IBM Software Group

Modern Architectures for IMS Application Development

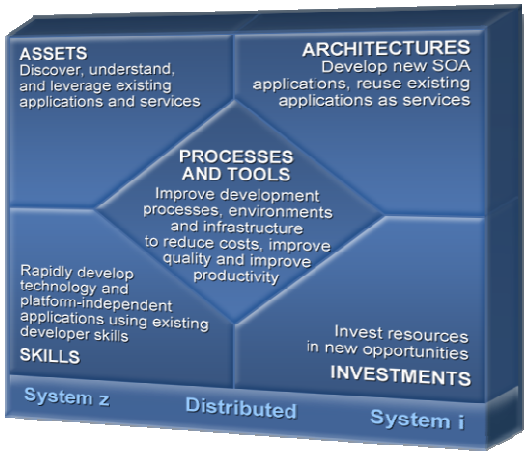
Rational. software

Agenda

- Modernization is SO IMS
  - Terminology
  - Classic and current layers
  - Classic and current tools
  - Applications can learn new tricks
- Tier by Tier – the Distributed Model
  - Client
  - Integration, app server, MVC
  - Servlet, JSP and JSF
- Rational Tools – Rapid Development & Deployment
  - Rational Application Developer for System Z
  - Rational Business Developer
- Connectivity - the IN Game
  - Web Services, XML, SOAP, WSDL, HTML
  - Callout, COBOL
- Summary

Aspects of Enterprise Modernization

To improve IT flexibility, you should modernize your enterprise in the following areas:



XML, SOAP, WEB Terminology

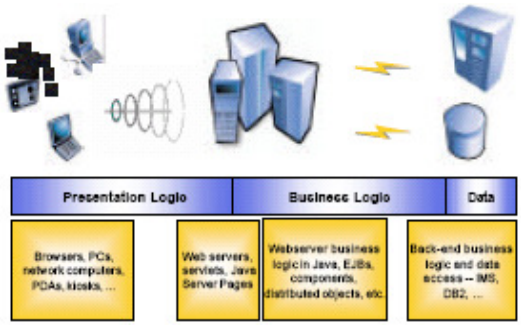
- **A tag / attribute based syntax**
- **Format of XML file described in**
  - **DTD** – Document Type Definition
  - **XSD** – XML Schema Definition
- **XML files are**
  - Well-formed (syntax is ok – matching tabs, etc.)
  - Valid (obeys rules in DTD or XSD)
- **SOAP and WSDL are based on XML**
  - SOAP - Simple Object Access Protocol - for exchanging XML-based messages over a network
  - WSDL - Web Services Description Language – XML document that specifies the location and operation the service exposes
- **Web Services** - allows an application to publish its function or message over a network (typically the internet). Uses XML to code and decode the data and SOAP to transport it.

J2EE Architecture

- **Java 2 Platform, Enterprise Edition**

A standards based architecture to enable development of multi-tier distributed applications

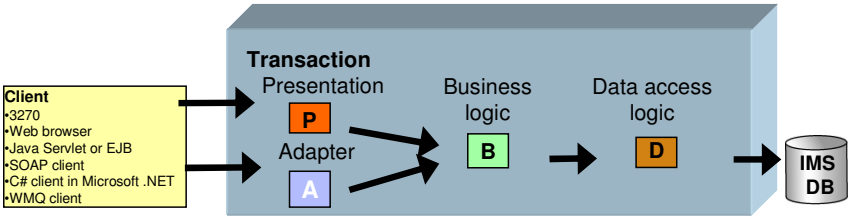
- **Recognizes long term dependency on existing legacy system**
  - "Enterprise Information Systems (EIS)"
- **The business logic in the middle tier runs as Enterprise Java Beans (EJB)**
  - EJBs can call other EJBs (anywhere) or EIS systems



J2EE Connector Architecture – J2C or (JCA)

- **An extension of J2EE architecture**
- **Defines a common set of interfaces that standardize interactions between:**
  - **Resource Adapters**
  - **Application Servers**
  - **Application Components**
- **CCI (Common Client Interface) is the standard API used by a Java application to interact with a resource adapter and run a transaction on EIS**
  - **e.g.. EJB access to an IMS Transaction**
- **No need to customize a connector for each application server, and vice versa**

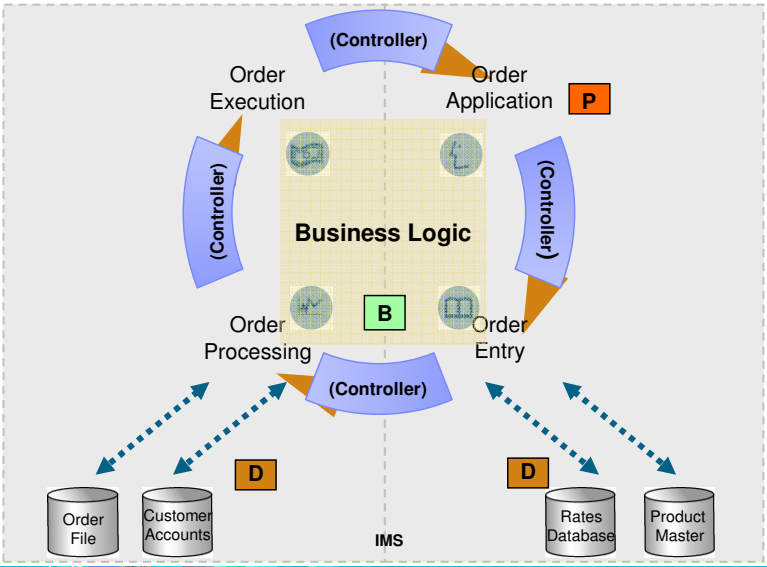
IMS / CICS Transaction Architecture – adaptable



- Best practice in IMS application design is to separate key elements of the application, in particular:
  - Client adapter or Presentation logic
  - Business logic
  - Data access logic
- Adapter (A) can be:

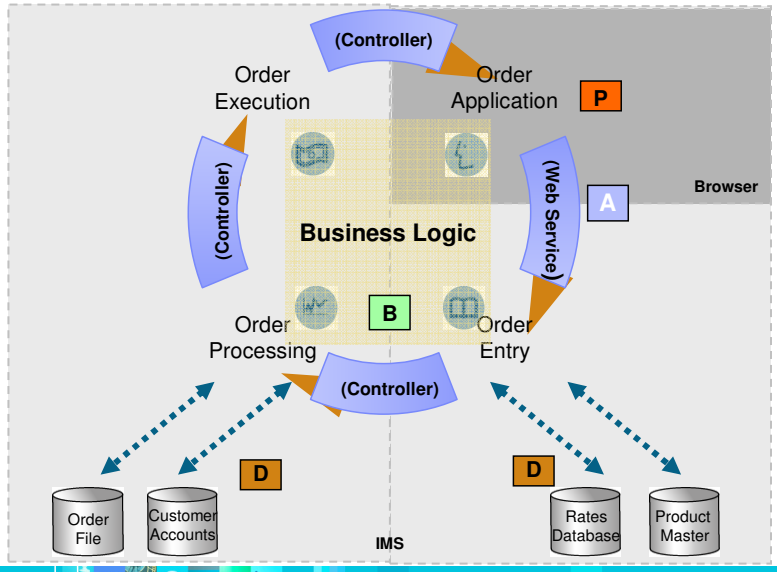
	IMS	CICS
▸	JCA (TM or DB) Resource Adaptors	JCA Resource Adaptors
▸	SOAP Gateway	JCA Transaction Gateway & SOAP
▸	Web Services (user or tool generated)	Web Services

Traditional Workload Application Components



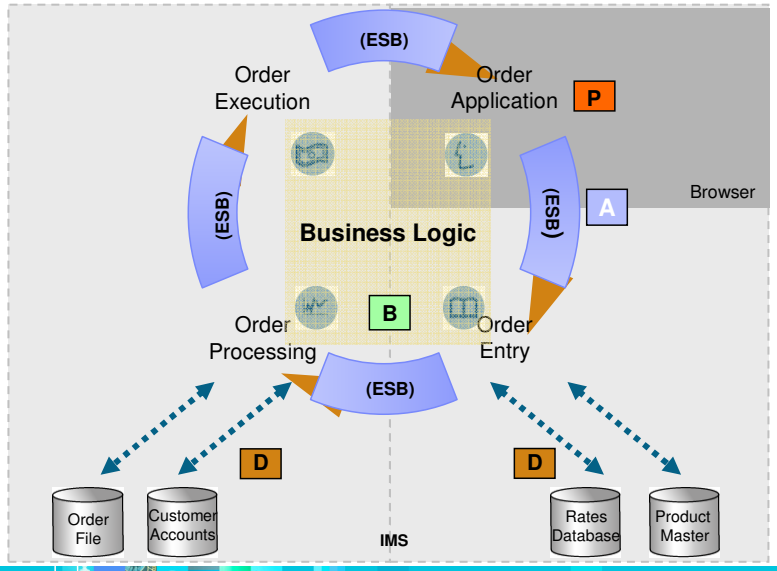
Workload Application Components - Composite

Spans multiple system and middleware boundaries

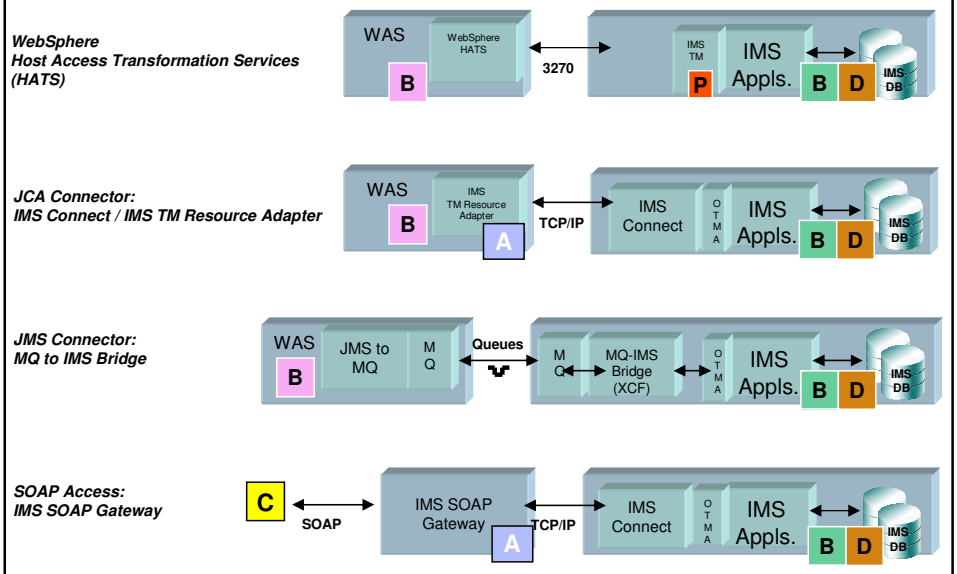


Workload Application Components - SOA

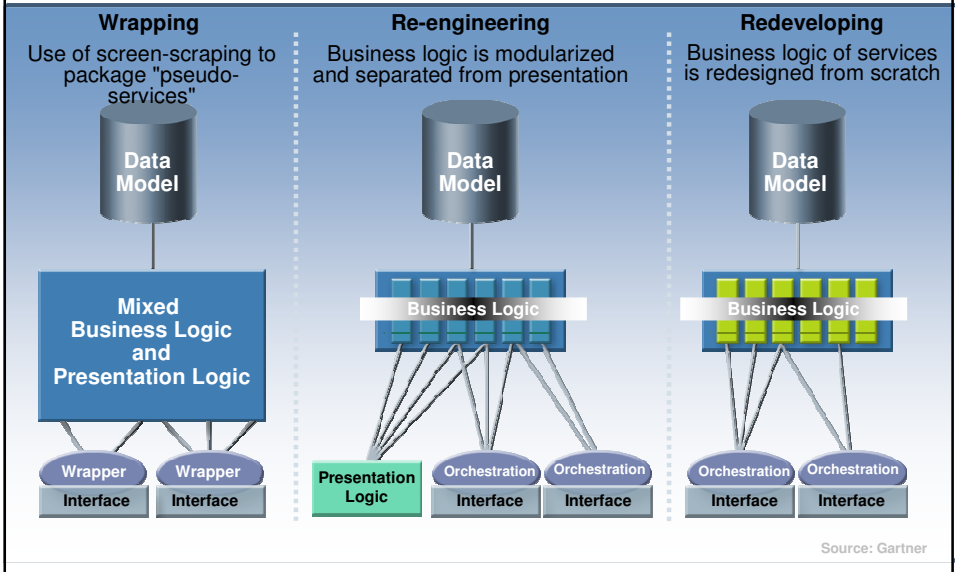
Spans multiple system and middleware boundaries – controller is the ESB



IMS Transactions - Connectivity Solutions



Three ways of re-using services from of existing applications



Source: Gartner

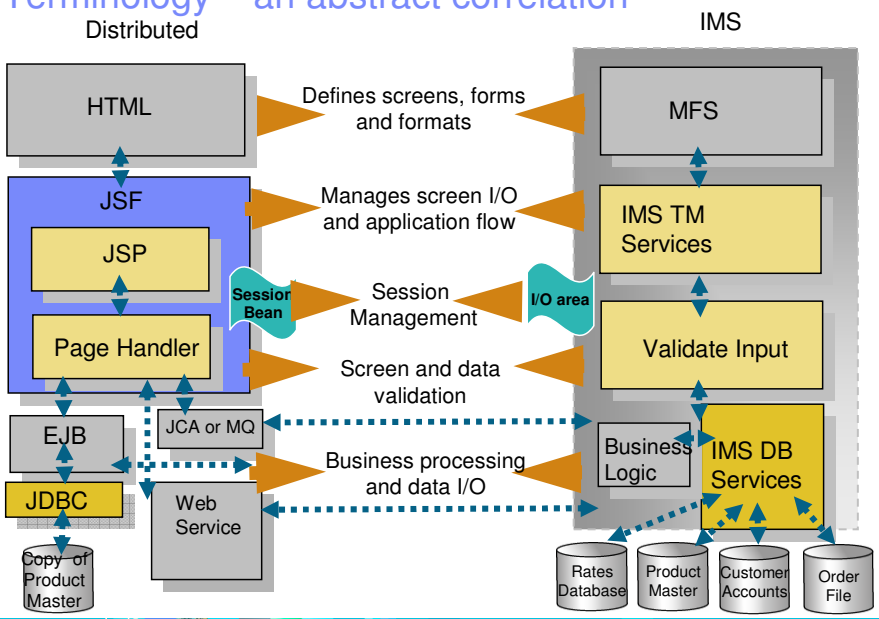
Tooling for Modern Application Development

Tools Used for Application Production

IDE	Mainframe Environment
IBM Rational Developer for System z **Host Tooling Integration** [JES, FA, FM, Debug Tool] **zOS Application Development** [COBOL, PL/I, C/C++, JCL, Screens, Stored Procedures, etc] **Enterprise Service Tools** [Web Services For CICS/IMS] **Host / Distributed SCM Integration** **IBM Rational Application Developer**	▸ TSO/E ▸ ISPF ▸ Screen Definition Facility II (SDF II) ▸ z/OS Problem Determination Tools ▸ SCLM

RDz was formerly WebSphere Developer for System z

Terminology – an abstract correlation

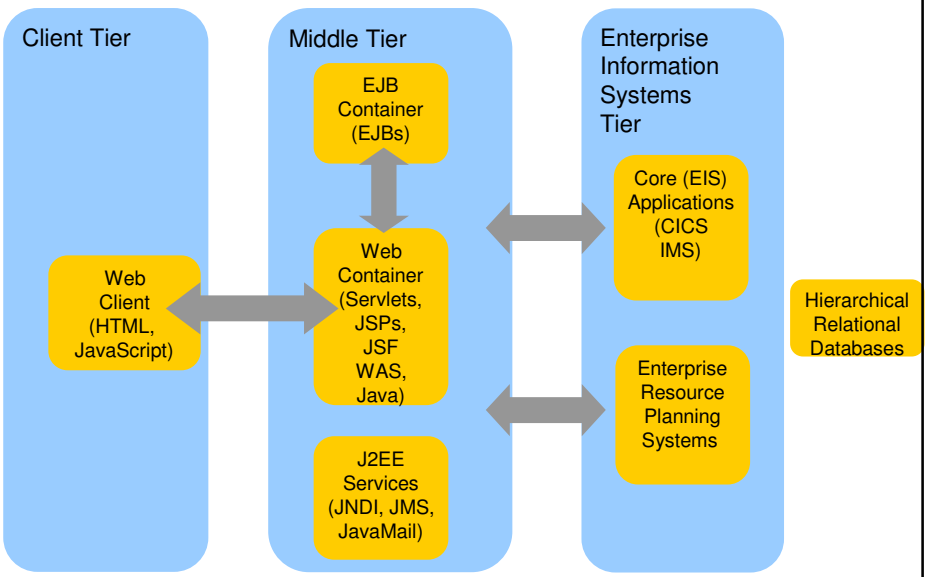


SOA - Procedural and object oriented approaches

- System requirement
  - ***Classic Bank Corp*** workload model expand into ***On Demand Bank Corp***
- Procedural approach
  - Identify where the data is stored
  - List the logical order necessary to perform the actions
- Object approach
  - Identify what objects are involved; these objects will directly relate to real life objects (Cost, Policies, Customer and Transaction)
  - Show how these objects interact:
- Either can be used to conduct B2C or B2B transactions

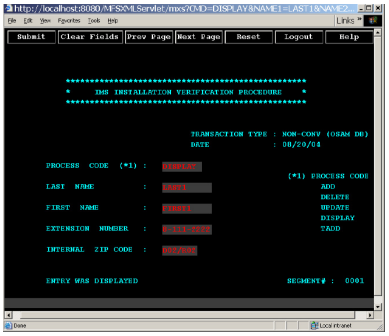
Both have advantages in SOA – in the right place

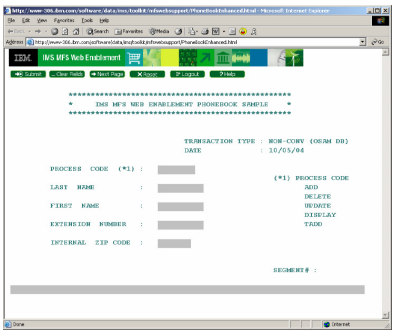
"Modern" Multitier Architecture



Client - HTML

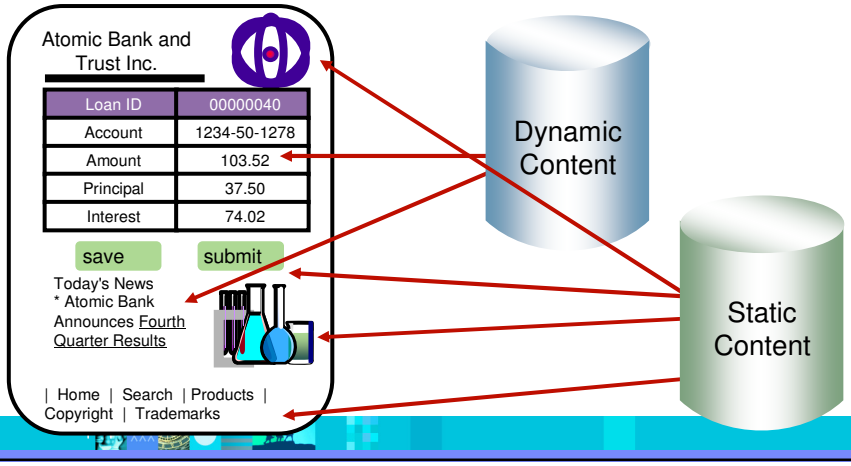
- HTML performs similar processing as BMS or MFS maps. It defines the screens and fields, colors, and interactions, although the technologies and implementations of course are different.
- Rational Developer for System Z has a Map Editor for MFS / BMS, combined with macros for MFS / BMS it generates the code that produces the traditional 3270 “green screen”.
  - RDz generates HTML for modern UI screens
  - HTML formatted page can resemble a 3270 if desired (keep 3270 look - not 3270 legacy)





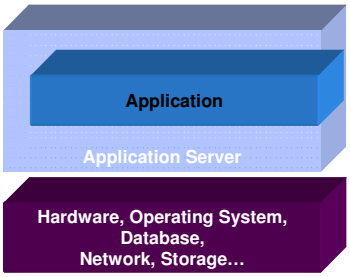
Web page content

- Content delivered to a client is composed from:
  - Static or non-customized content
  - Customized content
- Page layout and style are managed through HTML, XSL



What is J2EE?

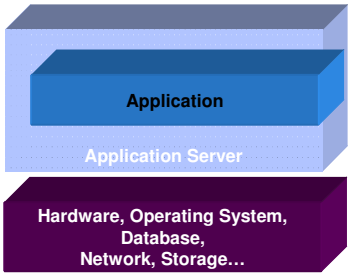
- J2EE – Java 2 Enterprise Edition
  - A run-time platform used for developing, deploying, and managing multitier server-centric applications on an enterprise-wide scale



- J2EE defines four types of components which must be supported by any J2EE product
  - Applets
    - Graphical Java components which typically execute within a browser
    - Can provide a powerful user interface for other J2EE components
  - Application client components
    - Java programs which execute on a client machine and access other J2EE components
  - Web components
    - Servlets and JavaServer Pages
    - These provide the controller and view functionality in J2EE
  - Enterprise JavaBeans
    - Distributed, transactional components for business logic and database access

What is an Application Server?

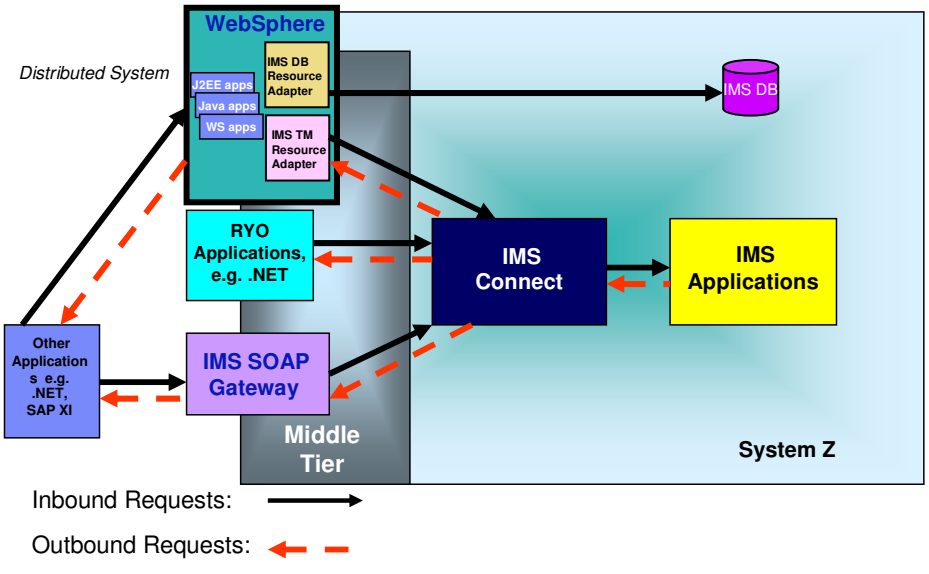
- Provides the infrastructure for running applications that run your business
  - **Insulates** applications from hardware, operating system, network…
  - Provides a common environment and programming model for applications
    - **Write once, run anywhere** (J2EE)
    - Platform for developing and deploying **Web Services**
  - Provides a **scalable, reliable** transaction engine for your enterprise



What is WebSphere Application Server?

- WebSphere Application Server is a platform on which you can run Java-based business applications
- It is an implementation of the Java 2 Enterprise Edition (J2EE) specification
- It provides services (database connectivity, threading, workload management, and so forth) that can be used by the business applications

IMS SOA Solutions – Service Provider and Service Consumer

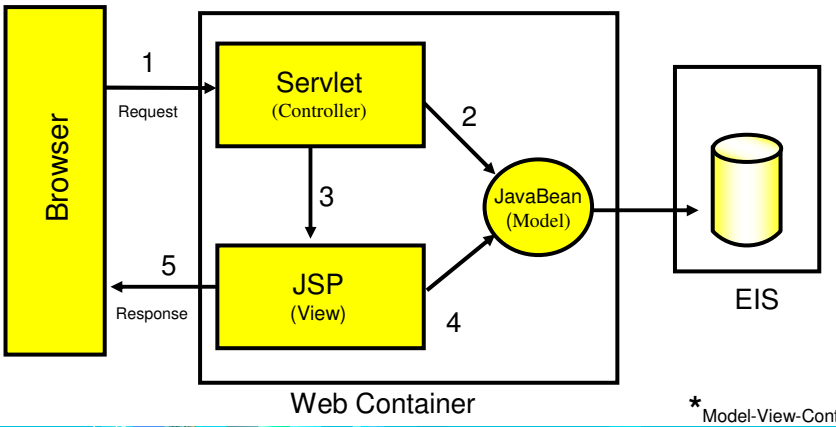


Typical J2EE Web Application Model

- A request is sent to a servlet that generates dynamic content and calls a JSP page to send the content to the browser, as shown:

MVC Design Pattern*

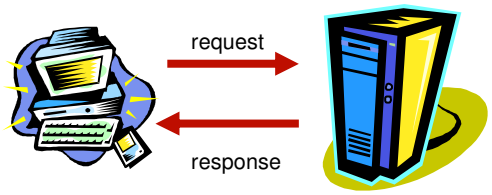
Combined Framework



*Model-View-Controller

What Is a Servlet?

- A servlet is a standard, server-side component of a J2EE application which executes controller logic on behalf of an HTTP request
  - Runs in the server tier (and not in the client)
  - A pure Java alternative to other technologies, such as CGI scripts
  - Managed by the Web container
- Servlets form the foundation for Web-based applications in J2EE



What is JSP (JavaServer Pages)?

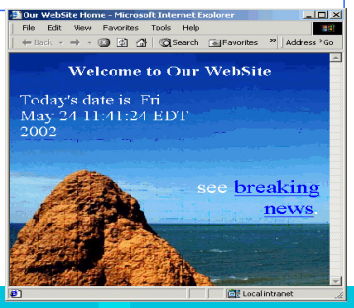
- JavaServer Pages is a technology that lets you mix static HTML with dynamically generated HTML. XML tags may also be included.
- JSP technology allows server-side scripting

```
<HTML>
<HEAD><TITLE>Our WebSite Home</TITLE></HEAD>
<BODY background="image.jpg" text="#ffffff">
<TABLE>
<TR><TD>
<H1>Welcome to Our WebSite</H1>
</TD></TR><TR><TD>
<H3>Today's date is

<%= new java.util.Date() %>

</H3></TD>
<TD>see <A href="breaking.html">
breaking news</A>.
</TD></TR>
</TABLE>
</BODY>
</HTML>
```

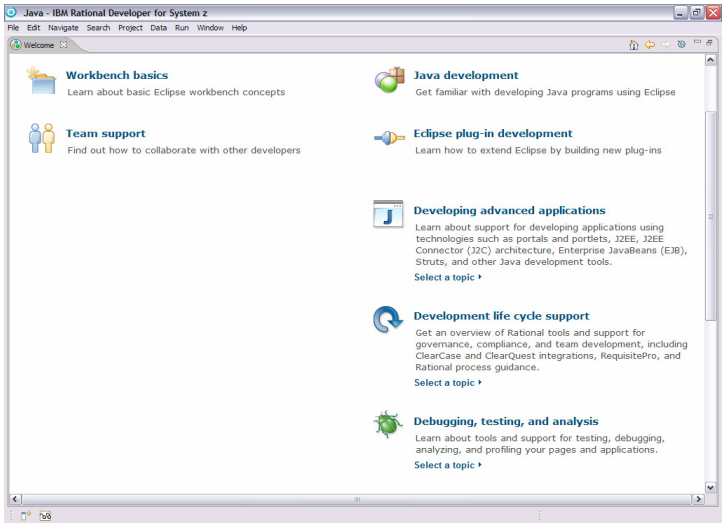
A simple JSP example



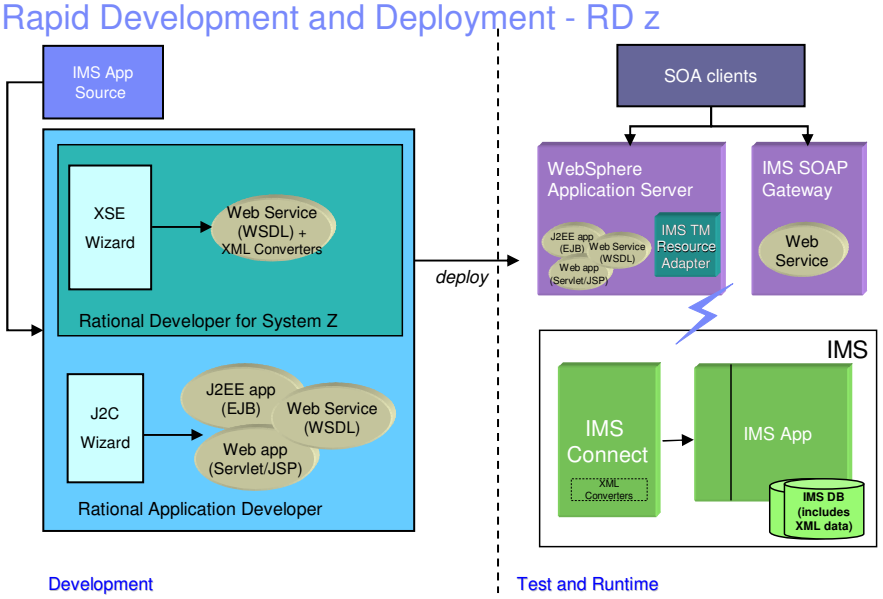
What is JavaServer Faces?

- JavaServer Faces (JSF) is a *framework* for developing Web-based applications.
  - A framework is a skeleton or foundation of an application
    - Provides code, resources, concepts and best practices upon which applications are constructed
- The main components of JavaServer Faces are:
  - An API and reference implementation for:
    - representing UI components and managing their state
    - handling events, server side validation, and data conversion
    - defining page navigation
    - supporting internationalization and accessibility
    - providing extensibility for all of these features
  - A JavaServer Pages (JSP) custom tag library for expressing UI components within a JSP page

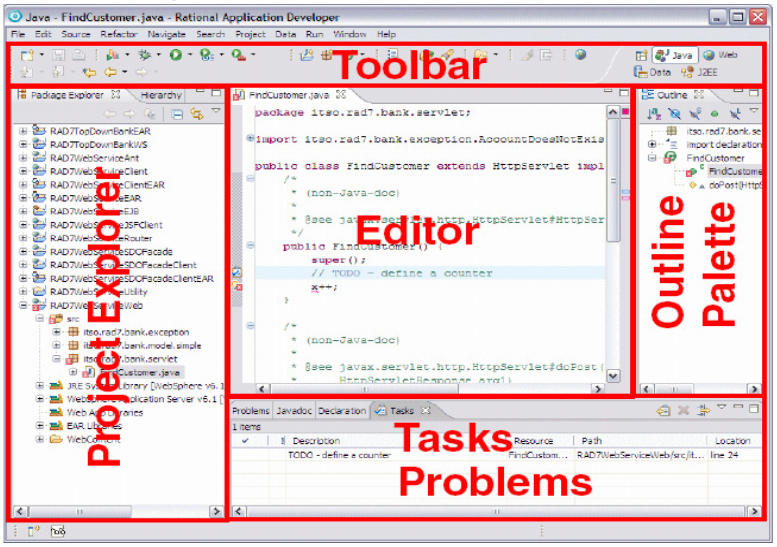
Rational Application Developer for System Z



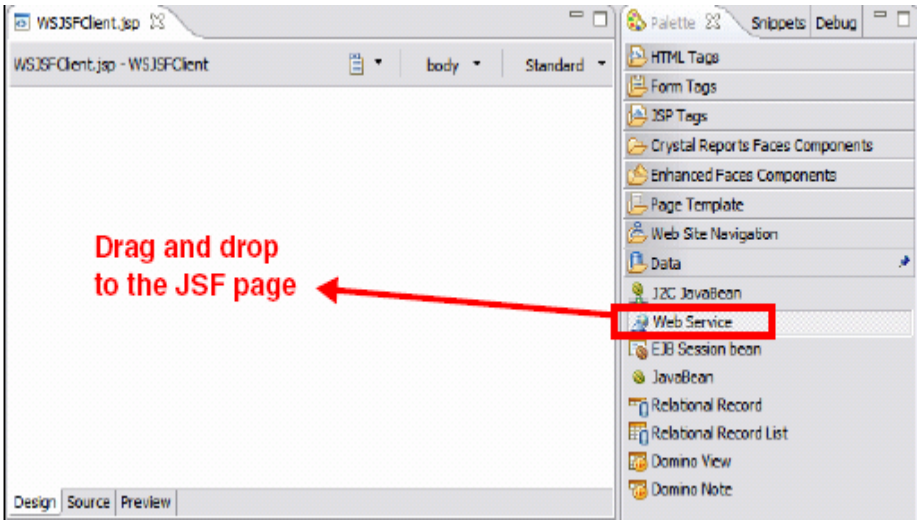
Rapid Development and Deployment - RD z



RDZ – sample view



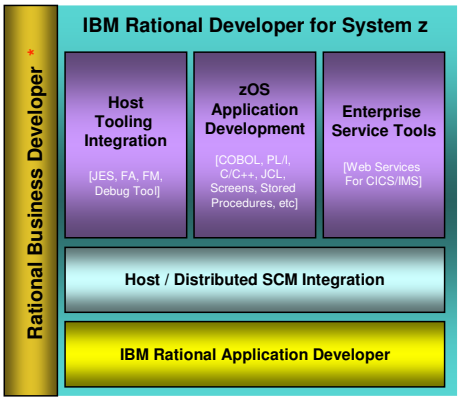
IDE drag and drop



IBM Rational Business Application Developer

Create modern application programs without knowing Java & J2EE

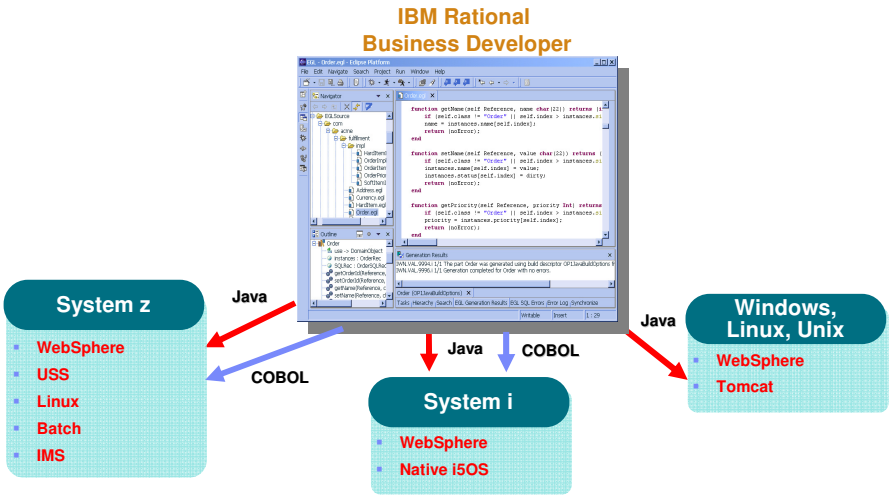
- EGL – Enterprise Generation Language – like Visual Basic for the mainframe.
- Develop true end-to-end Web-based applications without learning Java
- Uses an abstract development approach to build "business" developer skills to respond faster to business needs
- Easily integrate mainframe applications into service oriented architecture.
- Rapid development of new Services for all platforms including mainframe.



* Also available as a standalone product

Simplify and Accelerate cross-platform development

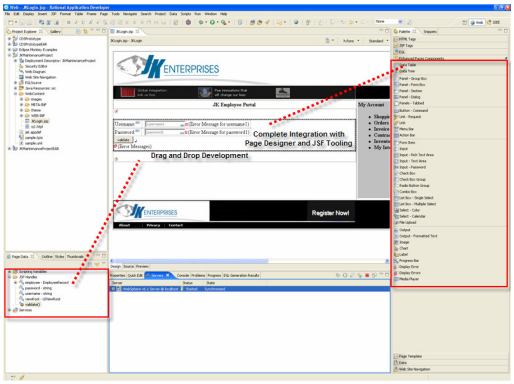
Build once, deploy anywhere



Platform Flexibility with IBM Rational Business Developer

Java Server Faces Integration

- First Class integration with Page Designer and JSF tools
  - Drop EGL data structures on JSP
    - Validation, editing, formatting rules from EGL Data Items applied
    - Appropriate UI controls rendered pre-bound to data declared in EGL Page
  - Server-side event handlers in EGL within context of page designer
- Integration is totally seamless
- No Java coding required to wire EGL data to JSF
- EGL logic can be used to handle user interaction with the JSP
- AJAX capability built in…partial refresh, etc…



EGL Web – IMS Programming Similarities

Page Data
~
MFS Map Fields

Load values from the database

"Send map"

"Receive map"

Process user-input values

Update Database

```
package pagehandlers;
import data.*;

PageHandler ordersbycustomer {view="ordersbycustomer.jsp", onPageLoad=onPageLoad}

//Page data - equivalent to I/O area for screen values
    customer Customer;
    dt   char(33);
    orders order[];
    sel int[] {selectFromList=Orders};  //Integer array - bind to Row Selection
    OrderRec Order; //Single Order record - for update of checked rows

    //vars for combo-box
    comboBoxSel char(12) {selectFromList=valueListArray,selectType=value}; //Display
    valueListArray  char(12)[];     //Temp holding array for state values
    j int ;             //Loop ctr - max number of rows in Customers dynamic array
    i int ;
    s int;

    Function onPageLoad(cid  int) //Receives control upon entry
    dt=sysvar.currentFormattedDate;
    customer.CUSTOMER_ID=cid;
    CustomerLib.getCustomer(Customer); //Load customer data from the database
    s = sqlcode;
    OrderLib.ordersByCustomer(cid, Orders); //Load order data from the database

  end

    Function updateOrders() //Receives control upon button-clicked event
        arrayMax int;   //sel (array) is created to the size of # of checked rows
        arrayMax = size(sel);   //Get this size (= # of checked rows)
        i int;  //Array loop ctr
        i = 1;  //Initialize loop ctr
        j int;  //Declare temp variable to hold indexed value in sel
        while (i <= arrayMax)   //Loop through all checked rows in Sel array
            j = sel[i];     //assign each sel[i] value to temp var.
            move  Orders[j] to OrderRec byname; //Move the fields
            orderrec.ORDER_STATUS = comboBoxSel;
            OrderLib.updateOrder(OrderRec); //Update the DB
            i = i+ 1;       //Don't forget to increment the array loop ctr
        end
```

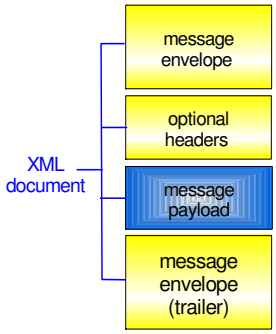

XML – Basic Parts

```
<?xml version="1.0" standalone="no" encoding="UTF-8" ?>
<!DOCTYPE shirt SYSTEM "http://shirts.com/xml/dtds/shirt.dtd">
<shirt>
   <model>IMS Tee</model>
   <brand>Tommy Hilltop</brand>
   <price currency="USD">10.95</price>
   <fabric content="70%">cotton</fabric>
   <fabric content="30%">polyester</fabric>
   <on_sale/>
   <options>
      <colorOptions>
         <color>red</color>
         <color>white</color>
      </colorOptions>
      <sizeOptions>
         <!-- Medium and large are out of stock -->
         <size>small</size>
         <size>x-large</size>
      </sizeOptions>
   </options>
   <order_info>Call &phone;</order_info>
</shirt>
```

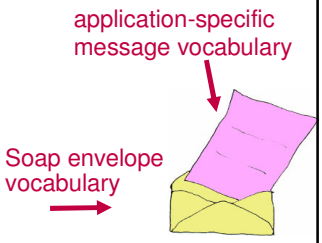
XML Declaration
Document type declaration
root element
child of root
end tag
start tag
attribute
attribute
empty element
comment
entity reference

Simple Object Access Protocol (SOAP)

- **An XML-based protocol for exchanging 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**



- *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!

SOAP: Request Message

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV=
    "http://www.w3.org/2001/06/soap-envelope"
      SOAP-ENV:encodingStyle=
    "http://www.w3.org/2001/06/soap-encoding">

   <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>IBM</symbol>
        </m:GetLastTradePrice>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

app-specific message

SOAP envelope

SOAP: Response Message

Result returned in Body

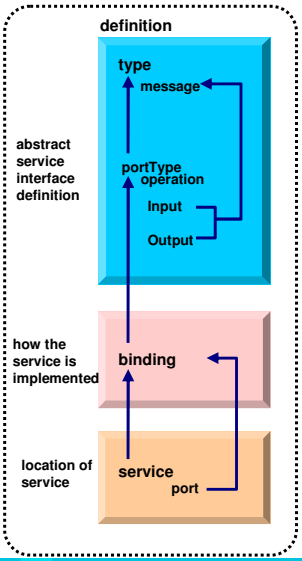
```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV=
     "http://www.w3.org/2001/06/soap-envelope"
      SOAP-ENV:encodingStyle=
     "http://www.w3.org/2001/06/soap-encoding">

   <SOAP-ENV:Body>
      <m:GetLastTradePriceResponse
         xmlns:m="Some-URI">
          <Price>134</Price>
      </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

app-specific message

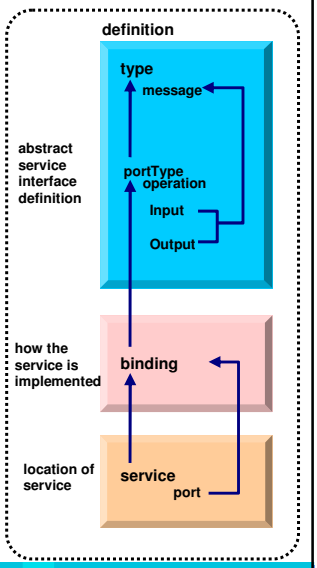
WSDL - Web Service Description Language

- **Open Standard**
- **XML model 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**
- **Can be One or multiple documents**
- **Major sections are:**
  - Service Interface (operations, input, output)
  - Service binding (protocol binding)
  - Service implementation (location of service)



WSDL: Logical Contents

- Service Interface
  - Operation (business functions)
    - Input Message (0 or 1) and 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 service interface implementation
- Service Implementation
  - Location of the service

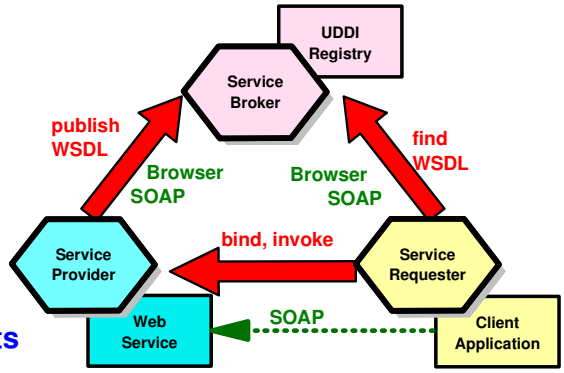


WSDL: Physical Contents

- Definitions – highest level tag
  - **types** – definition of complex parts
  - **message** – a grouping of 1 or more parts
    - **parts** – simple or complex (complex points to a type)
  - **portType** – a grouping of operations
    - **operation** – correspond to business functions
      - **input** – points to input message
      - **output** – points to output message
      - **fault** – can be returned when stuff goes wrong
  - **binding** – physical associations to operations
    - **operation** – implementation of a portType operation
  - **service** – grouping of ports
    - **port** – location of associated binding

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
- WS-I.org (Web Services Interoperability)

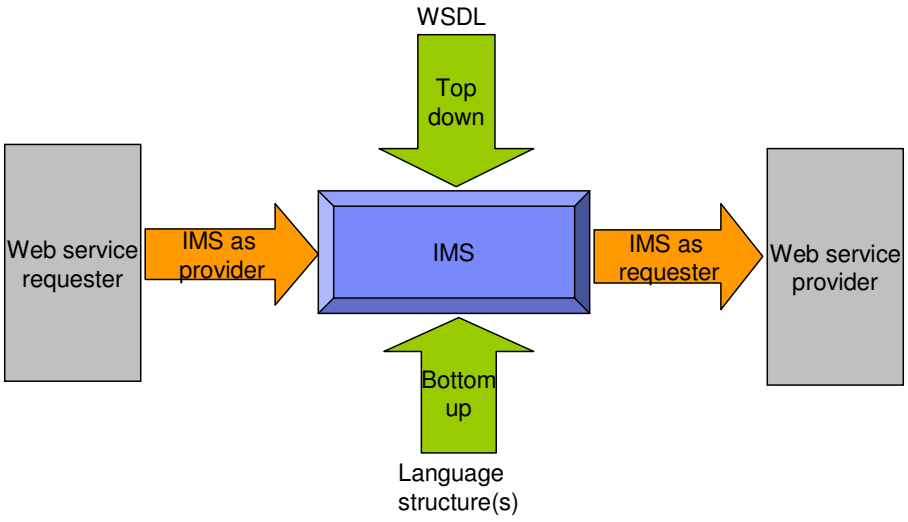


WS-I WEB SERVICES INTEROPERABILITY ORGANIZATION

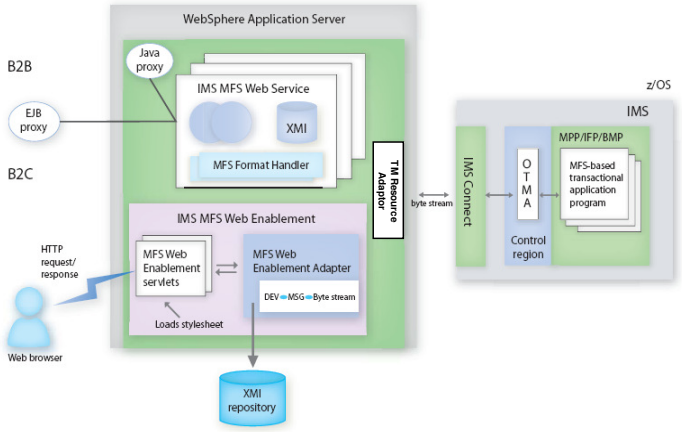
uddi.org

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

Web Services Enablement Styles



MFS Web Services and Web Enablement



MFS and HTML

MFS

Name and overall format of map - Includes items such as input/output, whether keyboard should be enabled, types of terminal, colors, size etc. are defined.

```
SAMPIMS   FMT
DEV   TYPE=3270-A02,FEAT=IGNORE,SYSMSG=SYSMSGA,DSCA=X'00E0'
      DIV   TYPE=INOUT
```

Headings and text fields. Defined with DFLD statement. You see position, length, initial value, and field attribute below.

```
EMPNO    DFLD  'Please type Employee Number
'POS=(3,1),LTH=27,ATTR=PROT
```

Input Fields. Defined with DFLD statement. You see a name (which ultimately defines storage size (and Cobol copybook field definition), and a difference with the field defined as unprotected – information can be entered.

```
EMPINUM DFLD  POS=(3,29),LTH=6,ATTR=(HI,MOD)
```

HTML

Headings – Overall definition, including whether Java Server faces tags will be used, a heading, and stylesheet definition.

```
<HEAD>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ page language="java" contentType="text/html; charset=CP1252"
    pageEncoding="CP1252"%>
<META http-equiv="Content-Type" content="text/html; charset=CP1252">
<META name="GENERATOR" content="IBM Software Development Platform">
<META http-equiv="Content-Style-Type" content="text/css">
<LINK href="theme/Master.css" rel="stylesheet" type="text/css">
<TITLE>MAP1</TITLE>
```

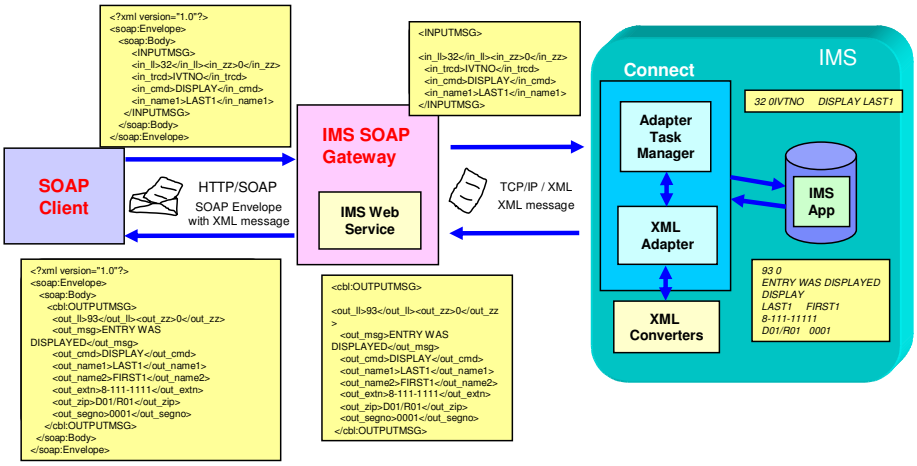
Text headings including location definition, colors, attributes, etc.

```
f:view> <BODY>
<h:scriptCollector id="scriptCollector1"><h:form styleClass="form" dir="ltr"
    id="form1"><table><tr><td colspan="20"> </td>
<td colspan="22" nowrap><font color="#ffff00">Employee Record Viewer</font></td>
<td> </td>
<td nowrap><font color="#0000ff"></font></td>
<td colspan="36"> </td>
<tr><td colspan="80"> </td>
<tr><td> </td>
<td colspan="27" nowrap><font color="#00ffff">Please type Employee Number</font></td>
```

Input fields

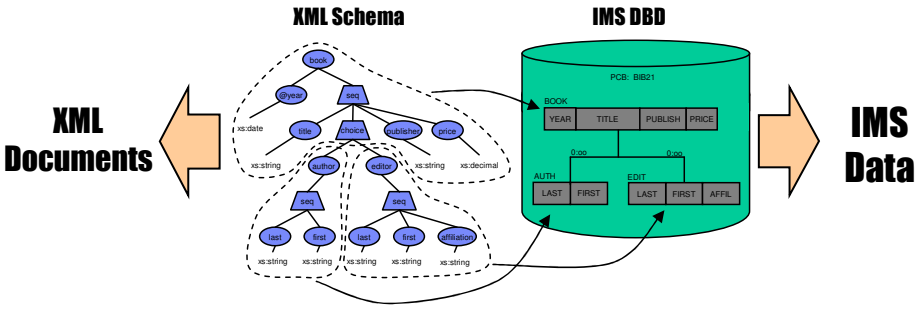
```
<td> </td>
<td colspan="6" nowrap><h:inputText value="#{pc_APPFMTPage.APPMSGBean.empnum}"
    required="false" style="color: #00ff00" size="6" id="empnum"></h:inputText></td>
<td> </td>
<td colspan="44"> </td>
<tr><td colspan="80"> </td>
<tr><td> </td>
```

SOAP Client Invokes IMS COBOL Application as a Web Service

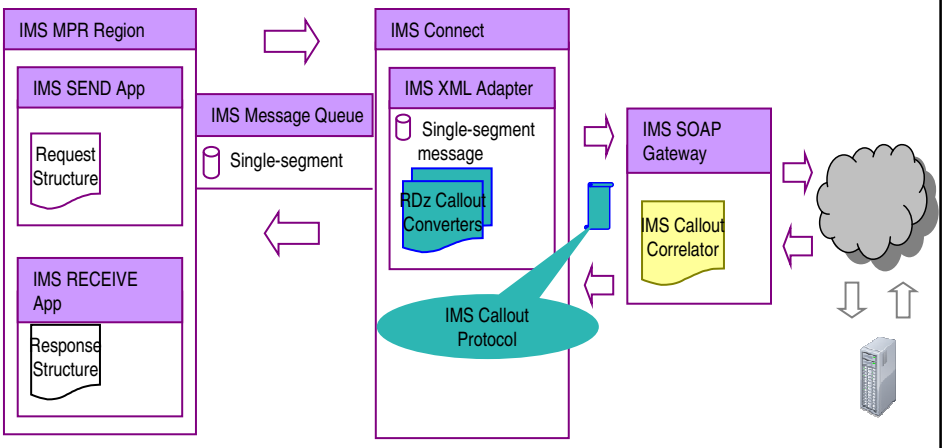


IMS XML Database

- Introduces a way to view/map *native* IMS hierarchical data to XML documents
- Aligns IMS Database (DBD) with XML Schema
- Allows the retrieval and storage of IMS Records as XML documents with ***no change*** to existing IMS databases
- Enables query of IMS data using XQuery



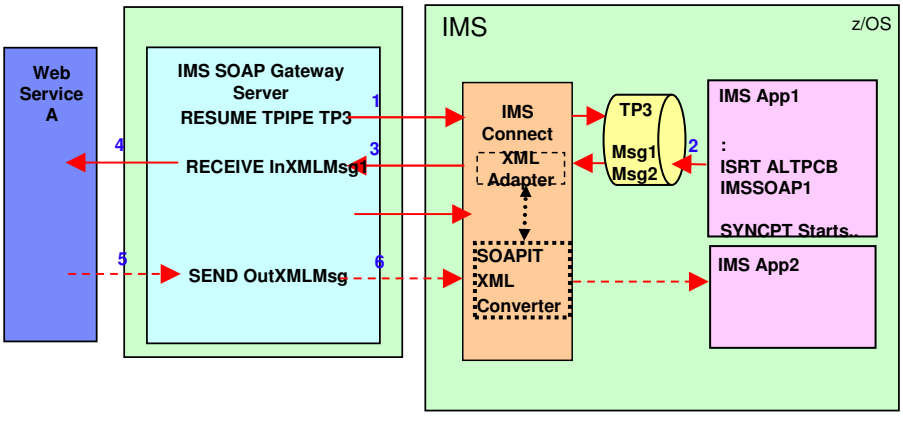
IMS invokes a Web service



IMS V10 Asynchronous Callout

IMS application receives data from a web service

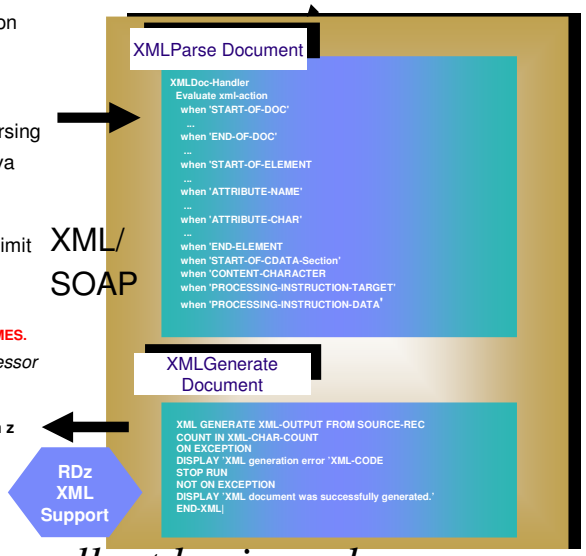
- Enables IMS application to act as a client to asynchronously invoke Java EE applications and Web Services
- Receiving output from external application is possible



IBM Enterprise COBOL

CICS/IMS/Batch/DB2 COBOL

- XML Language based generation from COBOL data structure
  - XMLGenerate Verb
  - WebSphere EJB support
- High speed XML Sax based parsing
- Object Oriented Support for Java COBOL Interoperability
- Unicode support
- Raise 16Mb COBOL data size limit
  - Picture clause replication:
    01 A PIC X(134217727).
  - OCCURS::
    05 V PIC X OCCURS 134217727 TIMES.
- *CICS and DB2 integrated preprocessor*
- Tool integration
  - **Rational Developer for System z**
  - **File Manager, Fault Analyzer**
  - **Rational Asset Analyzer**



XML/ SOAP

RDz XML Support

COBOL is an excellent business language

Why COBOL?

- Large portfolios
- Many developers
- High performance
- Self documenting
- Proven Maintainability
- Business oriented, eases technology burden

Summary

- MVC application model provides high levels of flexibility
- IMS provides leading edge support of Web Services
  - Allows for re-use of existing business assets and new development of high QOS assets
- Developers need “complete” application skills
- IMS and WebSphere Application Server are strategic middleware products that together…
  - Interoperate - Web services, JCA, Enterprise JavaBeans
  - Exploit and complement z/OS qualities of service
  - Provide the capabilities and SOA advantages of maximizing your application assets and save time and money to grow or change those assets when needed.

Demos

Modern Application Architecture Demos

- Rational Developer for System Z used to create a Web enabled IMS transaction from the laptop.
- RDz used to show how a COBOL application is developed, compiled, and run. Debugging will also be demonstrated.
- DLIModel utility used to transform PSB, DBD and COBOL copybook data into application independent metadata.

Additional Documentation

- IMS
  - **An Introduction to IMS: Your Complete Guide to IBM's Information Management System** (Book from IBM Press)
  - "Home Page"
    - http://www.ibm.com/software/ims
  - Library
    - http://www-306.ibm.com/software/sw-library/en_US/products/V364612O24358O76/Web Services Guide
  - **IMS Connectivity in an On Demand Environment:**

    A Practical Guide to IMS Connectivity(redbook) SG24-6794-00
- **IMS Version 10 Implementation Guide A Technical Overview**

  (redbook) SG24-7526-00

Resources (1 of 3)

- Web Services Architecture (@ W3C)
  - http://www.w3.org/TR/ws-arch/
- Web Services Zone (@ IBM developerWorks)
  - http://www.ibm.com/developerworks/webservices/
- Websphere V5 Web Services Handbook
  - Redbook: SG24-6891
- Rational Application Developer V7 Programming Guide
  - Redbook: SG24-7501-00
- IMS Info Center

Resource (2 of 3)

- SOAP 1.1 Specification
  - http://www.w3.org/TR/SOAP/
- Apache SOAP4J: xml.apache.org
  - SOAP4J version 2.2, stable, ready for use
  - AXIS (First release available)
- W3 standardization: w3.org/2000/xp
  - SOAP 1.2 specification
  - XML Protocol working group requirements and charter
- SOAP - WebServices Resource Center
  - http://www.soap-wrc.com/webservices/default.asp
  - MANY resources - e.g., link to SOAP::Lite for Perl
- Xmethods lists publicly-accessible web services
  - http://www.xmethods.net

Resources (3 of 3)

- WSDL 1.1 Specification
  - http://w3.org/TR/wsdl
- WSDL4J
  - http://oss.software.ibm.com/developerworks/projects/wsdl4j
- WSDL Toolkit (part of WSTK)
  - http://ibm.com/alphaworks (look under xml on left)
- Rational Developer for System z
  - http://ibm.com/software/awdtools/devzseries
- WSDK (WebSphere SDK for Web Services):
  - http://ibm.com/developerworks/webservices/wsdk/
- Articles and tutorials:
  - http://ibm.com/developerworks/webservices
- COBOL wiki

Modern Application Architecture Has Concluded