J2EE

- Simplifies the complex task of writing distributed, reliable, scalable, secure applications..
  - Provides a common application model available to all developers as a starting point
  - Provides a standard platform for hosting applications (different yet same!)
- Increased portability between infrastructure vendors
  - Introduces generic APIs to provide compatibility across vendors
- A cleaner partition between development, deployment, and runtime
  - e.g. Database, server, other info specified generically by the developer. Actual details provided when application is deployed
- Enables componentization
  - Simplifies application maintenance and update
  - Improved division of labor among skill sets
  - Easier to add/replace individual pieces, custom or third party

J2EE 1.4 Specifications

Software Developer's Kit (SDK)	Java Virtual Machine base, with Java classes and basic routines required to execute Java applications. Java 2 Platform, Standard Edition (J2SE).
Servlets & Java Server Pages (JSP)	Server applications that execute within a web application server that supports dynamic HTML generation.
Enterprise Java Beans (EJB)	Server transactional components that are reusable and provide portability across applications servers while implementing transaction services.
Remote Method Invocation / Internet Inter-ORB Protocol (RMI/IIOP)	RMI creates remote interfaces for Java-to-Java application communications. CORBA IIOP used for ORB (Object Request Broker) communications.
Java Interface Definition Language (IDL)	Creates remote interfaces to support Java-to-CORBA application communications. Includes an IDL-to-Java compiler and an ORB.
Java Naming and Directory Interface (JNDI)	JNDI provides access to naming and directory services such as LDAP, Novell Directory Services, and CosNaming.
Java Transaction Specification (JTS) and API (JTA)	A distributed transaction management service and associated API based on CORBA's Object Transaction Service.
Java Database Connectivity (JDBC)	JDBC database access API provides uniform access to relational databases such as DB2, Oracle, Sybase, and SQL Server.
Java Messaging Service (JMS)	JMS supports asynchronous communications using either a reliable queuing or publish/subscribe programming model.

J2EE 1.4 Specifications (contd)

Java Mail	Provides a protocol-independent framework to build mail applications. Requires the JavaBeans Application Framework API.
JavaBean Activation Framework (JAF)	JAF provides standard services to determine the type of an arbirtrary piece of data and activate an appropriate bean component to manipulate the data.
J2EE Connector Architecture (JCA)	Provides schema mapping and persistence management to underlying procedural data systems -- including IMS, CICS, etc.
Java Authentication and Authorization Services (JAAS)	Provides a security framework for the runtime and applications.
Java API for XML processing (JAXP)	Provides a framework for pluggable XML parsers and XSLT transformation engines.
WebServices for J2EE (JSR109)	Defines the deployment and execution model for web services
Java API for XML based RPC (JAX-RPC / JSR101)	Defines the deployment and execution model for JAX-RPC based clients and services.
SOAP with Attachment API for Java (SAAJ)	Provides an API to manipulate SOAP messages and attachments
Java API for XML based registries (JAXR)	Provides an API to access XML based registries such as ebXML and UDDI.
Java Management Extension (JMX)	Defines the APIs used to manage a J2EE environment

J2EE Required APIs

Optional Package	App Client	Applet	Web	EJB
EJB 2.1	Y[a]	N	Y[b]	Y
Servlet 2.4	N	N	Y	N
JSP 2.0	N	N	Y	N
JMS 1.1	Y	N	Y	Y
JTA 1.0	N	N	Y	Y
JavaMail 1.3	Y	N	Y	Y
JAF 1.0	Y	N	Y	Y
JAXP 1.2	Y	N	Y	Y

Several Java 2 Standard Edition (J2SE) Enterprise APIs are required by J2EE (Java IDL API, JDBC API, RMI-IIOP API,JNDI API, JAXP API, JAAS API)

J2EE Required APIs

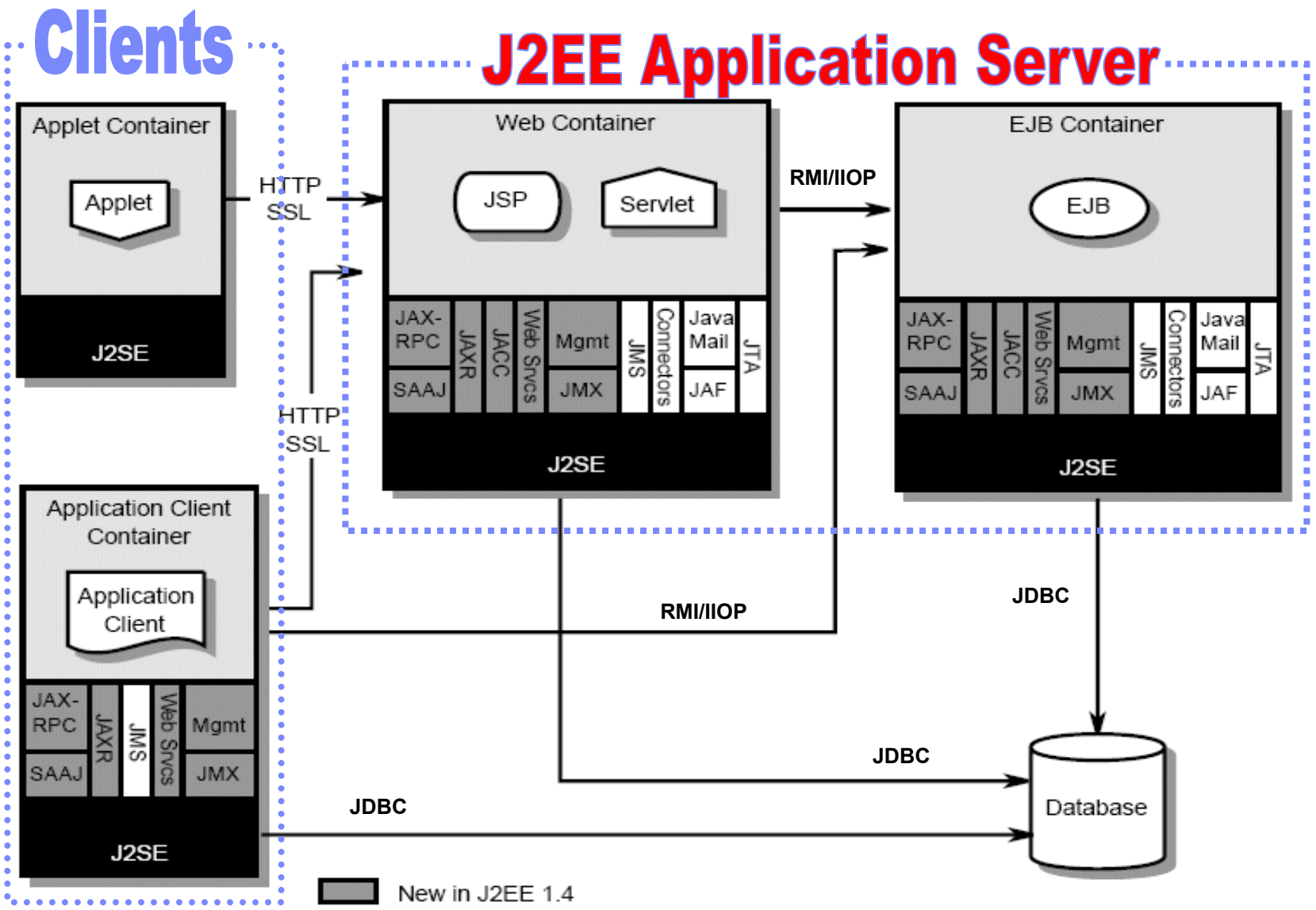
Optional Package	App Client	Applet	Web	EJB
Connector 1.5	N	N	Y	Y
Web Services 1.1	Y	N	Y	Y
JAX-RPC 1.1	Y	N	Y	Y
SAAJ 1.2	Y	N	Y	Y
JAXR 1.0	Y	N	Y	Y
J2EE Management 1.0	Y	N	Y	Y
JMX 1.2	Y	N	Y	Y
J2EE Deployment 1.1[c]	N	N	N	N
JACC 1.0	N	N	Y	Y

Java EE 5

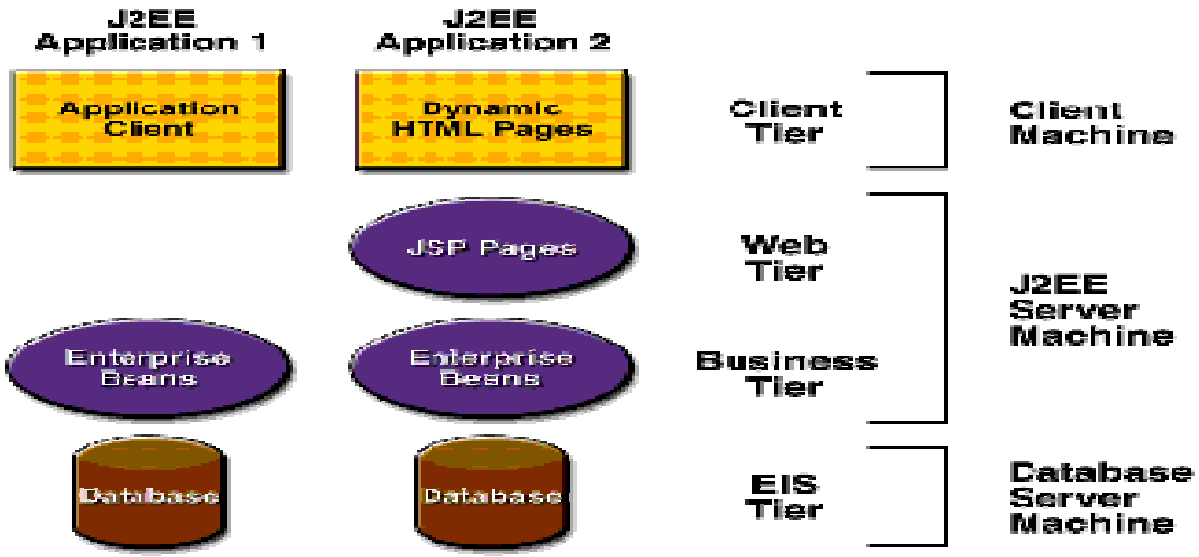
- Overall Goal - Ease of development
  - Take advantage of J2SE 1.5/5.0 Platform (Annotations)
  - JSR-181 (Web Services Metadata for the Java Platform)
  - JSR-220 (EJB 3.0)
    - POJO Based EJB
  - JSR-222 (JAXB 2.0)
  - JSR-224 (JAX-RPC 2.0)
  - JSR-127 (JavaServer Faces)
    - Adding JSF to J2EE spec. Supported today by IBM Tools

J2EE 1.4 Platform Architecture

Source: J2EE 1.4 Specification



J2EE Application Tiers

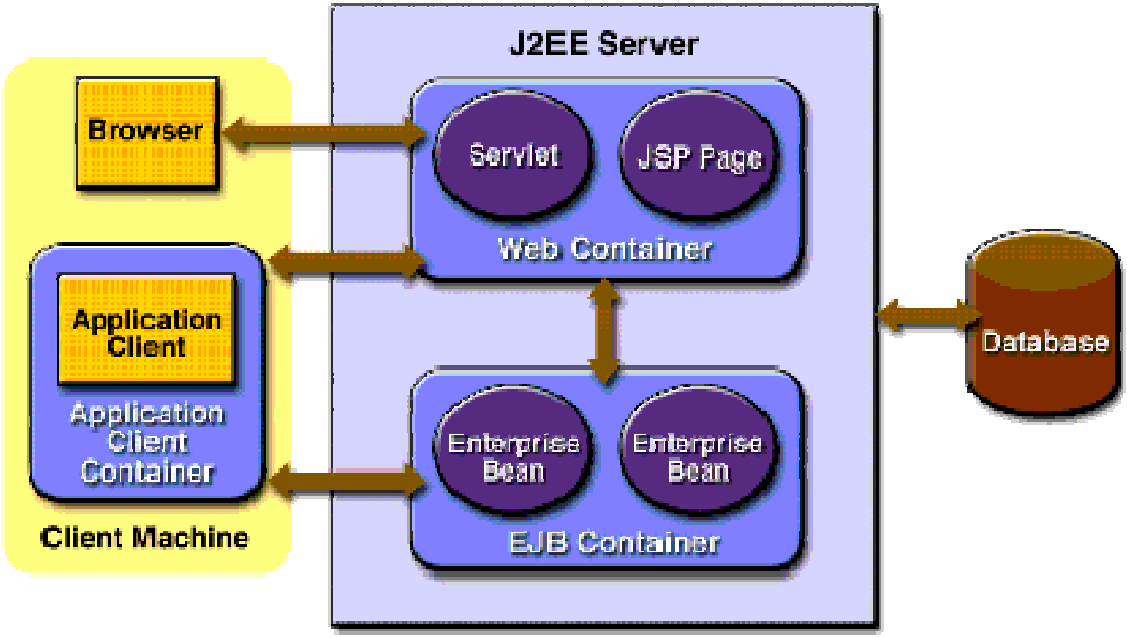


A typical J2EE application includes 3 or 4 tiers:

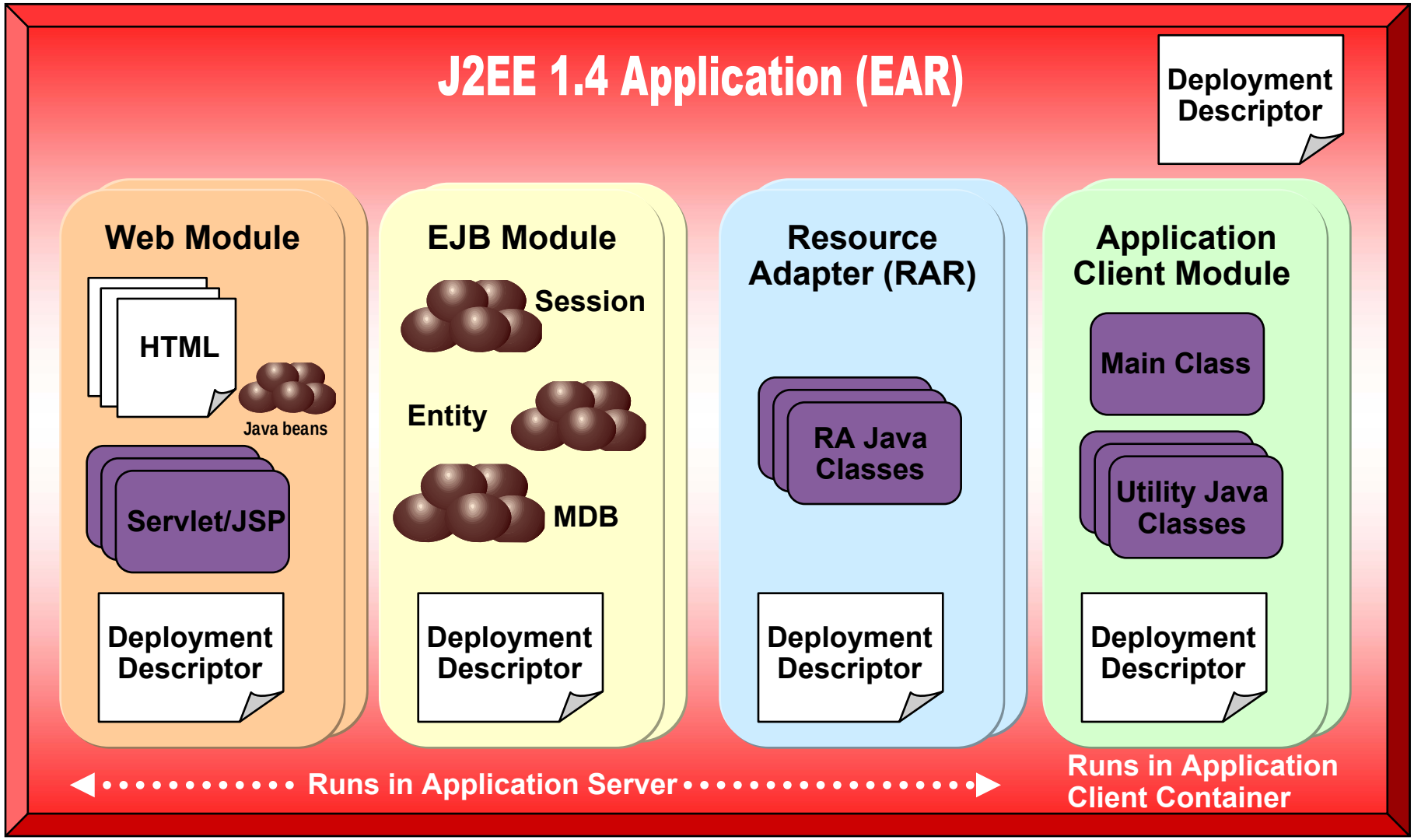
1. Client Tier – client interface (web browser, Java applet or stand alone Java application client)
2. Web Tier – Servlets and JSPs
3. Business Tier - Enterprise Java Beans (EJB) – encapsulate business logic
4. Enterprise Information System (EIS) Tier – databases, Enterprise Resource Planning (ERP) systems and other legacy systems.

J2EE Containers

- Each tier has a J2EE container which provides the necessary services to the components (servlets, JSPs, EJBs)
- Container services:
  - Security
  - Transactions and state management
  - JNDI
  - Remote connectivity
  - Resource pooling



J2EE 1.4 Application Components



The Model-View-Controller (MVC) pattern

- The MVC design pattern is commonly used in J2EE application development

- The goal of the MVC design pattern is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller).

  Model - The Model object knows about all the data that need to be displayed. It also knows about all the operations that can be applied to transform that object. However, it knows nothing whatever about the GUI.

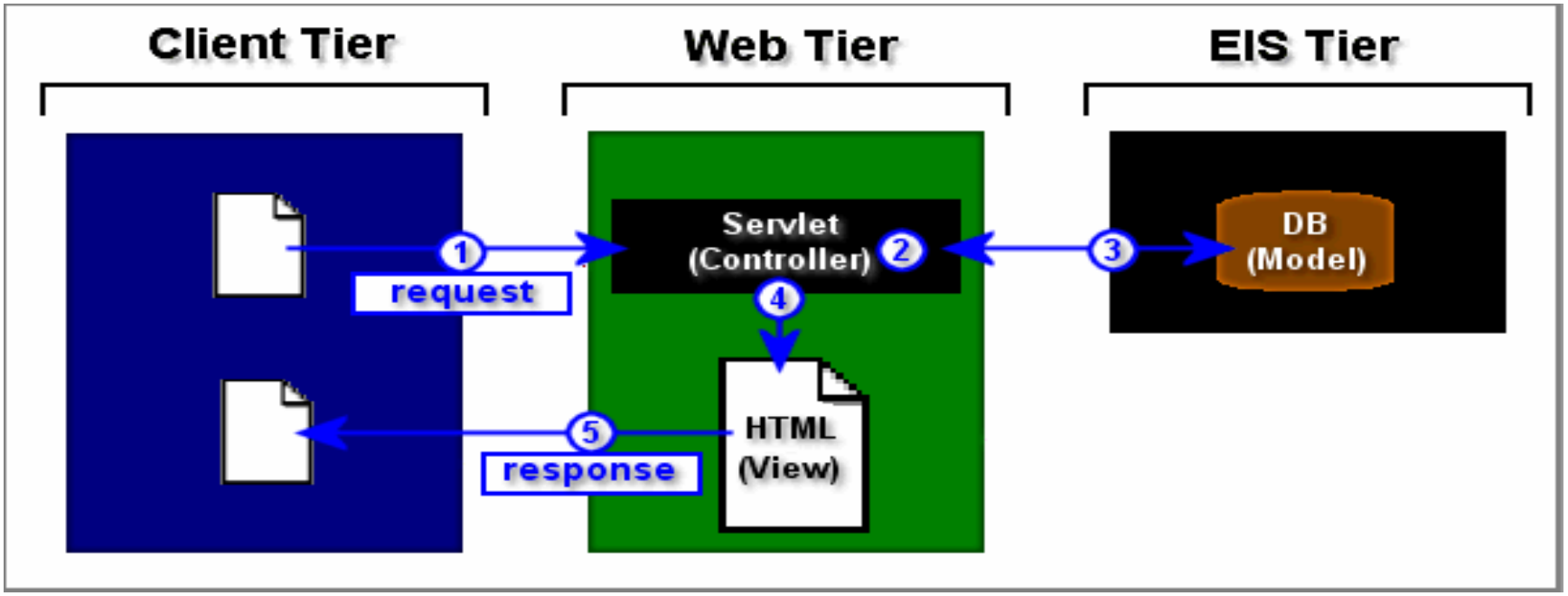
  View - The View object refers to the model. It uses the query methods of the model to obtain data from the model and then displays the information. A view renders the contents of a model.

  Controller - The Controller object knows about the physical means by which users manipulate data within the model. A controller translates interactions with the view into actions to be performed by the model. (Example a user submitting a HTML Form.)

A Servlet …

- Is a Java class that acts as the middle layer between the client tier and the Enterprise Information System (EIS) tier.
- Provides the developer with a server-side programming environment which can be used to respond to requests, access data and process results.
- Uses a request-response architecture. Most commonly used to respond to HTTP requests from web clients.

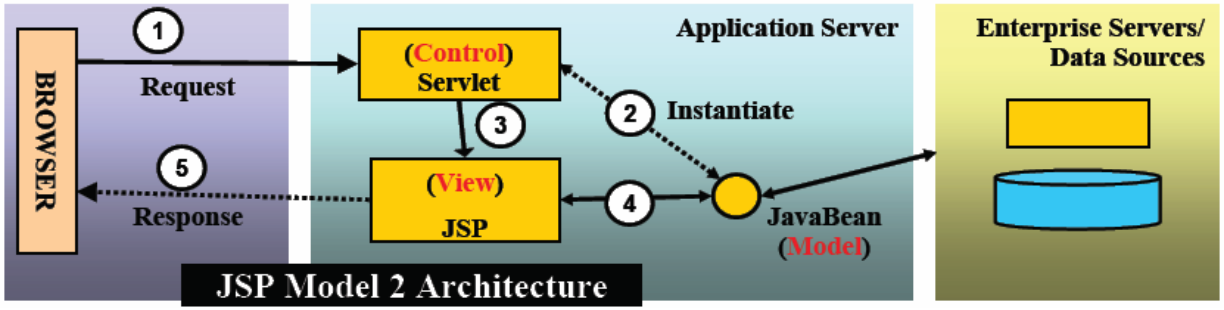
How do Servlets work …



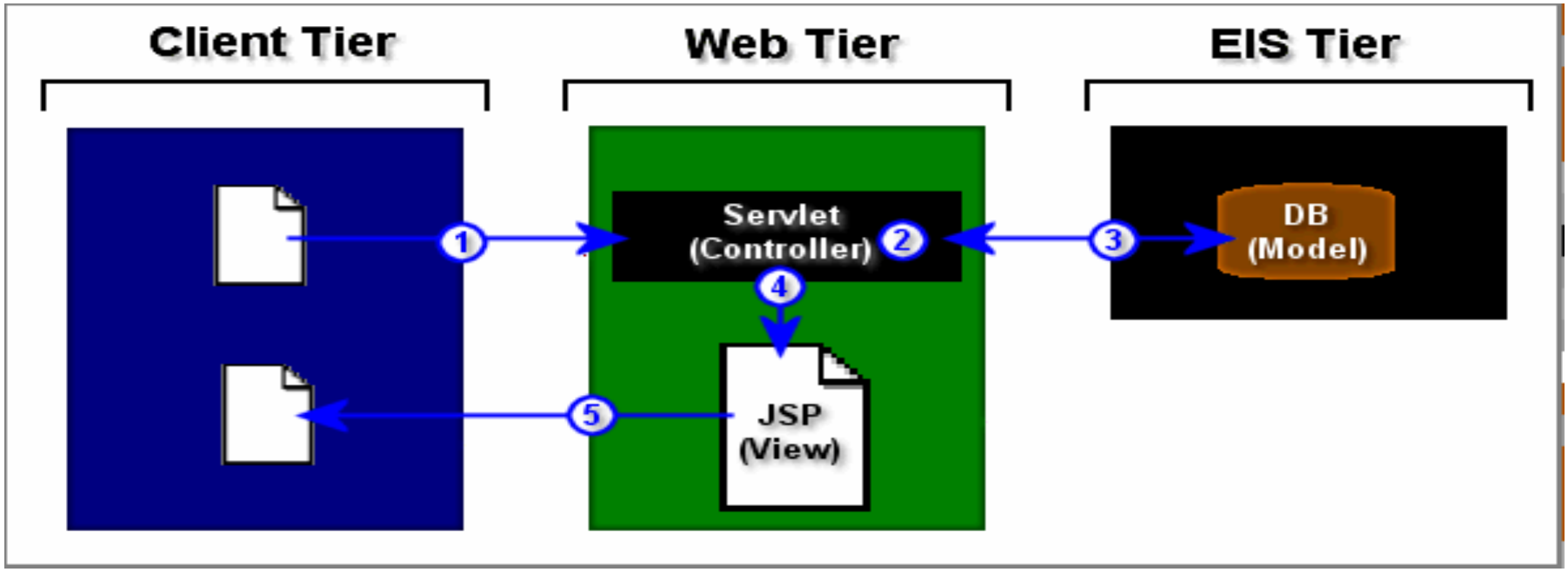
- Receives a request from the client tier: either explicit data entered by an end user or implicit HTTP request data. The servlet begins its execution thread.
- Processes the request: can talk to a database, execute an Remote Method Invocation (RMI) or compute the response directly.
- Sends a response back to the client tier: either explicit data such as documents or implicit HTTP response data.

Java Server Pages (JSP)

- Server-side technology, and are an extension to the Java servlet technology.
- JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements --the actual design and display of the page.
- Embedded in the HTML page, the Java source code and its extensions help make the HTML more functional, being used in dynamic database queries, for example.
- JSPs are not restricted to any specific platform or server.
- JSP technology integrates numerous Java application technologies, such as Java servlet, JavaBeans, JDBC, and Enterprise JavaBeans. It also separates information presentation from application logic and fosters a reusable-component model of programming.



How do JSPs work ….



- The servlet is used to complete the program logic and computation. The results are then forwarded to a JSP for output to the requesting application.
- JSPs are compiled when first executed and recompiled when changed, a JSP can be modified to reformat the results without the need to restart the Java Servlet Container.

Servlets vs JSPs

Servlets	Java Server Pages
Written in Java.	Written in HTML or XHTML with Java code scripting
Java servlets use a request-response architecture.	Because JSPs are based upon the same API as Java servlets, the same request and response objects that are available to servlets are also available to JSPs as implied objects.
Changes to code require a recompile and server restart.	Changes to code do not require a server restart. JSPs are compiled when first executed and automatically recompiled when changed.
Presentation is included with the program logic.	Presentation is separated from programming logic.
Presentation output is difficult to code.	Presentation output is easily added and changed using a GUI interface.

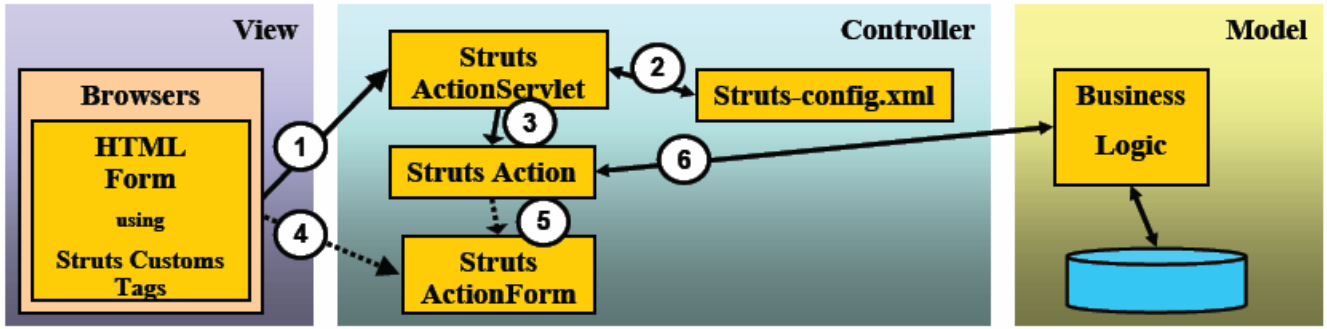
Struts

The Struts framework is the View and Controller components of MVC. Struts has three major components:

- **Action beans**
- **ActionServlet**
- **ActionForm** beans and custom tags.

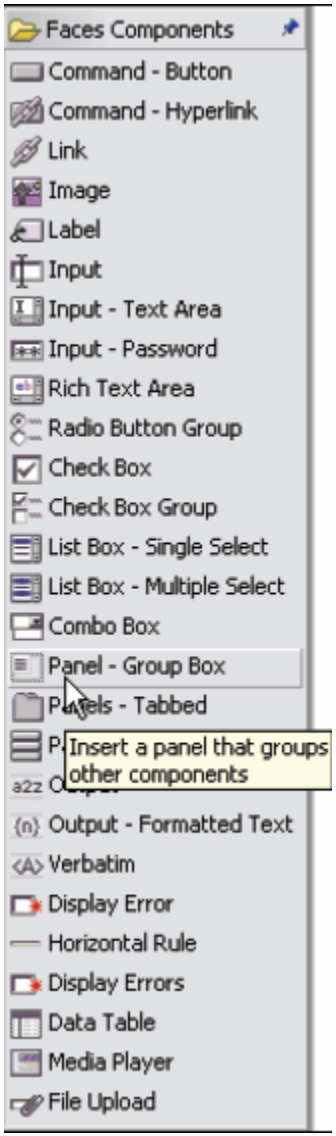
Struts provides a large set of JSP custom tags that support the ActionForm beans. The custom tags support:

- Pre-population of a HTML form with values taken from an **ActionForm** subclass.
- **Internationalization**, such as providing text that is determined by the user's locale.
- **Logic**, such as showing a different title for a page based on how it is used.

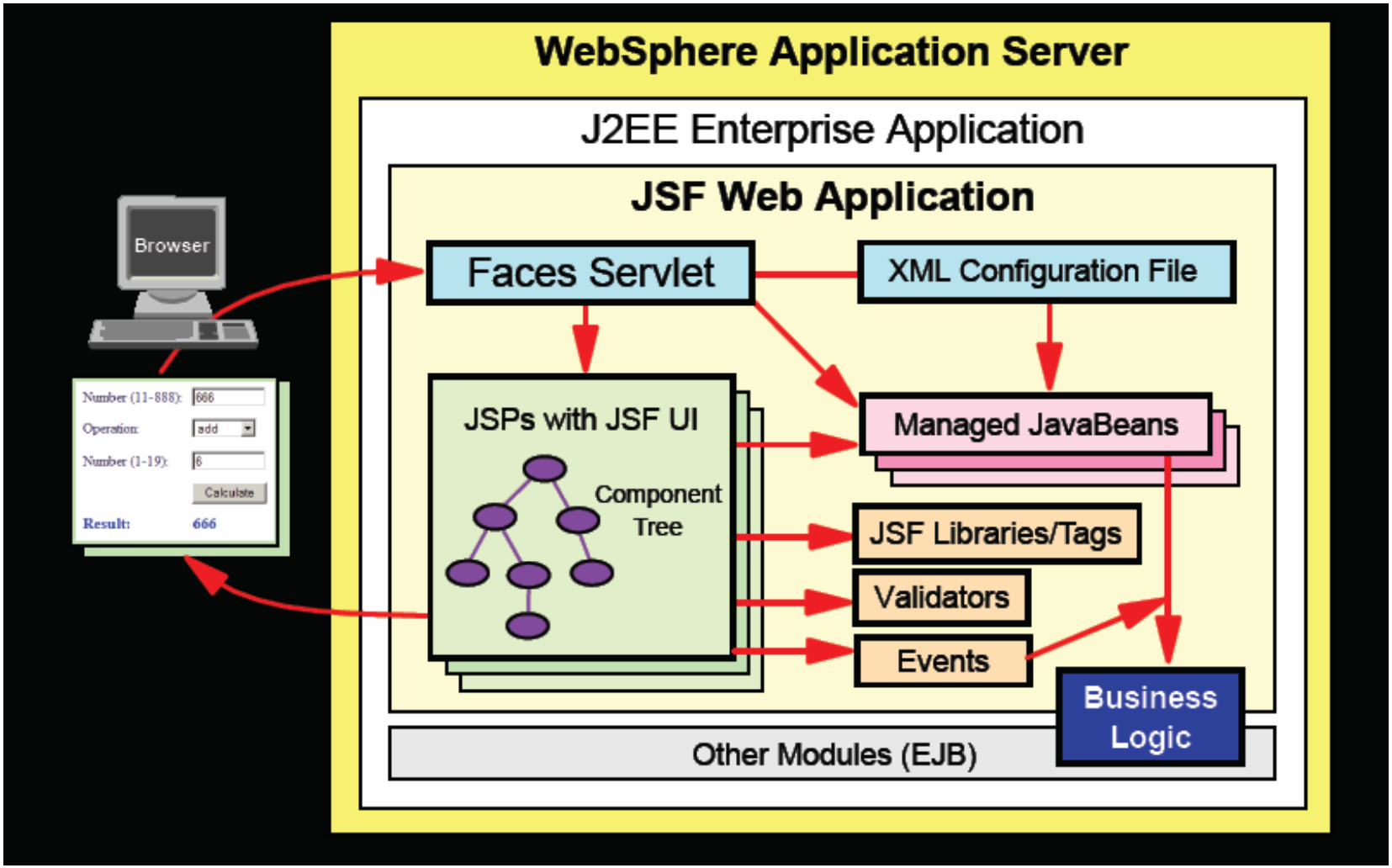


Java Server Faces (JSF)

- JavaServer Faces technology simplifies building user interfaces for JavaServer applications. Developers of various skill levels can quickly build Web applications by:
  - assembling reusable UI components in a page
  - connecting these components to an application data source
  - wiring client-generated events to server-side event handlers
- JavaServer Faces technology includes:
  - A set of APIs for representing UI components and managing their state, handling events and input validation, defining page navigation, and supporting internationalization and accessibility.
  - A JavaServer Pages (JSP) custom tag library for expressing a JavaServer Faces interface within a JSP page. Follows the MVC (model-view-controller) paradigm
- Server-side UI components that respond to client events
  - Many standard components provided by the framework
  - Extensible: IBM components in Application Developer
- Decoupling of UI components and rendering
- Targeted to Web developers with little Java background
  - Compete with .Net WebForms

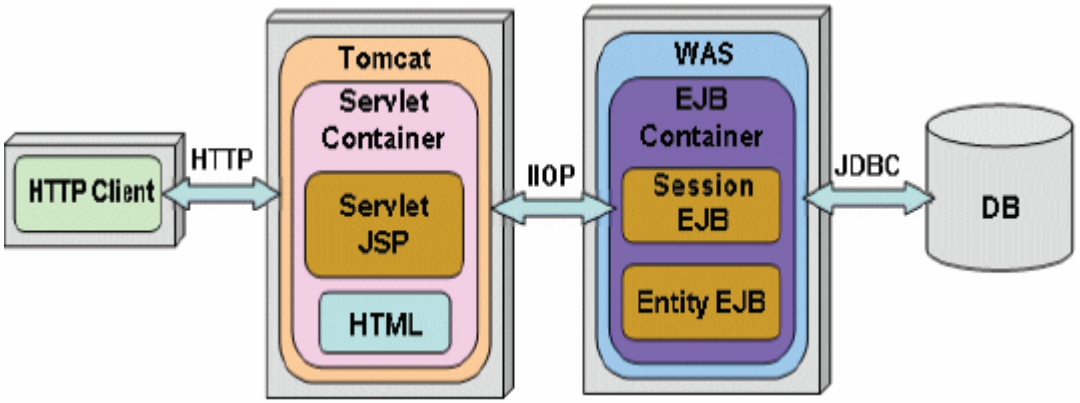


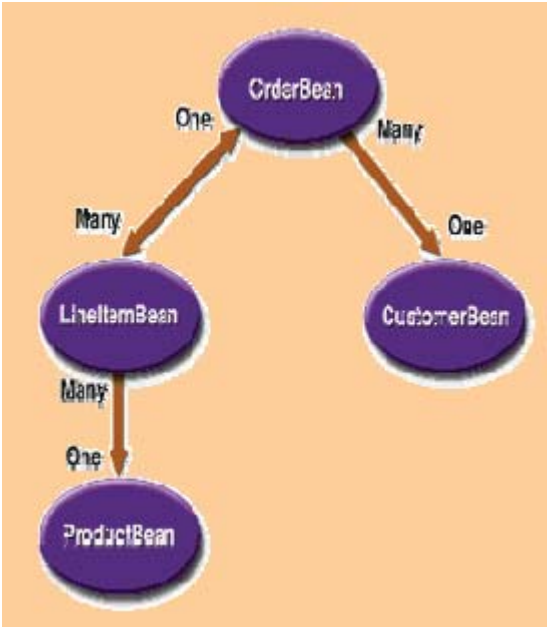
JSF Application



EJB

The encapsulation of business logic into business objects and a component architecture for multi-tier client/server systems. Because EJB systems are written in Java, they are platform independent. Being object oriented, they can be implemented into existing systems with little or no recompiling and configuring. The EJB server-side model simplifies the development of middleware applications by providing automatic support for services such as transactions, security, and database connectivity.

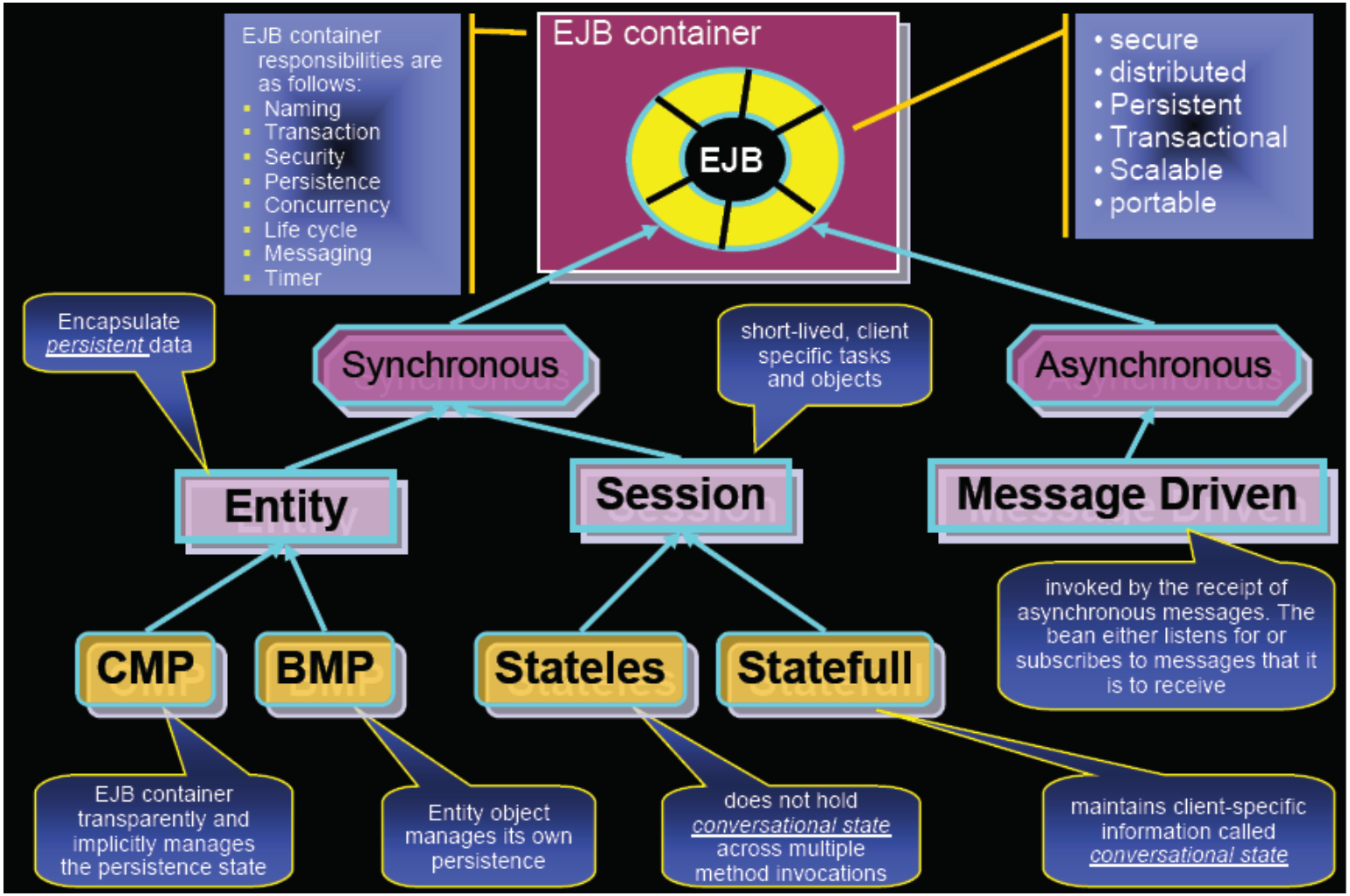




Java Archive File (JAR)

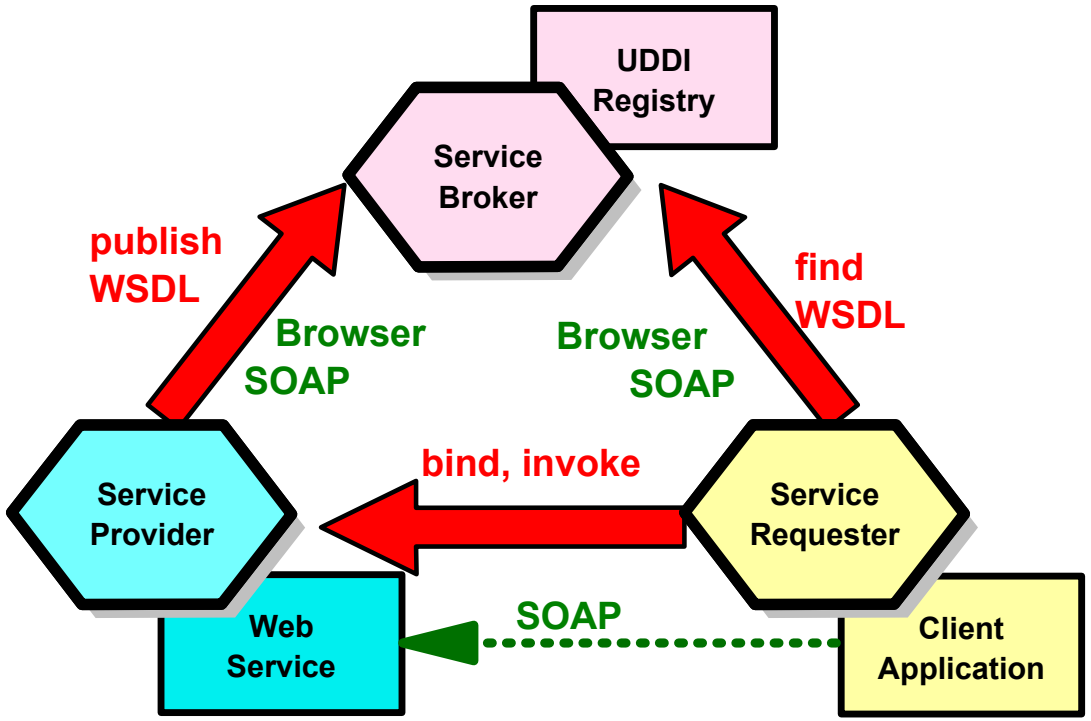
- **EJB class files:** OrderBean, LineItemBean, CustomerBean, ProductBean classes
- **Deployment Descriptor:** Life Cycle, Persistence, Transaction, Security

EJB Types



Web Services

- Architecture for
  - Application to application
    - Communication
    - Interoperation
- Definition:
  - Web Services are software components described via WSDL that are capable of being accessed via standard network protocols such as SOAP over HTTP

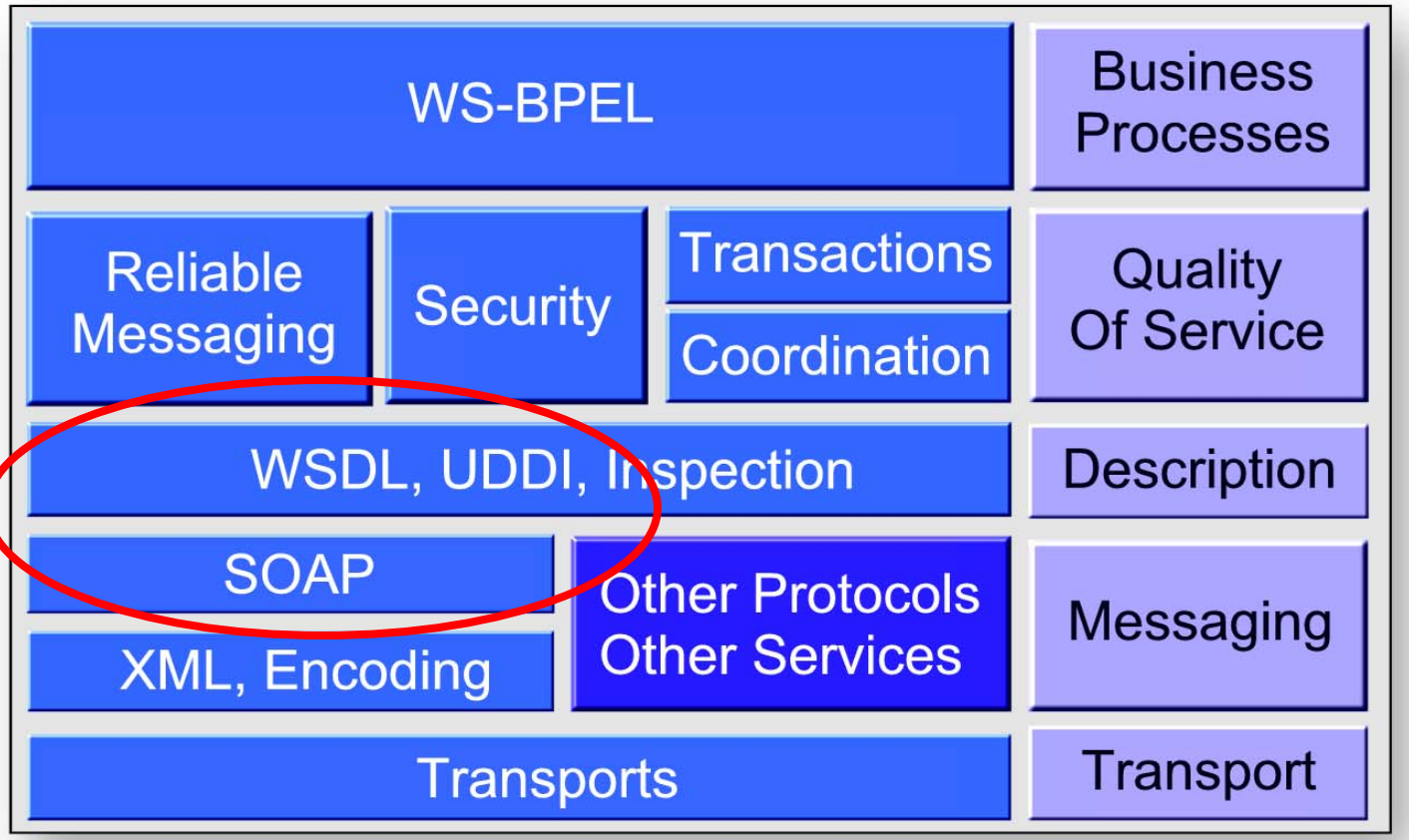


The entire industry is agreeing on one set of standards !!

What is a Service …?

- Consider interacting with a waiter in a restaurant:
  - Order food
  - Brings food
  - Refills glasses
  - Brings bill
- **How** the waiter achieves the task is not important, only that he **what does**
- Waiter acts as an Interface to the kitchen
- Waiter is your **view** of the restaurant service

Web Services Standards …

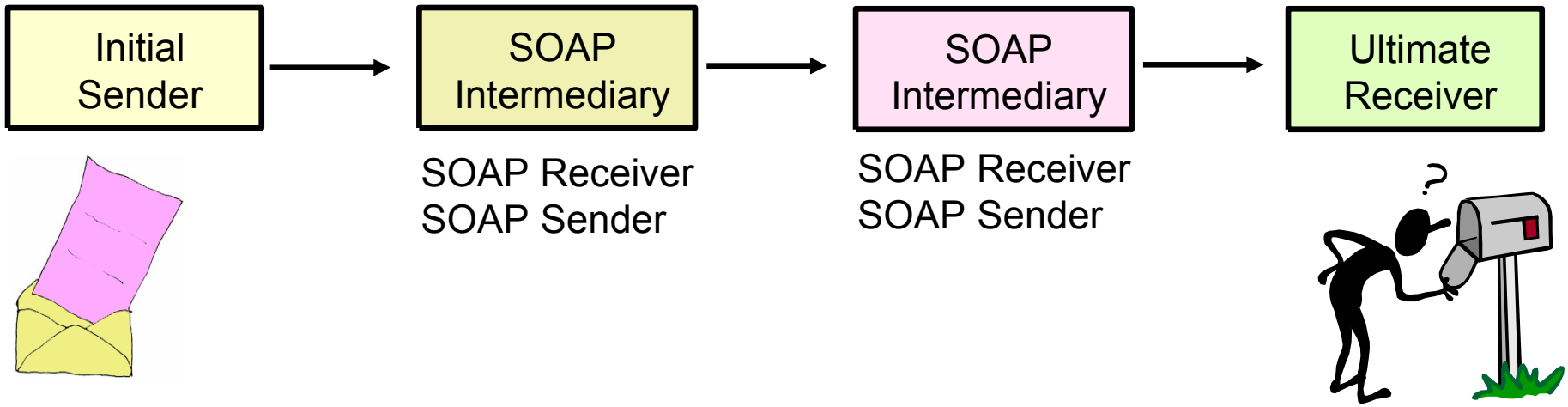


SOAP

- What is SOAP?
  - An XML-based protocol for exchange of information in a decentralized, distributed environment
  - An open standard whose main goal is to facilitate interoperability
  - A protocol which is not tied to any operating system, transport protocol, programming language, or component technology
- How many versions of SOAP are there?
  - SOAP 1.0
  - SOAP 1.1 – W3C Note dated 05/08/2000
  - SOAP 1.2 – W3C Recommendation dated 06/24/2003
- What type of HTTP request does SOAP use?
  - HTTP POST method
- What do the letters stand for?
  - Simple Object Access Protocol (SOAP 1.1)
  - Nothing (SOAP 1.2)

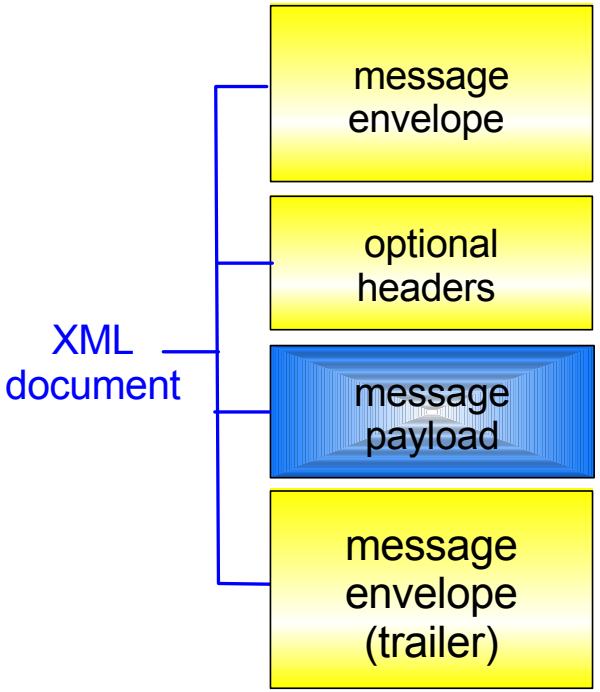
SOAP Message Path

- SOAP: A container for a message
- A SOAP message path is the set of SOAP nodes though which a single SOAP message passes
- Initial SOAP sender (originates message)
- Zero or more SOAP intermediaries
- Ultimate SOAP receiver (intended receiver)



SOAP: Parts

- Can have 0 or more headers, exactly one body



- ***XML Message Envelope***
  - service requested
  - routing information
  - message type
  - date/time stamp
- ***XML Message Headers***
  - authentication
  - transaction context
- ***XML Message Payload***
  - data understood by target application
- ***XML Message Trailer***
  - closing tags
  - optional message digest

SOAP spec defines how to do this!

WSDL

- **WSDL – Web Services Description Language**
  - Open Standard
  - XML document describing what a Web Service can do, where it resides, and how to invoke it
  - Machine readable, generated, used by IDEs
  - Similar in purpose to IDL, but in XML form
- **One or multiple XML documents**
  - Service Interface – input and output parameters, operations and methods
  - Service binding – protocol binding
  - Service implementation – location of service
- If you have a WSDL file, you **know** how to interact with a service!!

WSDL: Logical Contents

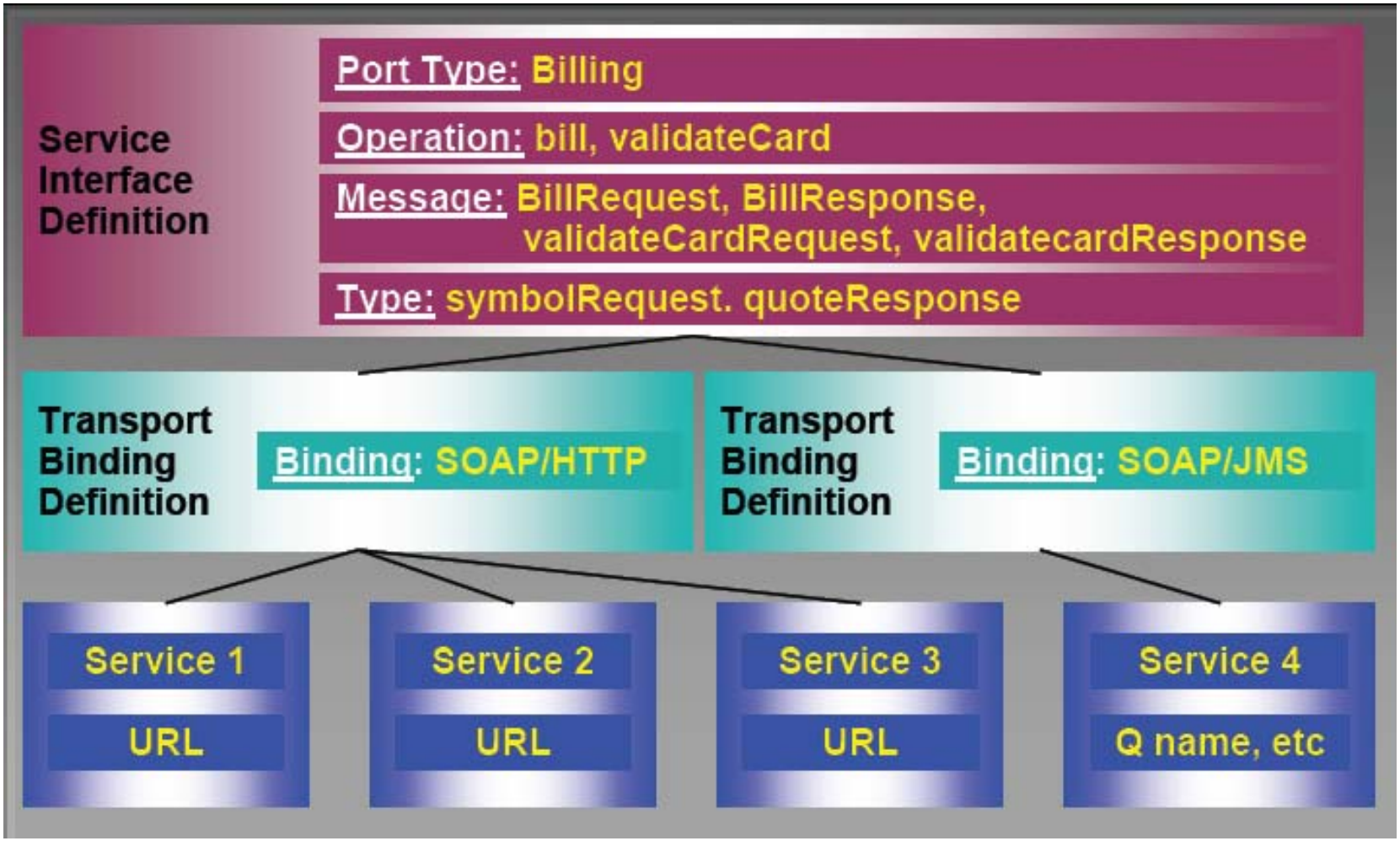
- **Service Interface**
  - Operation (business functions)
    - Input Message (0 or 1)
      - 1 or more parts
      - parts may be simple or complex
      - complex parts may have multiple elements
    - Output Message (0 or 1)
      - 1 or more parts
      - parts may be simple or complex
      - complex parts may have multiple elements
- **Service binding –** definition of the physical implementation of the service interface
- **Service implementation –** location of the service

UDDI

Universal Description, Discovery, and Integration (UDDI)

- Is a global, platform-independent, open framework that enables service providers and service consumers to discover each other
- Provides a mechanism to define how the service providers and service consumers interact with each other over a public shared network infrastructure
- Is a major cross-industry effort driven by platform providers, software developers, marketplace operators, and business leaders that comprehensively addresses the requirements of a service-centric computing and, in turn, an SOA
- Enables sharing of information between the service providers and service consumers in a global registry that accelerates the global adoption of service-centric computing

How do the pieces fit?



WSDL –Operations …

- Operations describe discrete functions that can be performed against the Service
- For a waiter:

Operation	Input	Output
Tell me specials		Todays specials are ...
Bring me food	Type of food	Here is your meal
Bring me check		Here is the check
Here is payment	Tip	

- For Technology
  - Similar to Functions or Subroutines
  - Messages are like parameters and return types
- We do not care **how** the operation is performed

WSDL –Messages …

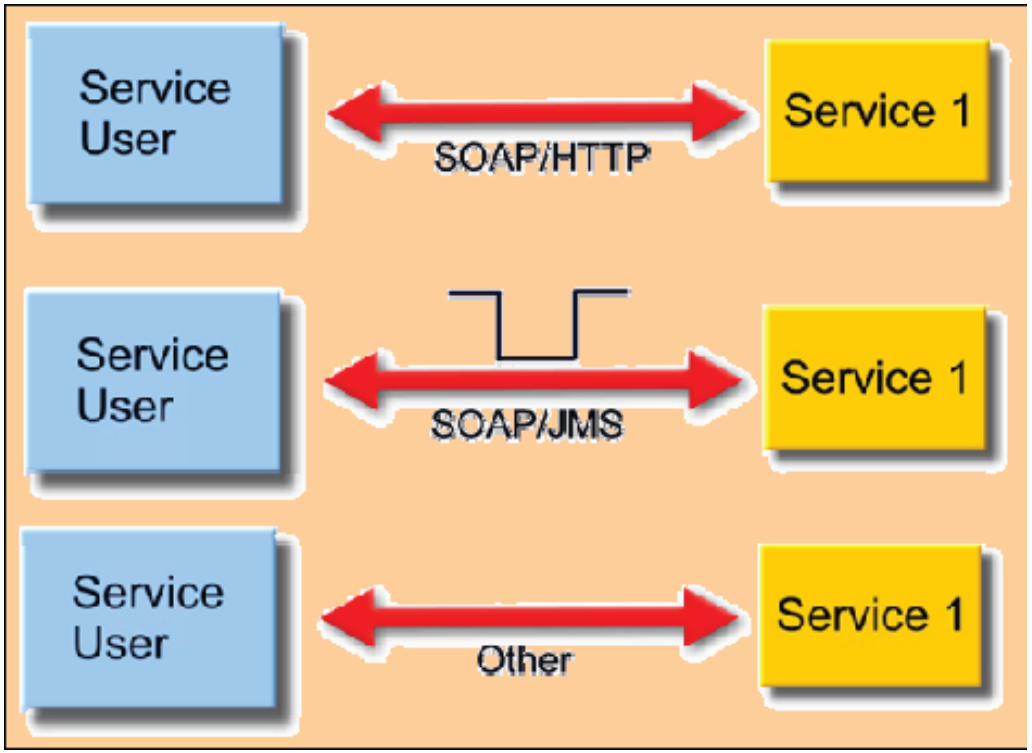
- Description of data to be send and received from the Service
- Described at high level in XML
- Supports description of every conceivable data type and structure

Strings	Integers
Floating Point	Booleans
Bytes	Arrays
More ...	

When you know the Message for a Service, you know how to send the data the Service expects.

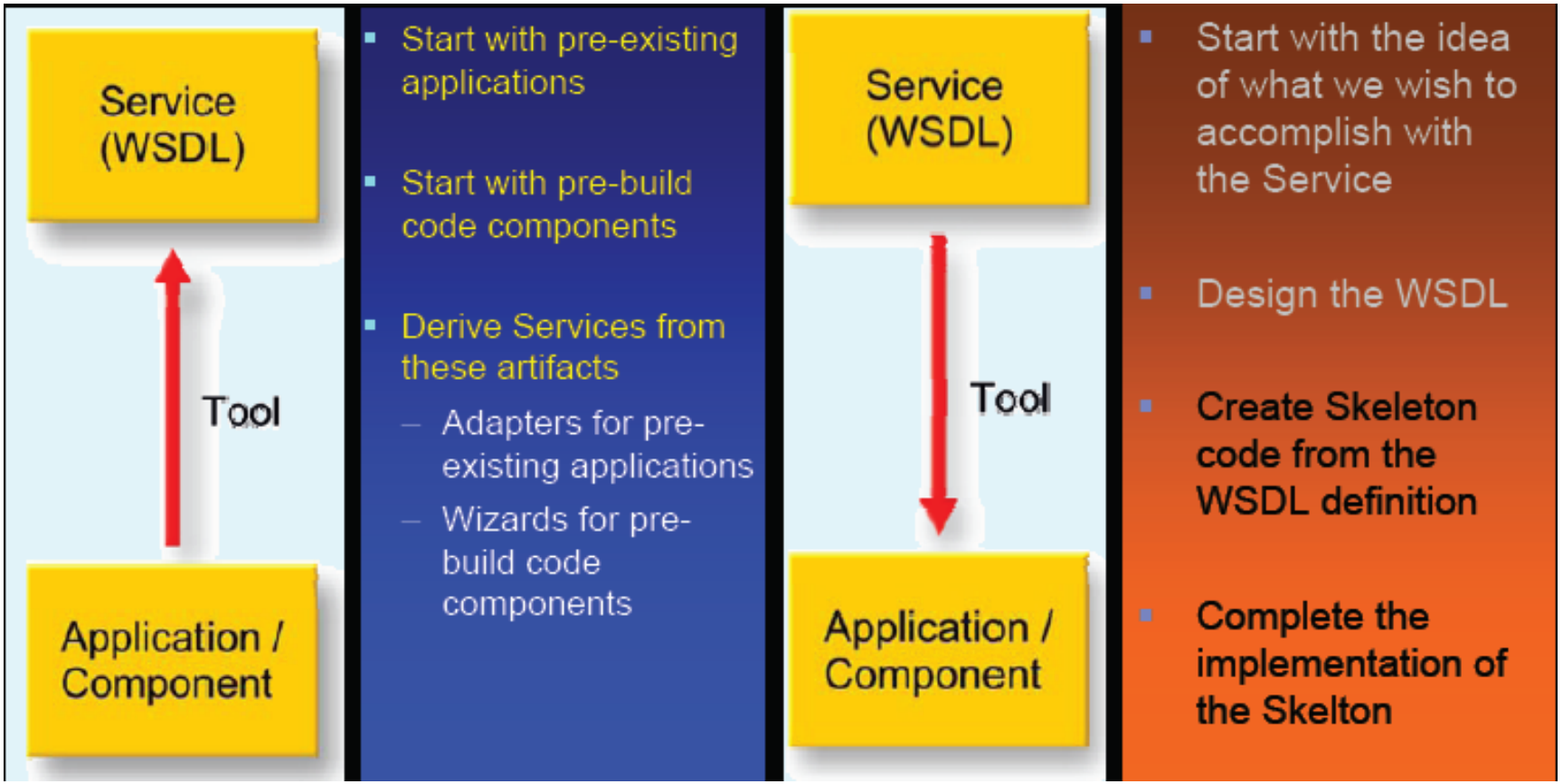
Techniques for remote Services …

- Many options available for invoking distributed services
- Examples:
  - SOAP/HTTP
  - SOAP/JMS
  - EJB Bindings
  - Java Bindings
- Hidden from user in WSDL description



Generating Services

Bottom Up Approach



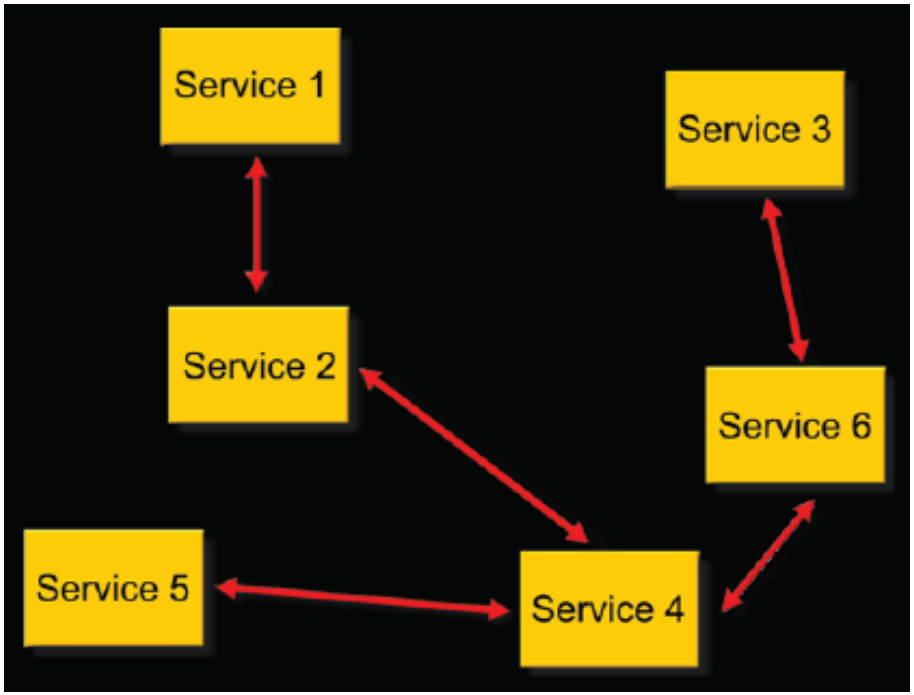
Top Down Approach

Composing Software Services …

- Software Solutions become composable
- Services become building blocks
- Solutions are now part of a new paradigm:

Service Oriented Architecture

A set of architectural principles and patterns which address characteristics such as modularity, encapsulation, loose coupling, separation of concerns, reuse, composable and single implementation



This new idea of composing software solutions from Service building blocks is called the Service Oriented Architecture

Services Oriented Architecture Changes the Game …

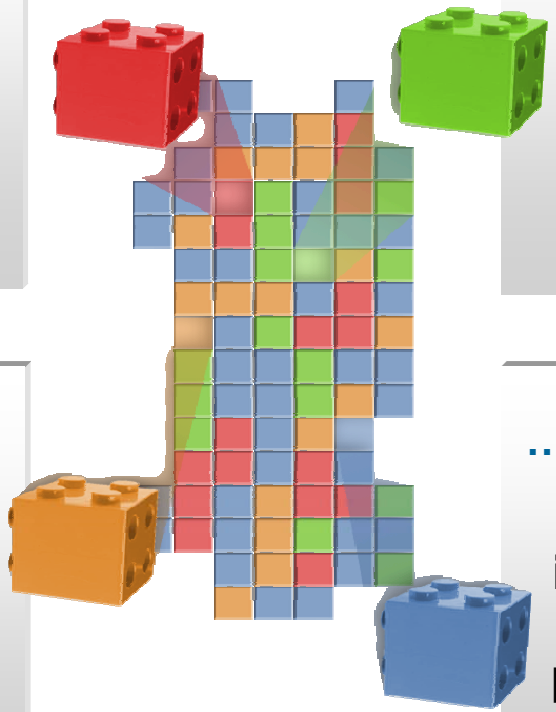
What is...

… a service?

A **repeatable business task** – e.g., check customer credit; open new account

… service orientation?

A way of integrating your **business as linked services** and the outcomes that they bring



… service oriented architecture (SOA)?

An IT **architectural style** that supports service orientation

… a composite application?

A set of **related & integrated** services that support a business process built on an SOA

New SOA Technologies

SCA – Service Component Architecture

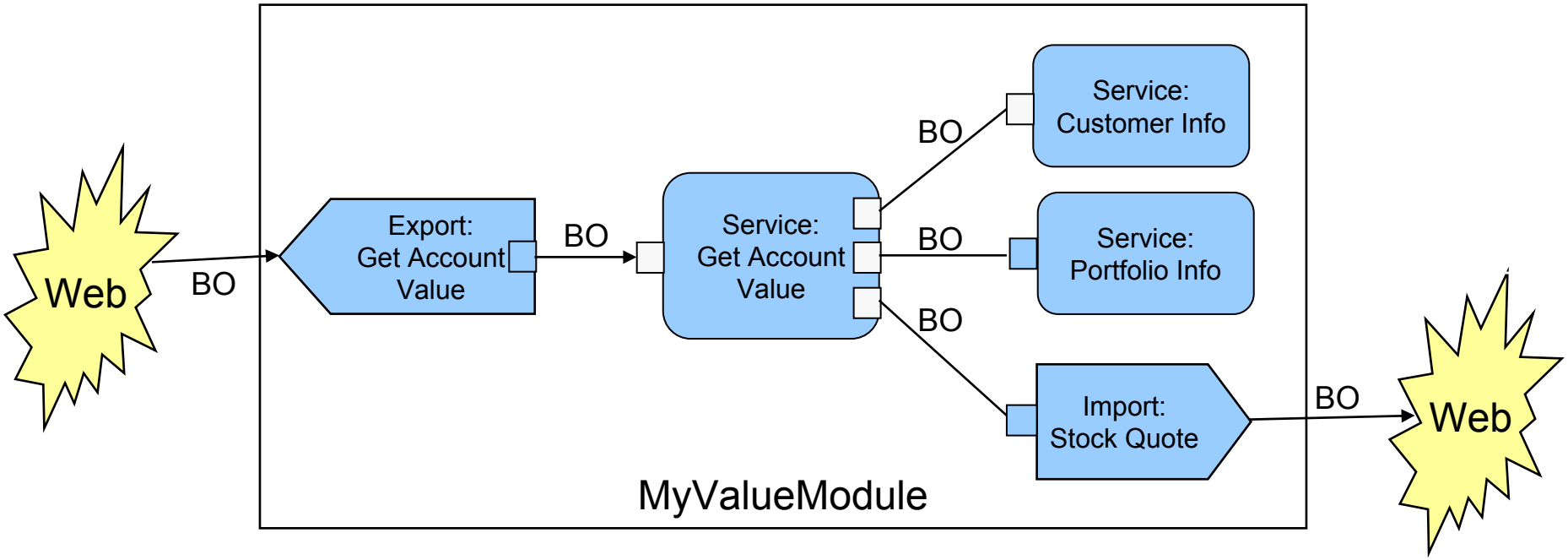
- SCA is our ***service-oriented component model***
- ***An abstraction*** that covers stateless session EJBs, Web services, POJOs, BPEL4WS processes, database access, EIS access, and so on.
- ***Separates "business logic" from "infrastructure logic"***
  - application programmers can focus on business problem
  - application logic can be much more portable
  - not everyone has to be an IT infrastructure expert
- Covers both ***usage of services*** and ***development of services***
- ***A uniform model for application programmers and for tools***
  - Enables advanced, domain-specific, task-oriented tools
- Uniform programming model for ***"programming in the small"*** (application development) and ***"programming in the large"*** (application integration)

New SOA Technologies

SDO - Service Data Object

- SDO is the analogous ***abstraction for Data***
- ***Single abstraction*** (common API) across JDBC ResultSet, JCA Record, XML DOM, JAXB, Entity EJB (with copyHelpers), CMI (for MQ messages), and so on
- ***Support for*** disconnected use (change history), relationship integrity, generated and dynamic interfaces, validation, rich meta-data, XML and XML schema, XPath navigation, and so on
- Defines a special ***Service "type" – DataMediator*** – with custom implementation languages

SCA and SDO



- SCA is the component model
- Components may be wired together
- Business Objects are the data flowing on wires between Components

SCA and SDO Standardization

- Intent to standardize SCA and SDO
- A series of specifications published on Nov 30th, 2005:
  - SCA and SDO
  - For Java, C++, PHP
  - http://www.ibm.com/developerworks/library/specification/ws-sca/
  - Terminology and XML syntax adjustments and simplications
  - Additional support for Subsystems (application integration)
- Broad industry support - partners: BEA, IONA, Oracle, SAP, Siebel, Sybase
- Open source runtime and simple tooling to support the programming model in C++ and Java.
  - Apache Tuscany project
  - http://incubator.apache.org/projects/tuscany.html
  - Early SCA and SDO 2.0 runtime implementations

Major Standards Organizations

- **W3C (World Wide Web Consortium) – http://www.w3.org**
  - develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. Covers HTML, HTTP, XML, SOAP, etc. Founded in 1994 and includes 350 member organizations from around the world
- **OASIS (Organization for the Advancement of Structured Information Standards) - http://www.oasis-open.org**
  - drives the development and adoption of Web services, security, e-business, public sector and application-specific standards (BPEL, ebXML, UDDI, WSRP, WS-Security, etc.). Founded in 1993, has more than 3,000 participants from 600 organizations in 100 countries
- **JCP (Java Community Process) – http://www.jcp.org**
  - holds the responsibility for the development of Java technology. It primarily guides the development and approval of Java technical specifications (JSRs). JCP is controlled by Sun, but is open to everyone. IBM, BEA, Oracle, Novell, JBoss, AOL, ATG, SAS, Sybase, Borland, Macromedia, HP are among many other contributors
- **WS-I**
  - is an open industry effort to promote Web Services interoperability across platforms, applications, and programming languages. Provides guidance, recommended practices, and supporting resources for developing interoperable Web services (WS Basic Profile)

J2EE Application Servers

- **Commercial (major players)**
  - IBM - WebSphere Application Server
  - BEA – WebLogic
  - Oracle – AS
  - SAP – NetWeaver
  - Sun – Application Server
- **Open Source or Open Source based**
  - IBM – WebSphere Application Server Community Edition (WAS CE)
  - Apache – Geronimo
  - JBoss – JBoss (includes Tomcat)
  - Apache – Tomcat (servlet engine only)
  - and many more …

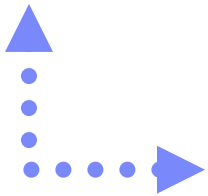
Trend: The Application Server Market is Maturing

Intense competition, emerging leaders:
•Consolidation of the market is occurring
•Clear market leaders such as IBM, BEA, Microsoft and Oracle have emerged
•SAP is growing

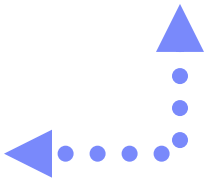
Maturation of standards:
Standards build confidence and encourage adoption of new technology
•(Gartner) Java 2 Enterprise Edition has emerged as the dominant standard and is now in its fourth release (J2EE 1.4).

Clones appear in the market:
•Low-end open source alternatives, such as JBoss and Red Hat App Server are addressing a market of commodity buyers
• **"Supported open source products provide a third choice** between 'freeware' and more robust products" (IDC)

Price Discounting:
•Intense competition is adding pressure to extend capabilities and make the application server highly consumable



Lighter & Heavier offerings: To contend with price pressure and open source competitors - leading app server vendors now offer light-weight versions at much lower price points, and ***application server platforms*** at higher price points to generate revenue.



The Middleware Game is Changing

"***Innovation in technology and in business models*** *continues to keep the enterprise application server market competitive and expanding, despite growing commoditization of basic J2EE application servers. New players continue to emerge to offer alternatives and to aim at the leaders' weaknesses."* – **Gartner Magic Quadrant for Enterprise Application Servers, 2Q05**

"After a period of testing by early adopters, ***firms are beginning to use open source software*** *…. in critical projects" –* **Forrester, Nov 2004**

"Forrester has found that availability of ***high-quality professional support*** *is the major barrier to widespread adoption of open source projects…. "* **Forrester, September 2004**

*"****New license revenue is no longer the singular measure of success*** *and influence in the Enterprise Application Server market."*
Gartner, 2005

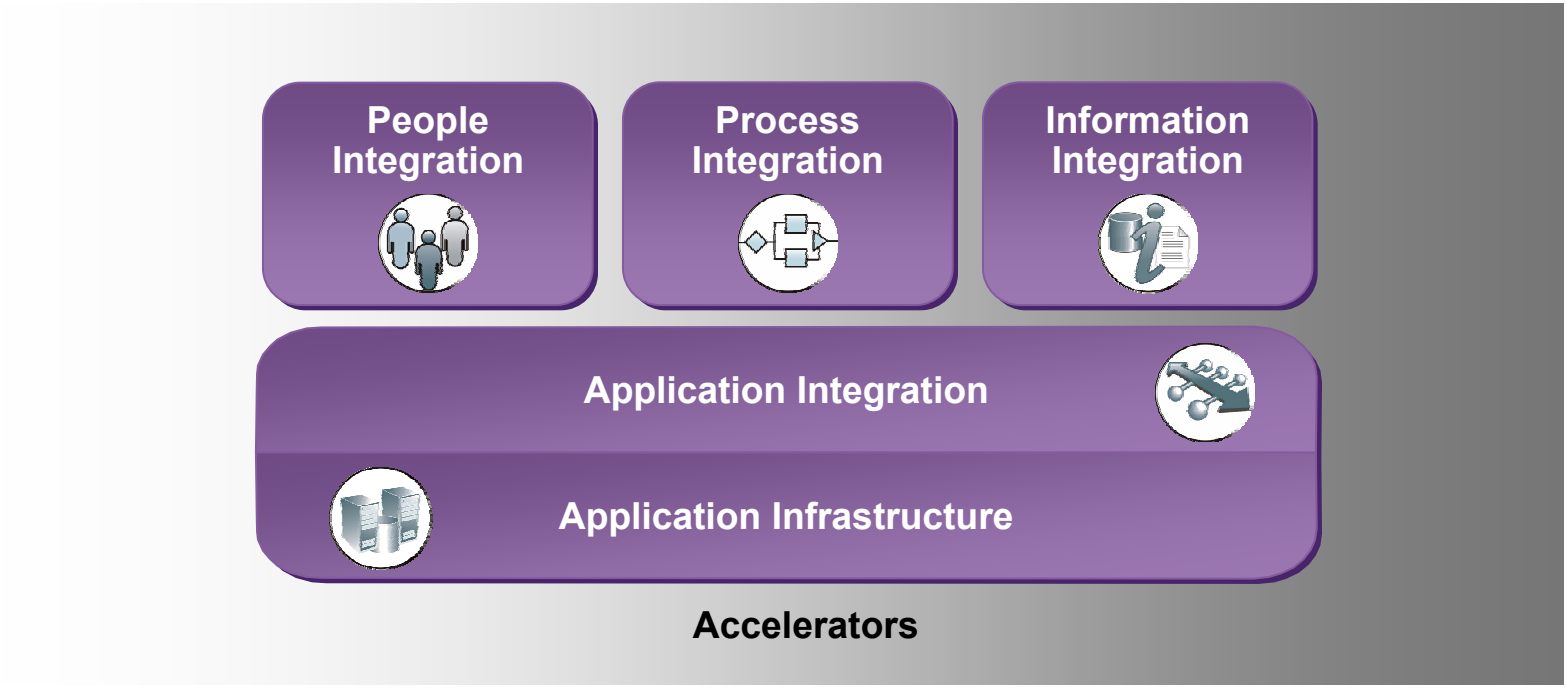
"Supported open source products *provide a third choice between 'freeware' and more robust products" –* **IDC**

Customers have begun to evaluate and deploy Supported Open Source Middleware for certain projects

Current Directions for the Application Server

- Commodity – everybody has one
- Infrastructure – operating environment for J2EE applications
- Support for Web Services, Service Oriented Architecture (SOA), Ease of Development, Deployment and Management
- Foundation for the Application Server Platform (APS)
  - People Integration
  - Process Integration
  - Information Integration
  - Application Integration
  - Application Infrastructure

SOA Middleware Enables On Demand Flexibility Through a Set of Integration and Infrastructure Capabilities



Integrate
people, processes and information

Optimize
application infrastructure

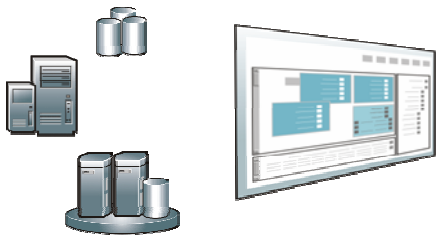
Extend
your reach

People Integration

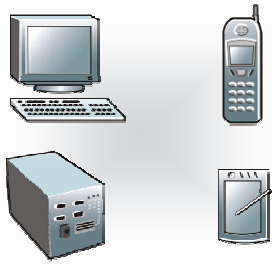
Interact with information, applications and business processes at any time from anywhere

Customer Challenges	Customer Benefits
▪ Systems and applications users need are not all integrated nor easy to use ▪ Mobile workers do not have access to information and applications they require in the field ▪ Customer service centers costs are high because time is spent on routine tasks, rather than value add inquiries	▪ Easy interaction with multiple processes and applications from a single access point ▪ Secure mobile access to business applications and information ▪ Automation of routine call center functions while improving customer experience and convenience

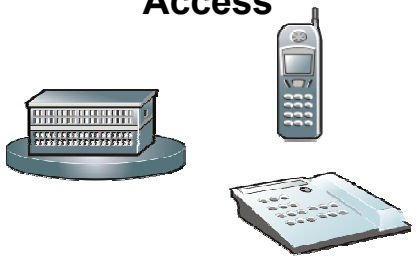
Enterprise Portal



Mobile Access



Voice\Conversational Access

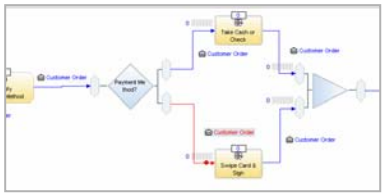


Process Integration

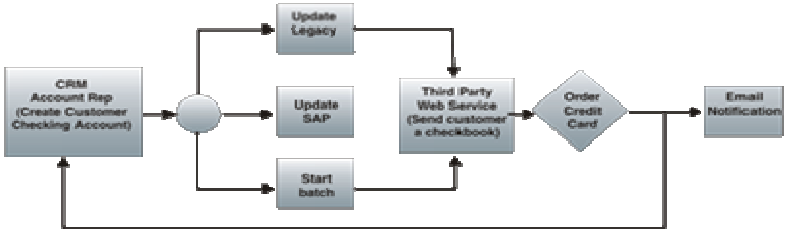
Optimize and integrate business processes to keep them in line with strategic goals

Customer Challenges	Customer Benefits
▪ Inability to streamline business processes, meet regulations, at low cost. ▪ Need to integrate people and applications in the business process ▪ Unable to monitor, control & continuously improve business operations	▪ Model, simulate and optimize business processes ▪ Choreograph process activities across the organization ▪ Monitor and manage process performance

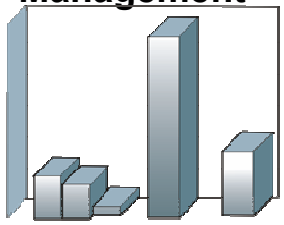
Process Modeling and Simulation



Process Automation



BAM & Process Management

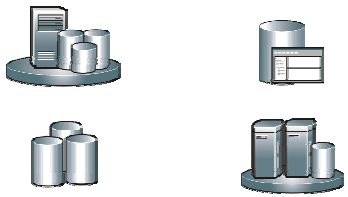


Information Integration

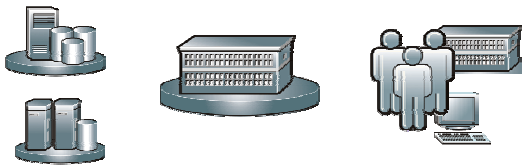
Access and manage information that is scattered throughout the enterprise and across the value chain

Customer Challenges	Customer Benefits
▪ Both structured and unstructured information are spread across one or more enterprises in a variety of databases, packaged applications, master files, mainframes, etc. ▪ Information gathering and review processes to coordinate multiple channels leveraging multiple customer touch points are lengthy ▪ Business processes to access and manage product information span departments and/or enterprises	▪ Manage and synchronize product reference information across the enterprise ▪ Centralize structured and unstructured information from disparate sources for easy access and use by users such as merchandisers ▪ Create a consistent, unified view of diverse data and content

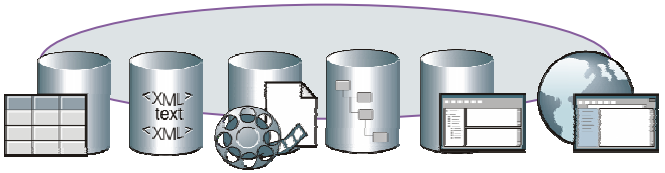
Global Data Synchronization



Multi-channel Commerce



Heterogeneous Information Integration



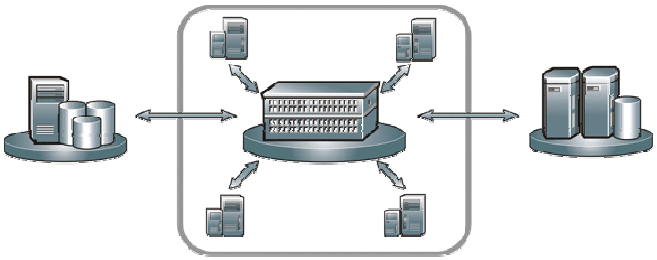
Application Integration

Assure reliable and flexible information flow between diverse applications and organizations

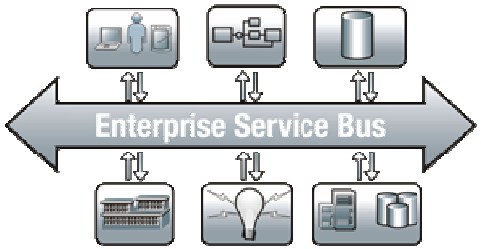
Customer Challenges	Customer Benefits
▪ Applications are not integrated in a flexible and reliable method across the enterprise, reducing business responsiveness ▪ Differences between many internal and partner applications must be managed ▪ Maintaining point to point or custom written integration interfaces is cost and time prohibitive	▪ Reliably and seamlessly exchange data between multiple applications ▪ Manage differences between multiple applications and business partners ▪ Adopt an enterprise wide, flexible, service oriented approach to integration

Application Connectivity

Application and Partner Mediation



Enterprise Integration Backbone



Application Infrastructure

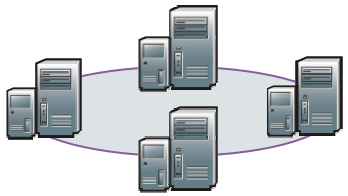
Build, deploy, integrate and enhance new and existing applications

Customer Challenges	Customer Benefits
▪ High turnover and training costs due to antiquated applications ▪ Unable to extend the business logic in legacy applications into new applications being developed ▪ Unable to meet customer and competitive demands on infrastructure performance, scalability, and manageability	▪ Quickly web-enable green-screen applications ▪ Adapt legacy applications for use in new java environments ▪ Deliver operational efficiency and enterprise Quality of Services (QoS) for a mixed-workload infrastructure

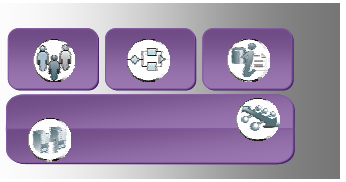
Modernizing the User Interface

Extending Legacy Applications into Web Infrastructure

Building a Robust, Scalable, Secure, Application Infrastructure



Accelerators

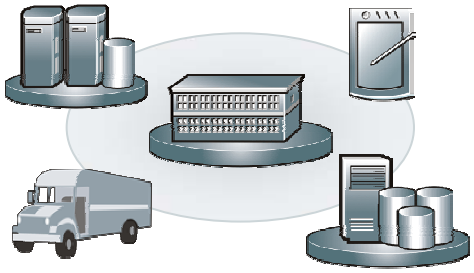


Pre-built capabilities and solution expertise to speed WebSphere implementations

Customer Challenges	Customer Benefits
▪ Lack of experience / expertise leading to greater project risk, time and cost ▪ Inefficient, disparate processes without re-usable components ▪ Rising development costs with each new business functionality request	▪ Pre-built capabilities reduce deployment time, effort and costs ▪ Proven technology, architecture and best practices to decrease project risk ▪ Buy vs. Build: out of the box capabilities save 7-10 times over customer built

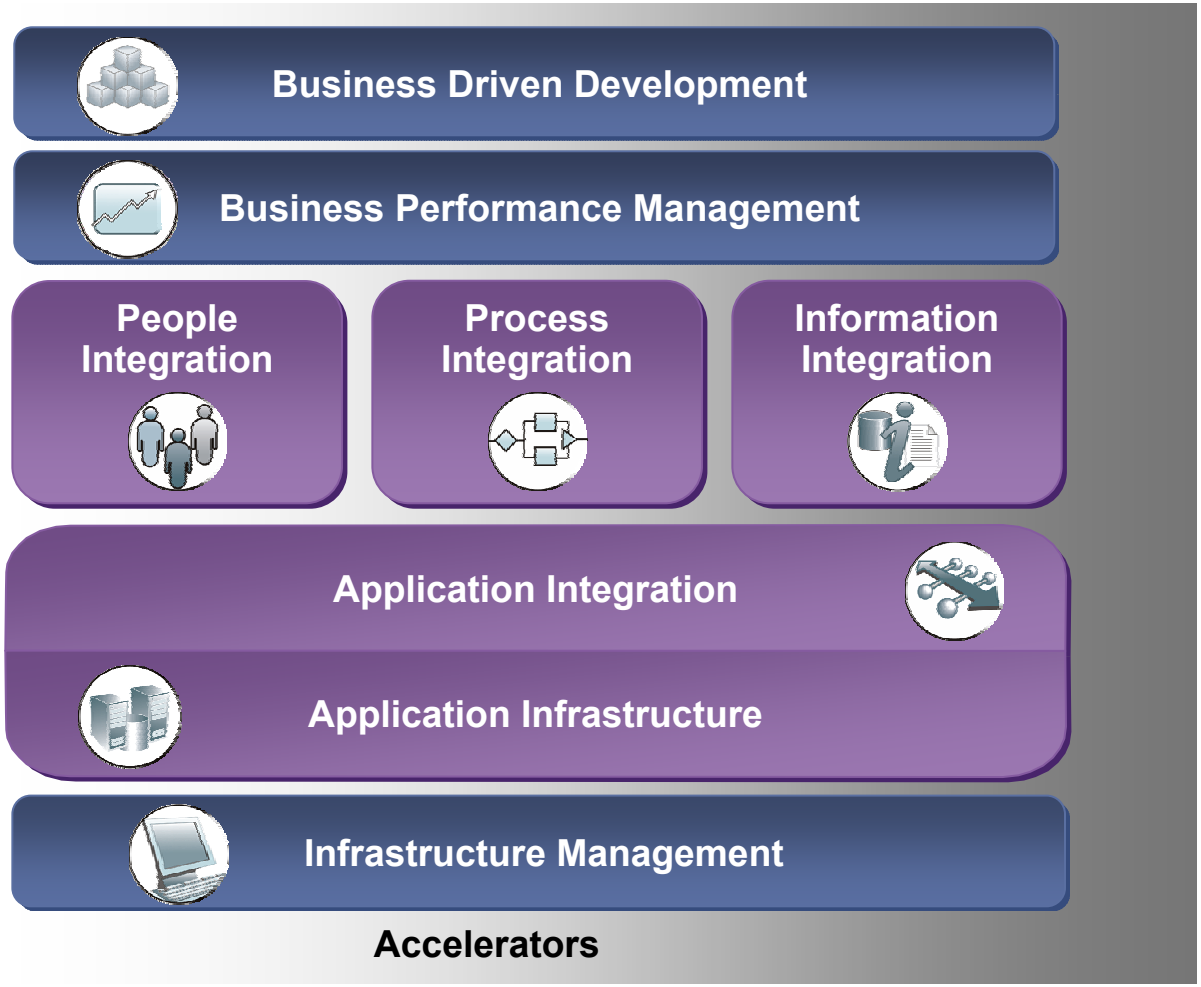
Pre-Built
Sell-Side Processes

Pre-Built
Supply Chain Integration



Pre-Built
Industry Specific Middleware

Robust Integration & Infrastructure Capabilities Connected in an Open, Flexible Manner



Modular product portfolio built on open standards

Functionally rich, adopted incrementally

Simple to develop, deploy and manage

Integrated role-based tools for development & administration

*…utilizing **common** install, administration, security and programming model*

Thank You