Software Group

Modern Application Architecture

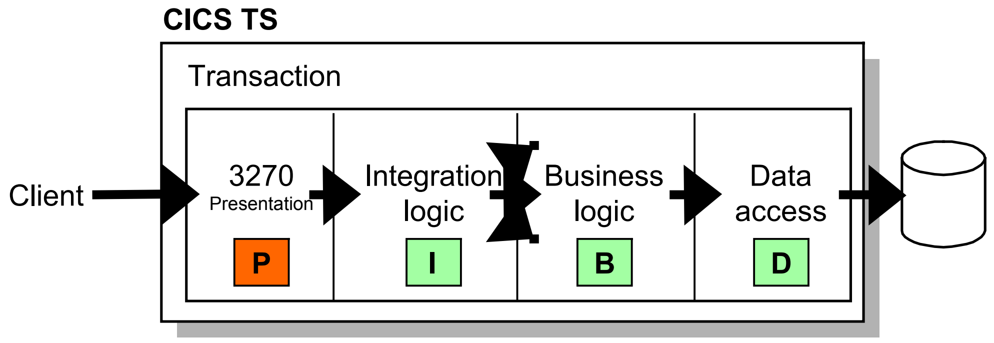
SOA : Part 2

Paolo Chieregatti
Certified IT Specialist
paolo.chieregatti@it.ibm.com

Agenda

- **Introduction**
- **Middle Tier**
  - WebSphere Application Server
    - J2EE
    - Servlets, JSP’s and JSF
    - EGL
- **Client**
  - HTML
- **Connectivity**
  - Web Services, XML, SOAP, WSDL
- **Business Tier**
  - CICS
  - COBOL

Modern “CICS” architecture

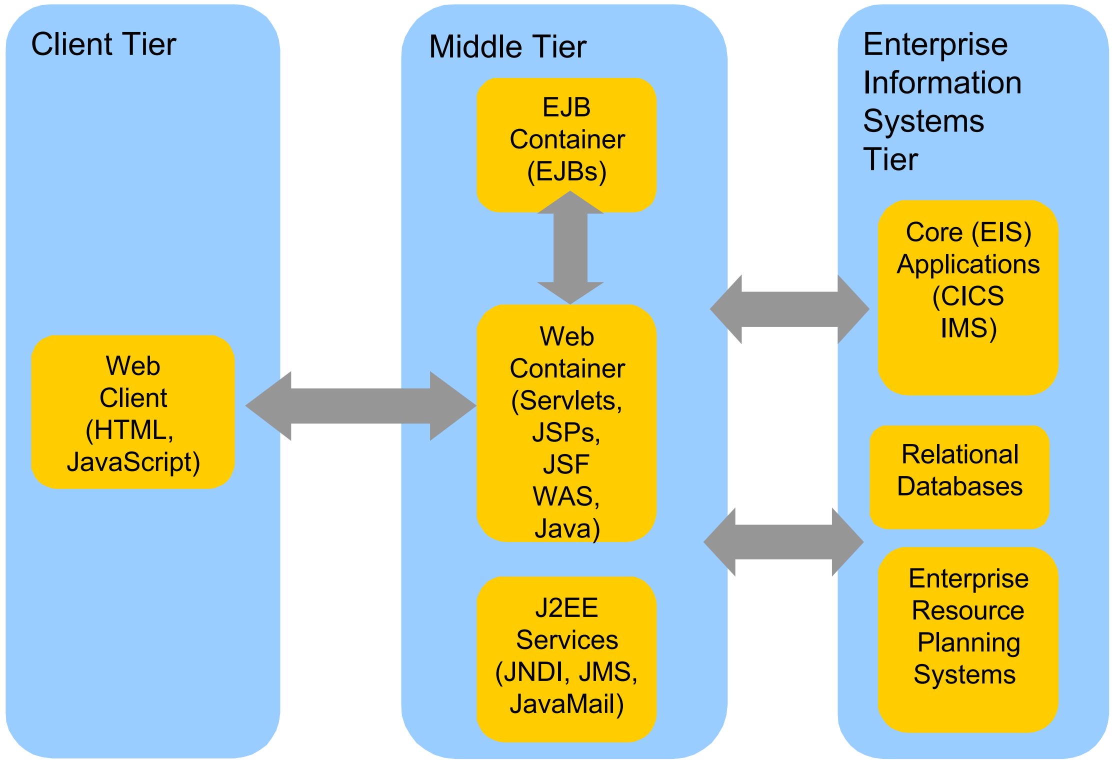


- **Best practice in CICS application design is to separate key elements of the application, in particular:**

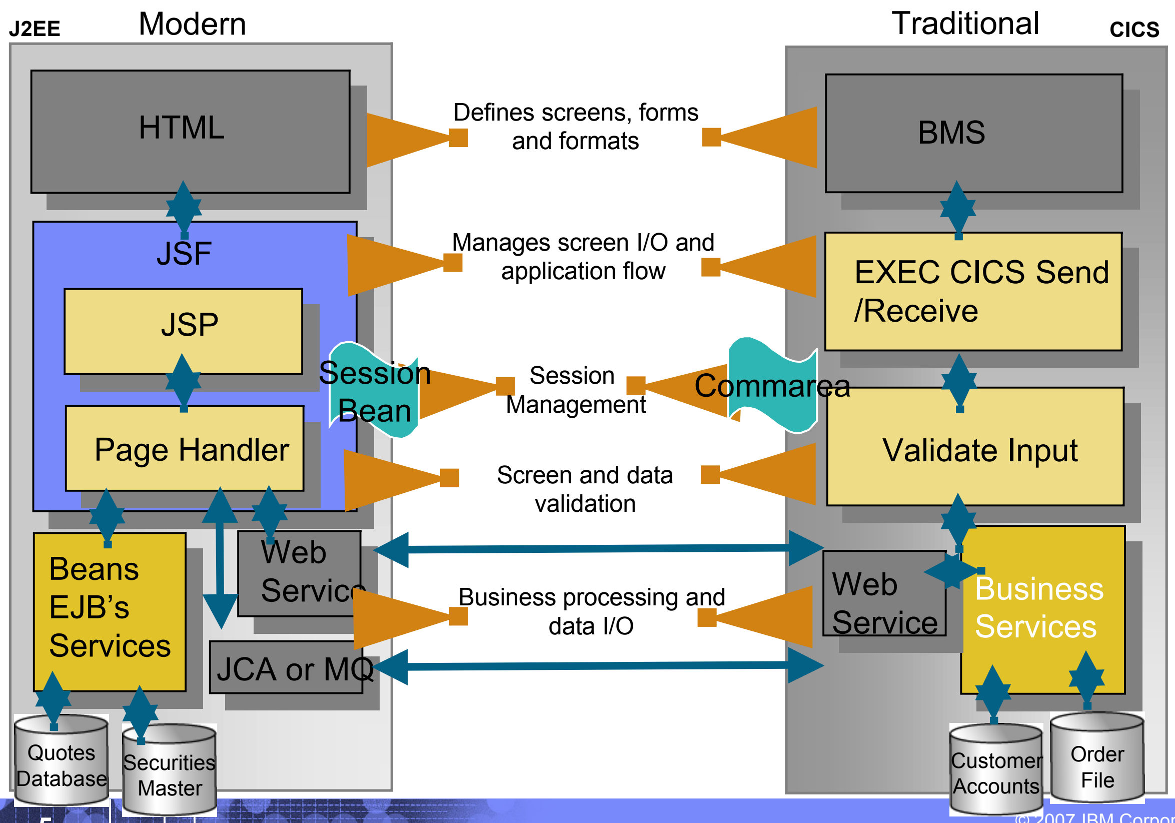
– Presentation logic	3270, HTML, XML
– Integration or aggregation logic	Menu, router, tooling
– Business logic	COBOL, PL/I, Reusable component
– Data access logic	VSAM, DB2, IMS, …

- **Provides a framework for reuse and facilitates separation of concerns, clear interfaces, ownership, and optimisation**

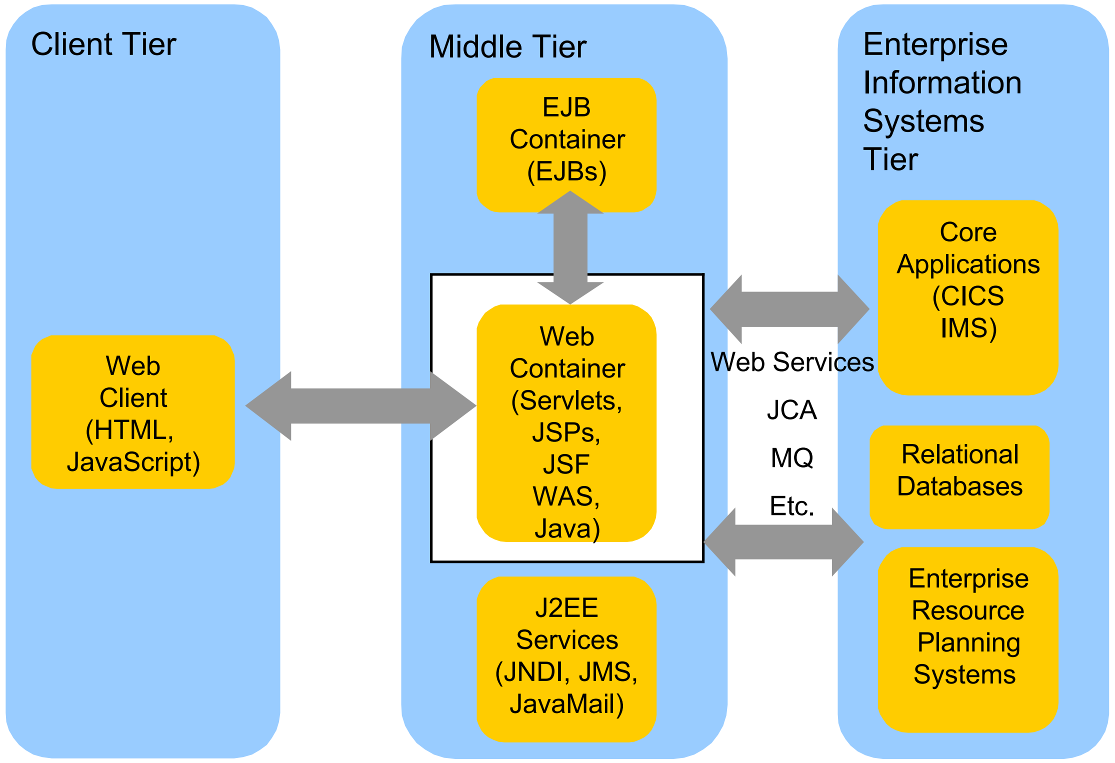
“Modern” Multitier Architecture


Client Tier
Web Client (HTML, JavaScript)
Middle Tier
EJB Container (EJBs)
Web Container (Servlets, JSPs, JSF WAS, Java)
J2EE Services (JNDI, JMS, JavaMail)
Enterprise Information Systems Tier
Core (EIS) Applications (CICS IMS)
Relational Databases
Enterprise Resource Planning Systems

It’s not that different

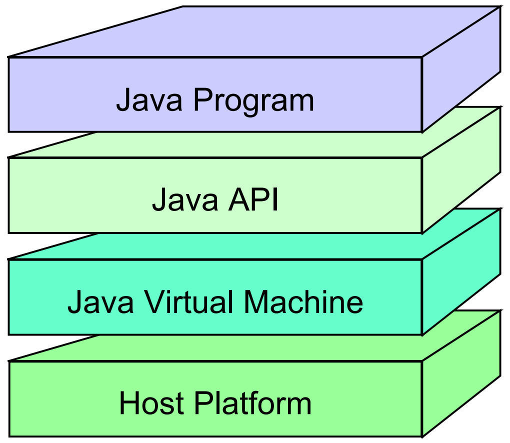


The Middle Tier


Client Tier
Web Client (HTML, JavaScript)
Middle Tier
EJB Container (EJBs)
Web Container (Servlets, JSPs, JSF WAS, Java)
J2EE Services (JNDI, JMS, JavaMail)
Web Services
JCA
MQ
Etc.
Enterprise Information Systems Tier
Core Applications (CICS IMS)
Relational Databases
Enterprise Resource Planning Systems

The Java platform

- Java is an object-oriented programming language developed by Sun Microsystems
- Java has a set of standardized class libraries that support predefined reusable functionality
- Java has a runtime environment that can be embedded in Web browsers and operating systems
- Many popular UI / Session frameworks are built on Java processing



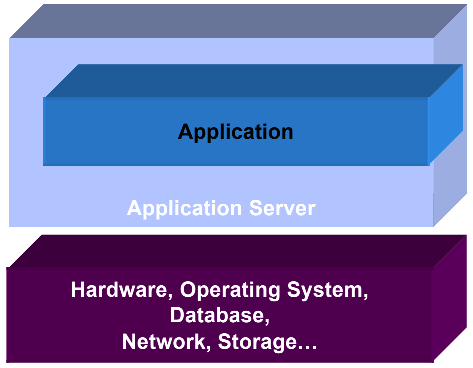
Procedural and object oriented approaches – example

- **System requirement**
  - Banking system model withdrawing money from a savings account
- **Procedural approach**
  - Identify where the data is stored
  - List the algorithmic steps necessary to perform the action
- **Object approach**
  - Identify what objects are involved; these objects will directly relate to real life objects (Bank, SavingsAccount, Teller and Transaction)
  - Show how these objects interact:
    - To enforce business rules for withdrawals
    - To modify the balance

Both have advantages in SOA – in the right place

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

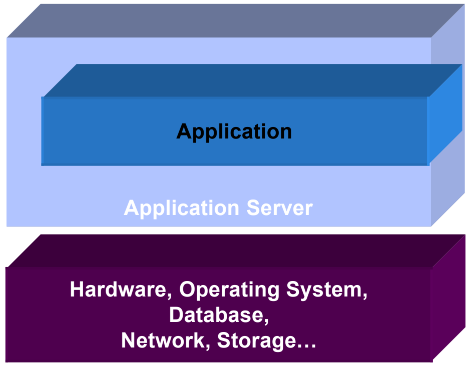


What is WebSphere Application Server?

- **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**

What is J2EE?

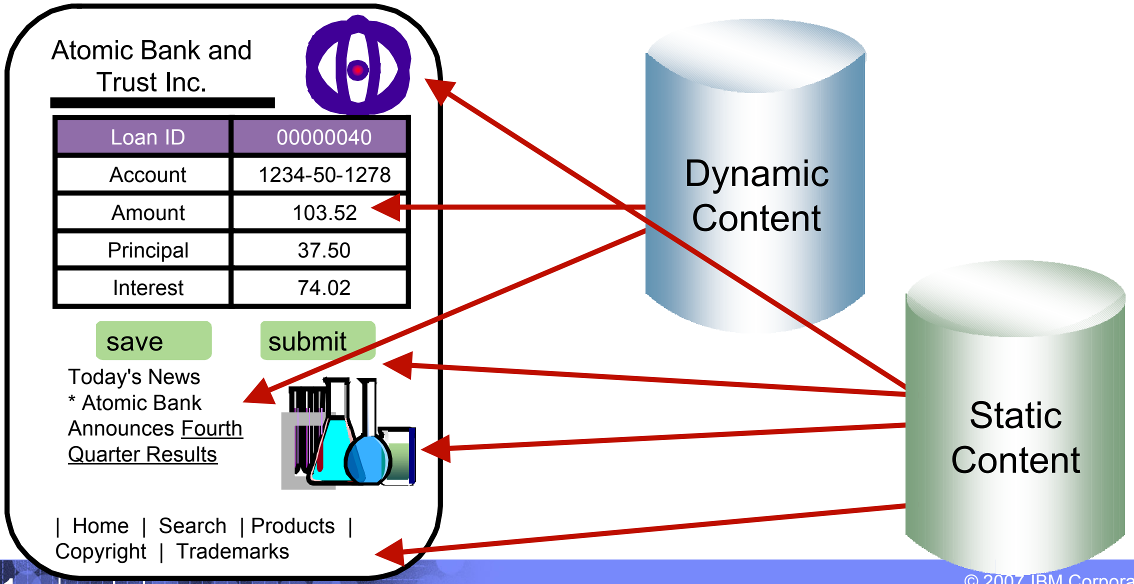
- **J2EE – Java 2 Enterprise Edition**
  - A run-time platform used for developing, deploying, and managing multitier server-centric applications on an enterprise-wide scale



- **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

Web page content

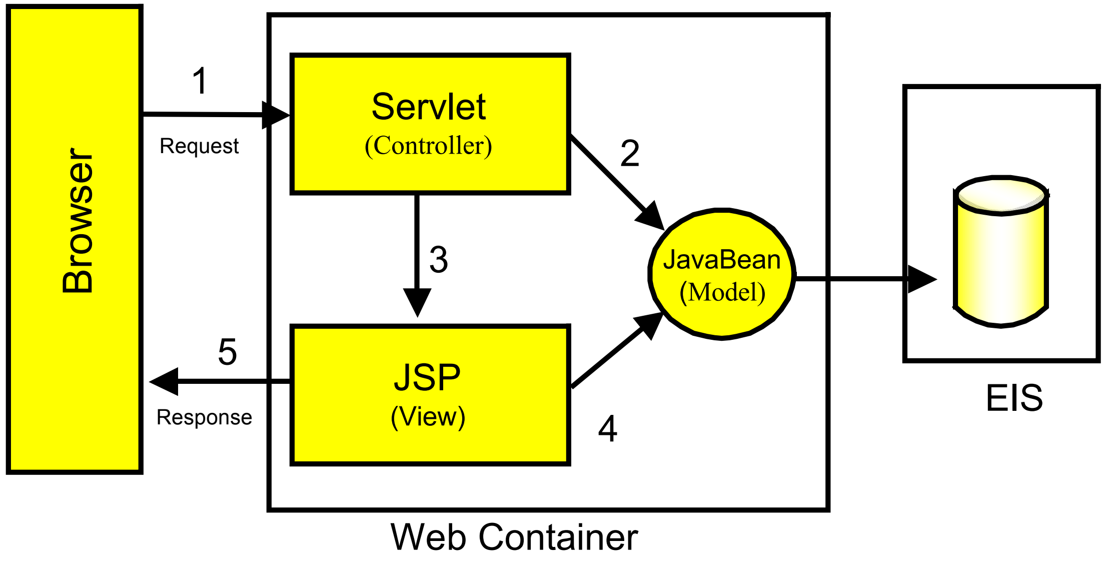
- **Content delivered to a client is composed from:**
  - Static or non-customized content
  - Customized content
- **Page layout and style are managed through HTML, XSL**



Typical J2EE Web Application Model

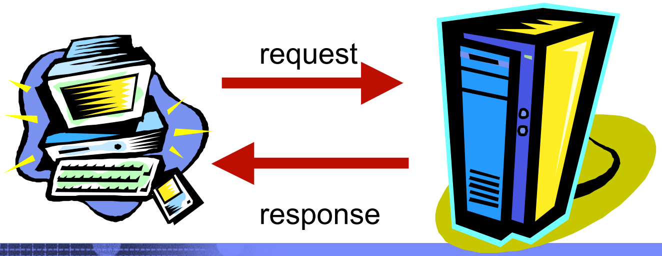
- **A request is sent to a servlet that generates dynamic content and calls a JSP page to send the content to the browser, as shown:**

MVC Design Pattern



What Is a Servlet?

- **A servlet is a standard, server-side component of a J2EE application which executes business 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**



A Simple Java Servlet Example

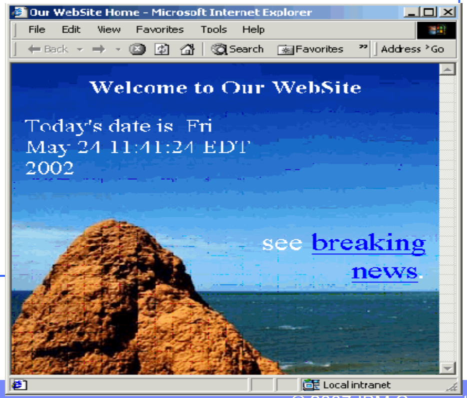
```
package com.ibm.example.servlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;
public class VerySimpleServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                         throws ServletException, IOException {
        String browser = request.getHeader("User-Agent");
        response.setStatus(HttpServletResponse.SC_OK);  // default
        response.setContentType("text/html");          // default
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>Simple servlet");
        out.println("</TITLE></HEAD><BODY>");
        out.println ("Browser details: " + browser);
        out.println("</BODY></HTML>");
}
}
```

What is JSP (JavaServer Pages)?

- JavaServer Pages is a technology that lets you mix static HTML with dynamically generated HTML
- JSP technology allows server-side scripting
- A JSP file (has an extension of .jsp) contains any combination of:
  - JSP syntax
  - Markup tags such as HTML or XML

A simple JSP example

```
<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>
```


Our WebSite Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Address Go
Local intranet

Welcome to Our WebSite

Today's date is Fri May 24 11:41:24 EDT 2002

see breaking news.

JSP or Servlet?

- **Writing HTML code in a servlet is tedious and difficult to maintain**
- **Java code embedded in a JSP is difficult to reuse and maintain**
- **Use servlets to:**
  - Determine what processing is needed to satisfy the request
  - Validate input
  - Work with business objects to access the data and perform the processing needed to satisfy the request
  - Control the flow through a Web application
- **Use JSP pages to format and displaying the content generated by your servlets**

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
  - **EGL**, IBM's enterprise generation or *business* language supports JSF

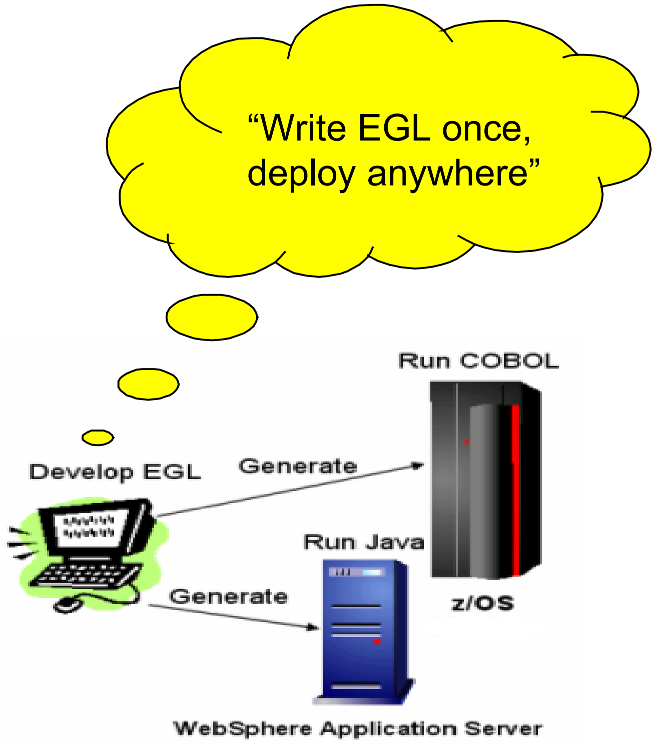
What is EGL?

Enterprise Generation Language (EGL)

- Is a development environment and programming language that lets users write full-function applications quickly
- Can be used to create text-based user interfaces for migration of existing applications
- Focus is on the business problem rather than on software technologies
- Is written independently of the target platform
- Can be generated into Java or COBOL programs
- Is well-suited to procedural programmers
- Is a high-level language which promotes iterative development and testing early in the development cycle

EGL - generation

- **Runtime code generated for appropriate platform**
  - **Java** for Windows, Linux, and so forth
  - **COBOL for z/OS**
  - Uses SQL transparently
- **EGL can be used to create "full" Web-based applications including Web UI's**
  - JavaServer Faces application is generated for runtime code
  - Runs on WebSphere Application Server


"Write EGL once, deploy anywhere"
Develop EGL
Generate
Run COBOL
z/OS
Run Java
Generate
WebSphere Application Server

EGL – key high level language abstractions

- **Data Access:**
  - Common Verbs for data access (Get, Add, Replace, Delete)
  - Abstracts access to SQL, Indexed, Relative, Serial, DL/I, MQ, Services
  - Allows complete access to SQL statement if needed
  - Common Error Handling

```
function allLoans()
      loans LoanRec[];
      get loans;
end

function loansInFlorida()
      loans LoanRec[];
      get loans with #sql{
            select *
            from LOANREC
            where state = "FL"};
end
```

- **Validation/Editing Rules**
  - Define formatting & validation rules once in common place
  - Reuse data items for Records, screens, reports

```
DataItem Password char(10) {
      validatorFunction = "passwordValidation",
      displayUse = secret,
      displayName = "Enter your Password",
      inputRequired = yes}
end
```

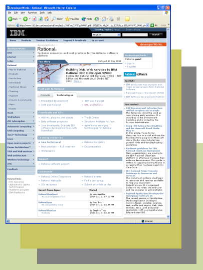
- **Remote Invocation**
  - Call COBOL, RPG, C, Java
  - Linkage information separated from code…simplifies development

```
function callHelloWorldOniSeries()
      salutation char(30);
      call helloworld salutation;
end
```

- **Transaction Control:**
  - JDBC, CICS, IMS

```
function loansInFlorida()
      startTransaction(myLoanTransaction);
      ...
      commit();
      ...
      rollback();
end
```

EGL – simple programming model



- **Page Handlers**
  - Contain functions and data related to a .jsp
  - Should be mostly “Controller Logic”

```
PageHandler customerInfoPage {
      view = "customerInfoPage.jsp",
      title = "Customer Information",
      onPageLoadFunction = "onPageLoad"}

function onPageLoad()
      loans LoanRec[];
      get loans;
end
...
end
```

- **Report Handler**
  - Call-out to EGL “ReportHandler”
  - Open Source Reporting Framework

EmplReport.jasper

```
ReportHandler customerList

function afterPageInit()
...
end

end
```

Controller Logic/User Interaction

Business Logic

- **Programs**
  - Used for single point of entry situations
  - TUI program, Batch program, GUI program

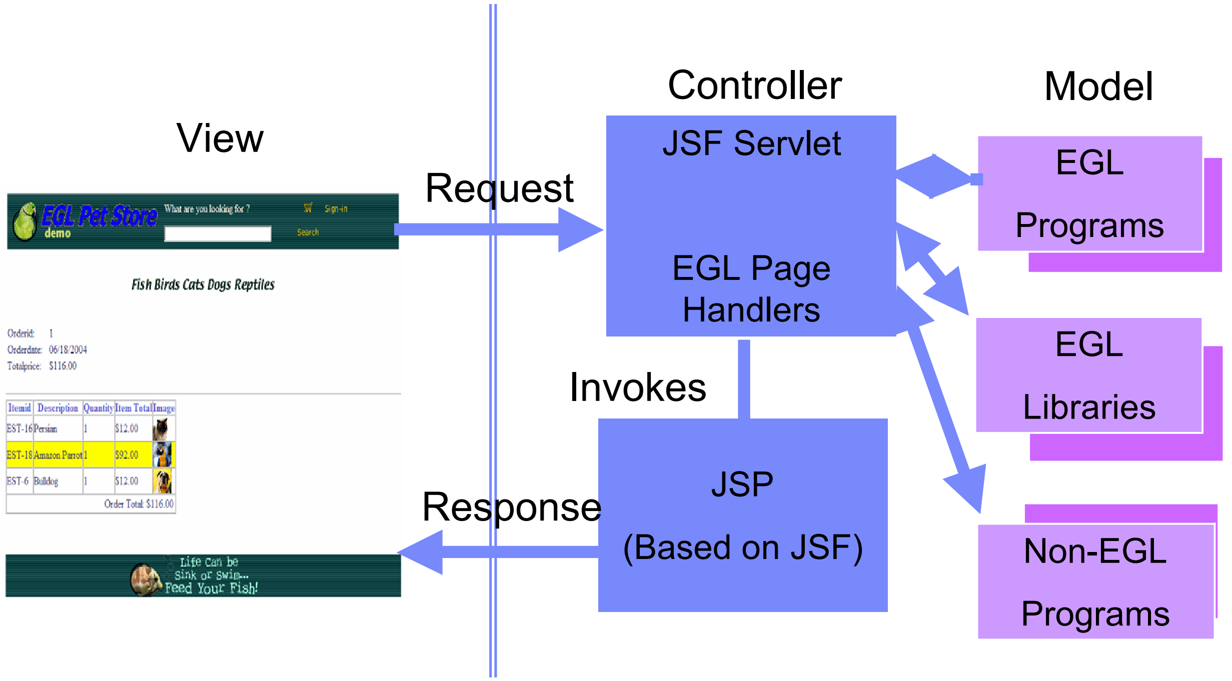
```
Program MyProgram

function myFunction()
      salutation char(30);
      call helloworld salutation;
end
```

- **Services**
  - “Business Logic” for web apps
  - Multiple entry points
  - Invoke by function

CustomerService

```
Service Customer

Function getAllCustomers()
end
...

end
```

EGL and JSF



Orderid: 1
Orderdate: 06/18/2004
Totalprice: $116.00

Itemid	Description	Quantity	Item Total	Image
EST-16	Persian	1	$12.00	
EST-18	Amazon Parrot	1	$92.00	
EST-6	Bulldog	1	$12.00	
			Order Total: $116.00	

JSP's / JSF / EGL and COBOL

- **JSP's are synonymous with EXEC CICS Send Map and Receive Map processing**
  - If a CICS program only processed screens – to request business processing – or work – it would need to either Link, XCTL or Calls in COBOL.
  - JSP is a similiar concept.
- **Java Server Faces provides a framework to build UI oriented forms linked with processes such as Web Services.**
  - Performs similar function as existing CICS programs which perform send/receive processing and input validation.
- **Java server faces consist of Java Server pages – which handle the build and catching of forms and user information – and page handlers which validate information and provide control calls into back end services.**

EGL Web – C.I.C.S. Programming Similarities

```
package pagehandlers;
import data.*;

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

Page Data
~
BMS Map Fields

```
//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;
```

Load values from the database

"Send map"

```
    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
```

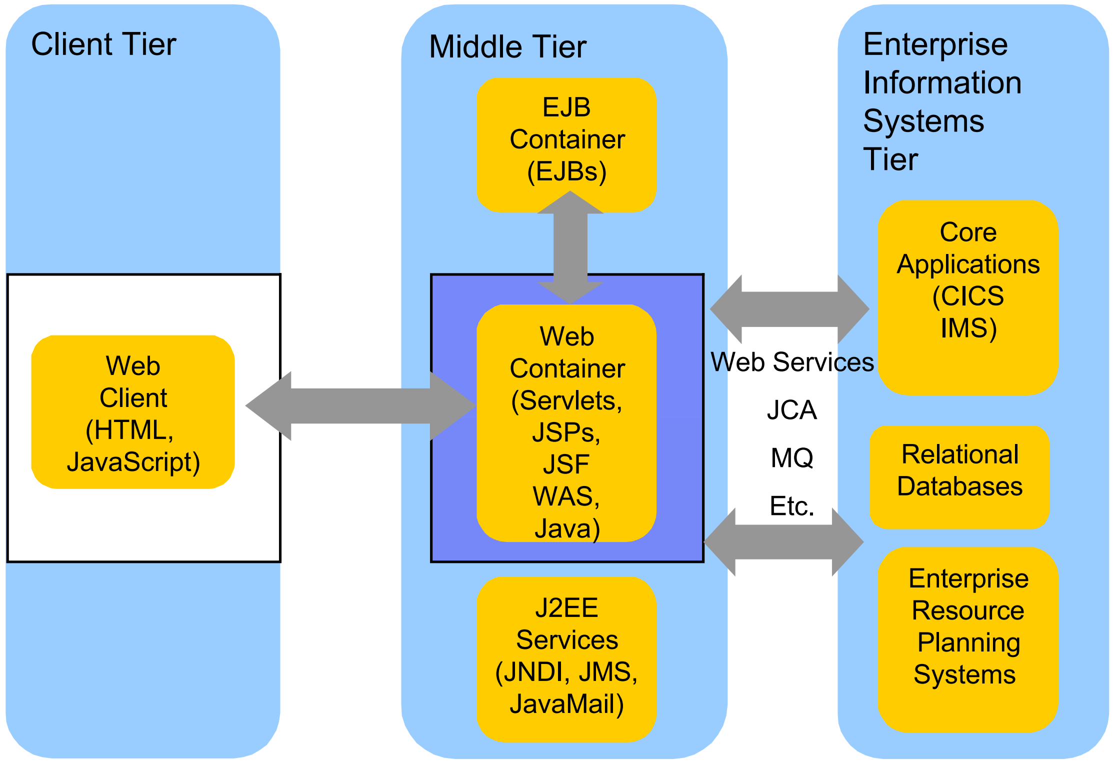
"Receive map"

Process user-input values

Update Database

```
    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
```

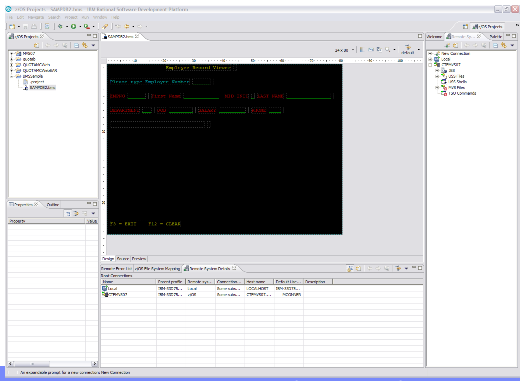
The Client


Client Tier
Web Client (HTML, JavaScript)
Middle Tier
EJB Container (EJBs)
Web Container (Servlets, JSPs, JSF WAS, Java)
J2EE Services (JNDI, JMS, JavaMail)
Web Services
JCA
MQ
Etc.
Enterprise Information Systems Tier
Core Applications (CICS IMS)
Relational Databases
Enterprise Resource Planning Systems

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.**
- **Hypertext Markup Language consists of:**
  - **Hypertext.** The way of creating web documents – and of linking multiple documents together. HTML offers support for both document as well as multimedia links.
  - Tags or controls: Pieces of code that are used to create links. All browsers let you know when you've selected an active area of the screen.
    - For example <head> marks where a heading starts and </head> marks where it ends.
    - Popular tags include:
      - Text Tags – Logical structure for content
      - Link Tags – to links such as hyperlinks, image links
      - Style sheet tags – how content is rendered
      - and many more….

- **See the green screenshot – displayed inside of the WDz BMS Map Editor, together with with the BMS Macros that are input to generate the code – that upon execution causes the "green screen" to be displayed.**
  - WDz provides similar support for HTML screens


SAMPDB2.bms - IBM Rational Software Development Platform

```
                    Employee Record Viewer

Please type Employee Number

EMPNO       First Name           MID INIT   LAST NAME

DEPARTMENT     JOB          SALARY          PHONE




F3 = EXIT    F12 = CLEAR
```

Remote System Details — Root Connections

Name	Parent profile	Remote sys...	Connection...	Host name	Default Use...	Description
Local	IBM-33D75...	Local	Some subs...	LOCALHOST	IBM-33D75...	
CTFMVS07	IBM-33D75...	z/OS	Some subs...	CTFMVS07....	MCONNER	

BMS and HTML

BMS

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.

```
SAMPDB2  DFHMSD
     TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL,STORAGE=AUTO,
     *
          CTRL=FREEKB,EXTATT=YES,TERM=3270-2,TIOAPFX=YES,       *
          MAPATTS=(COLOR,HILIGHT,OUTLINE,PS,SOSI),              *
          DSATTS=(COLOR,HILIGHT,OUTLINE,PS,SOSI)
MAP1      DFHMDI SIZE=(24,80),                                  *
          COLUMN=1,                                             *
          LINE=1
```

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

```
      DFHMDF POS=(3,1),LENGTH=27,                               *
          INITIAL='Please type Employee Number',                *
          ATTRB=(PROT,NORM)
```

Input Fields. Defined with DFHMDF macro. 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.

```
EMPONUMINPUT DFHMDF POS=(3,29),LENGTH=6,                        *
          ATTRB=(UNPROT,NORM),HILIGHT=UNDERLINE
```

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>
```

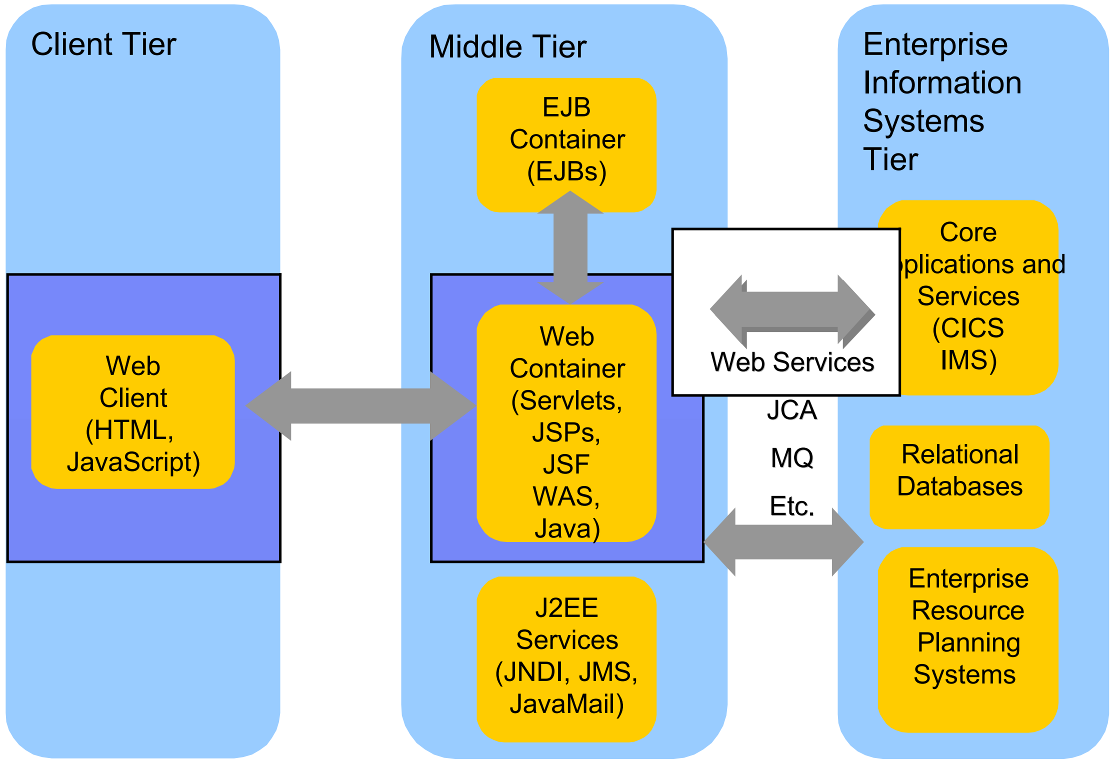
Text headings including location definition, colors, attributes, etc.

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

Input fields

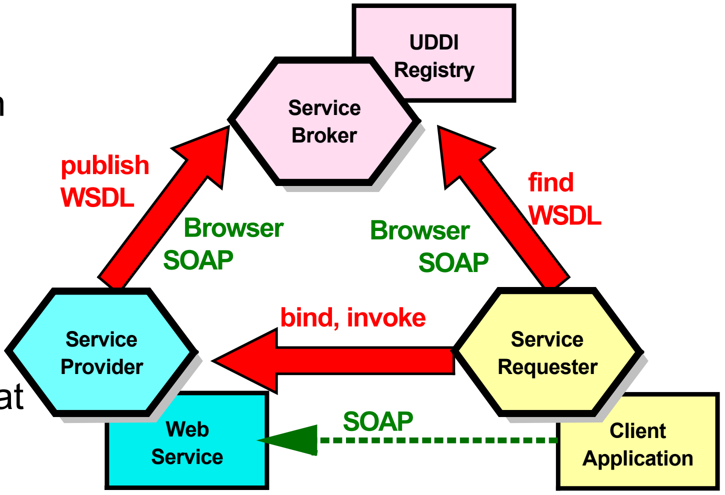
```
<td>&nbsp;</td>
<td colspan="6" nowrap><h:inputText
     value="#{pc_MAP1Page.map1Bean.emponuminput}" required="false" style="color:
     #00ff00" size="6" id="emponuminput"></h:inputText></td>
<td>&nbsp;</td>
<td colspan="44">&nbsp;</td>
<tr><td colspan="80">&nbsp;</td>
<tr><td>&nbsp;</td>
```

Connectivity


Client Tier
Web Client (HTML, JavaScript)
Middle Tier
EJB Container (EJBs)
Web Container (Servlets, JSPs, JSF WAS, Java)
J2EE Services (JNDI, JMS, JavaMail)
Web Services
JCA
MQ
Etc.
Enterprise Information Systems Tier
Core plications and Services (CICS IMS)
Relational Databases
Enterprise Resource Planning Systems

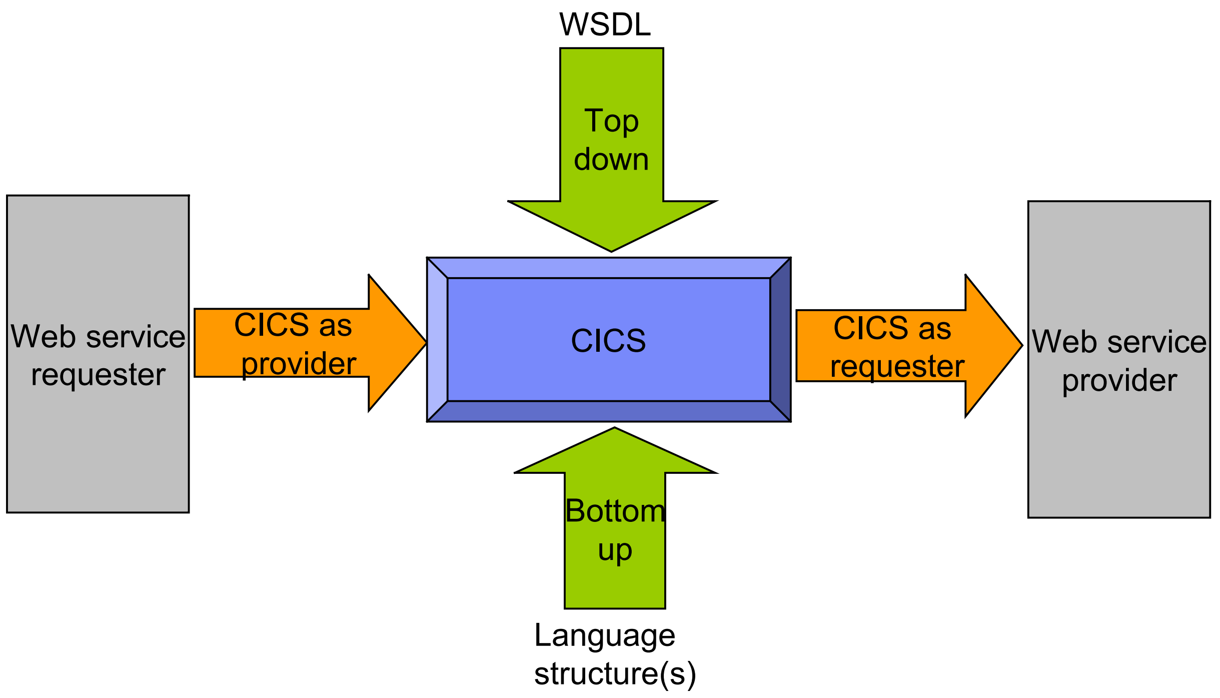
Web Services

- **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 Interoperablity Organization)**
  - Ensure interoperability

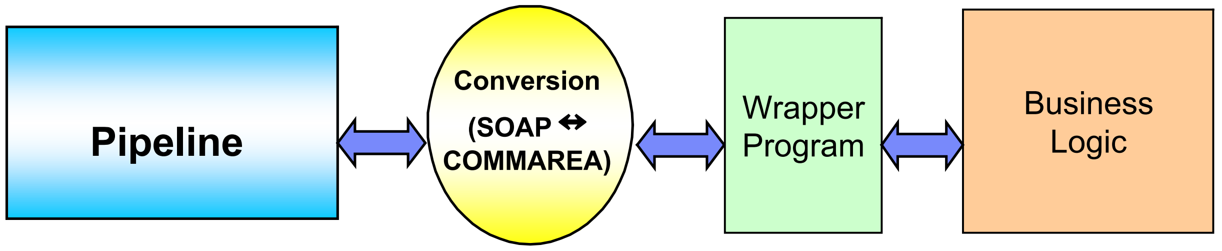


The entire industry is agreeing on one set of standards !!

Web Services Enablement Styles



Where a wrapper program fits in



XML Terminology

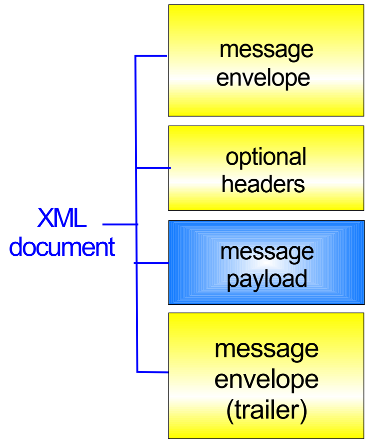
- SOAP and WSDL are based on XML
- A tag / attribute based syntax
- Format of XML file described in
  - **DTD** – Document Type Definition
  - **XSD** – XML Schema Definition
- XML files are
  - Well-formed (syntax is ok – matching tabs, etc.)
  - Valid (obeys rules in DTD or XSD) (CICS can validate)
- Namespaces
  - Avoids name collisions
  - A set of names (XML tags) that apply to a certain space in a document

XML – Basic Parts

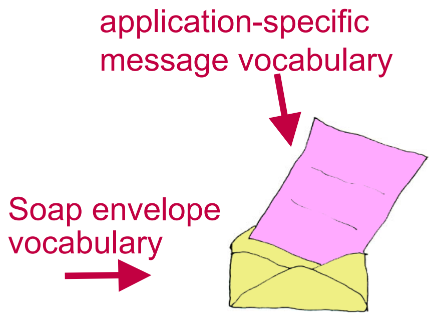
```
<?xml version="1.0" standalone="no" encoding="UTF-8" ?>  ← XML Declaration
<!DOCTYPE shirt SYSTEM "http://shirts.com/xml/dtds/shirt.dtd">  ← Document
<shirt>                                   ← root element           type
   <model>CICS Tee</model>          ← child of root                declaration
   <brand>Tommy Hilltop</brand>                ← end tag
   ↑———————————————————————————————————————————— start tag
   <price currency="USD“>10.95</price>         ← attribute
   <fabric content="70%“>cotton</fabric>       ← attribute
   <fabric content="30%">polyester</fabric>
   <on_sale/>                                  ← empty element
   <options>
      <colorOptions>
         <color>red</color>
         <color>white</color>
      </colorOptions>
      <sizeOptions>
         <!-- Medium and large are out of stock -->      ← comment
         <size>small</size>
         <size>x-large</size>
      </sizeOptions>
   </options>
   <order_info>Call &phone;</order_info>       ← entity reference
</shirt>
```

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

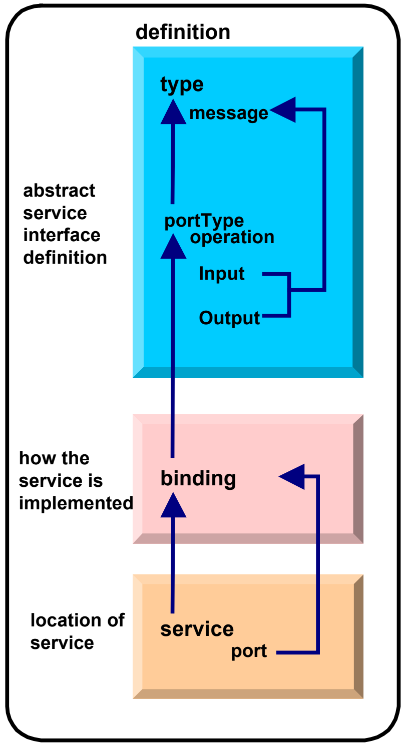
```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV=
     "http://www.w3.org/2001/06/soap-envelope"
      SOAP-ENV:encodingStyle=
     "http://www.w3.org/2001/06/soap-encoding">

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

Result returned in Body

app-specific message

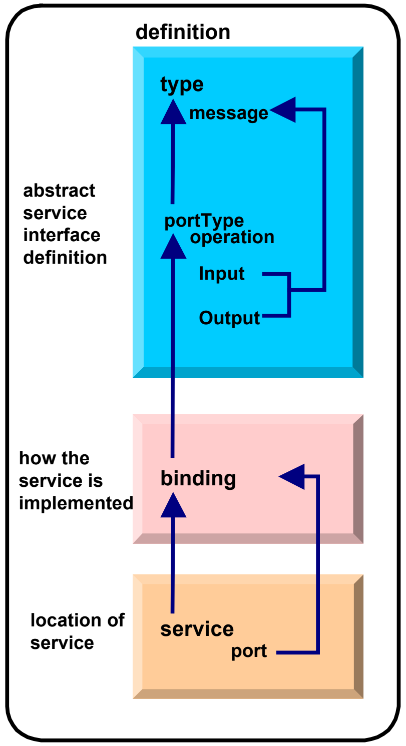
WSDL - Web Service Description Language

- Open Standard
- XML resume 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

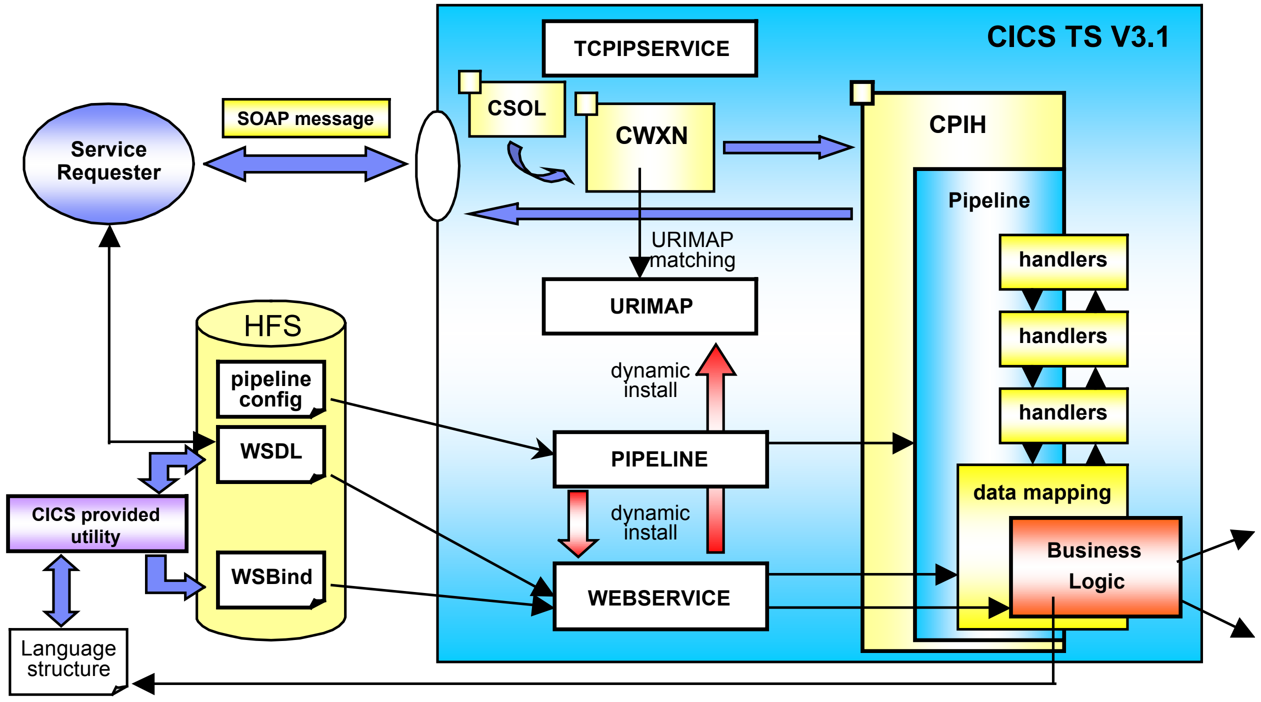
- **Service Interface**
  - Operation (business functions)
    - Input Message (0 or 1) and Output Message (0 or 1)
      - 1 or more parts
      - Parts may be simple or complex
      - Complex parts may have multiple elements
- **Service binding**
  - Definition of the physical service interface implementation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

CICS as a service provider

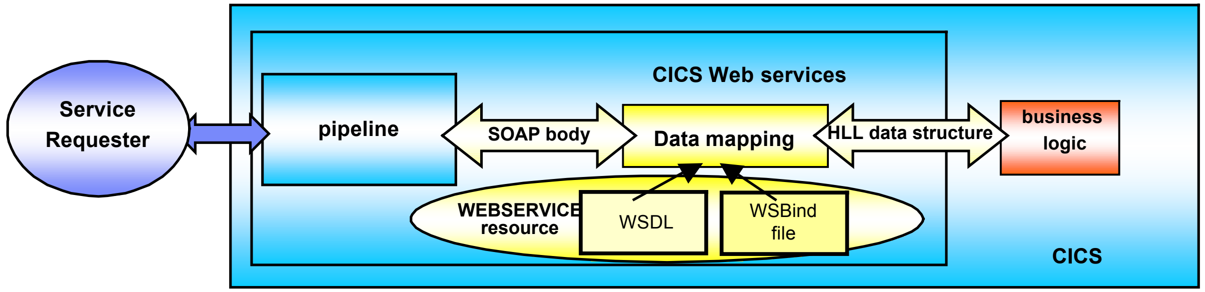


Defining the CICS Web Services Resources

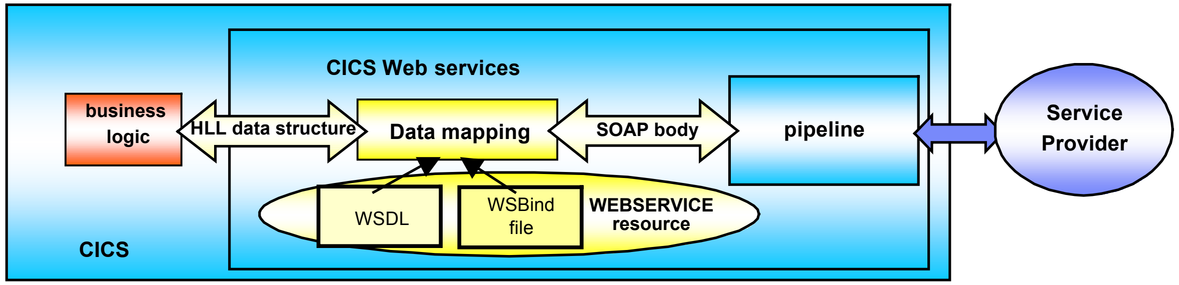
- **Define a TCPIPSERVICE (or WMQ) and a PIPELINE**
- **Then install the PIPELINE definition and issue CEMT PERFORM PIPELINE SCAN**
- **CICS uses the PIPELINE definition to**
  - Locate the WSBind file
  - From the WSBind file, CICS will dynamically create a WEBSERVICE resource
  - CICS will also dynamically create a URIMAP definition
- **Can define everything individually if preferred**

CICS usage of the WSBind file

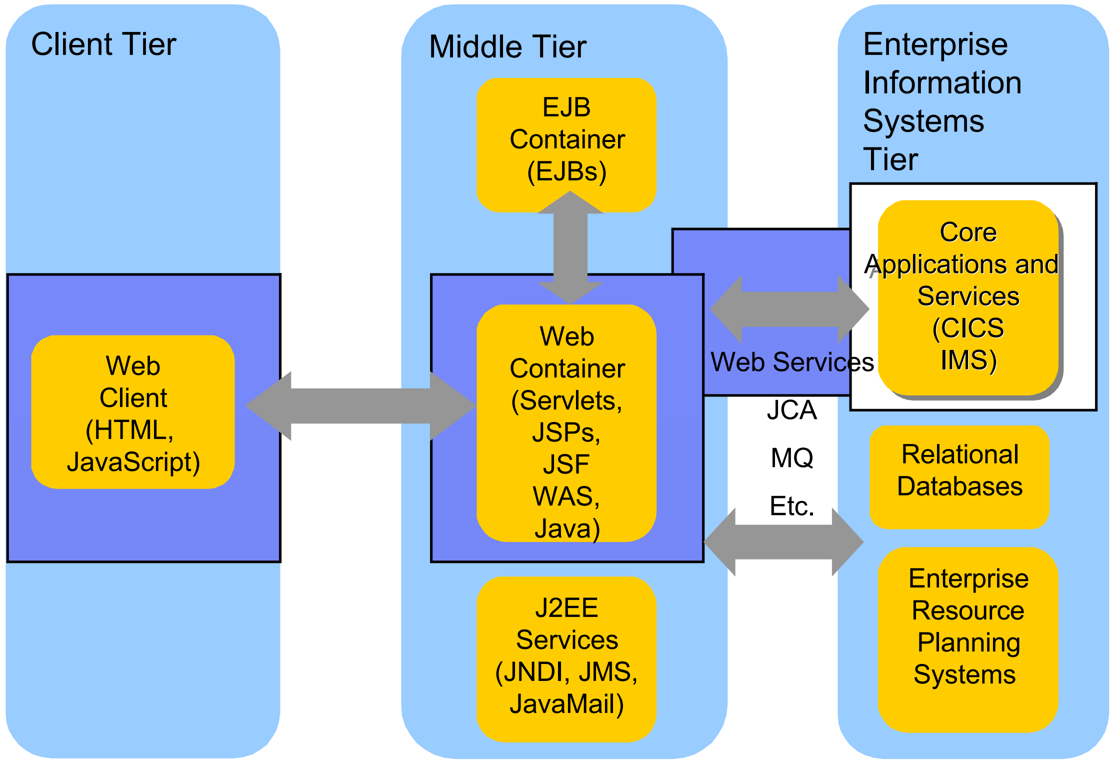
- CICS as a service provider



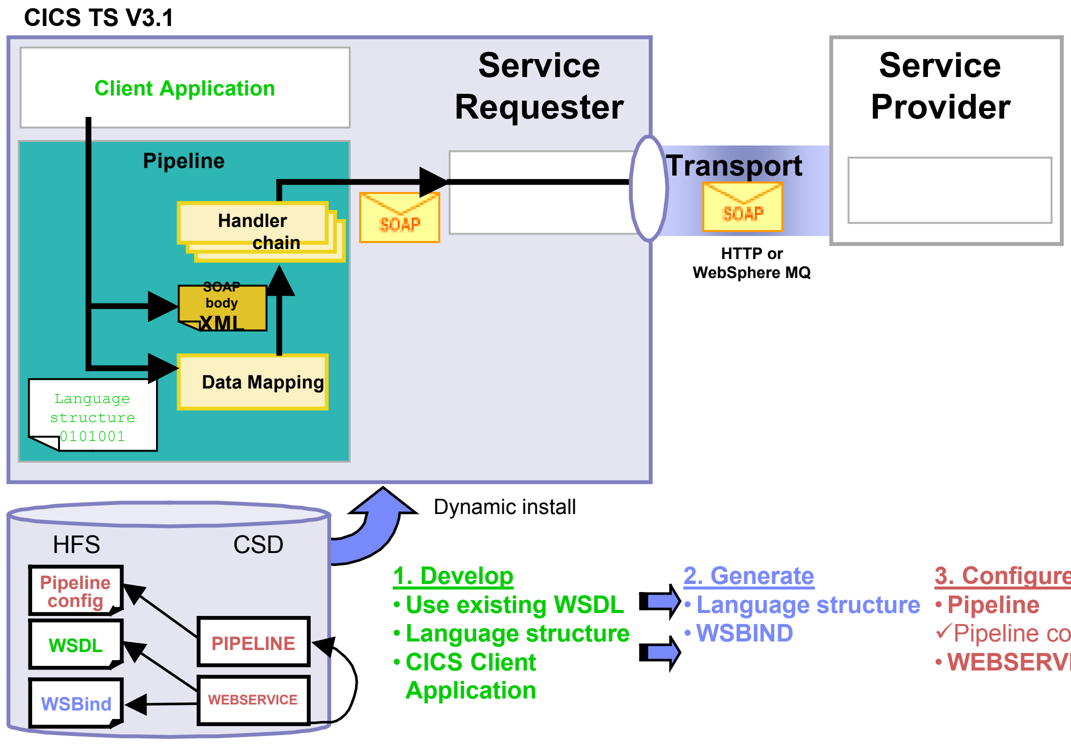
- CICS as a service requester



The Business Tier


Client Tier
Web Client (HTML, JavaScript)
Middle Tier
EJB Container (EJBs)
Web Container (Servlets, JSPs, JSF WAS, Java)
J2EE Services (JNDI, JMS, JavaMail)
Web Services
JCA
MQ
Etc.
Enterprise Information Systems Tier
Core Applications and Services (CICS IMS)
Relational Databases
Enterprise Resource Planning Systems

CICS as a Web service requester



1. Develop
- Use existing WSDL
- Language structure
- CICS Client Application

2. Generate
- Language structure
- WSBIND

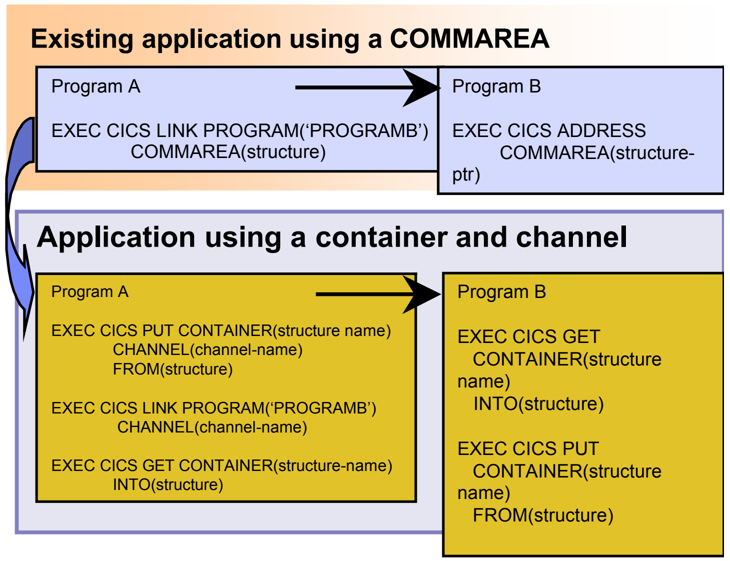
3. Configure
- Pipeline
✓Pipeline configuration
- WEBSERVICE

CICS API's

- **Invoking a Web Service from a CICS application program**
  - CICS as a service requester
    - EXEC CICS INVOKE WEBSERVICE ( ) CHANNEL ( ) URI ( ) OPERATION ( )
      - WEBSERVICE: name of the Web Service to be invoked
      - CHANNEL: name of the channel containing data to be passed to the Web Service (DFHWS-DATA container)
      - URI: Universal Resource Identifier of the Web Service (optional)
      - OPERATION: name of the operation to be invoked

Data Exchange between CICS programs with Containers and Channels

- Offers a more flexible and intuitive alternative to the COMMAREA
- Enables large amounts of data to be passed between CICS applications
  - Not subject to 32KB restriction
- Optimized and managed by CICS
- Requires minimal application changes required to use



Existing application using a COMMAREA

Program A

```
EXEC CICS LINK PROGRAM('PROGRAMB')
          COMMAREA(structure)
```

Program B

```
EXEC CICS ADDRESS
      COMMAREA(structure-
ptr)
```

Application using a container and channel

Program A

```
EXEC CICS PUT CONTAINER(structure name)
        CHANNEL(channel-name)
        FROM(structure)

EXEC CICS LINK PROGRAM('PROGRAMB')
        CHANNEL(channel-name)

EXEC CICS GET CONTAINER(structure-name)
        INTO(structure)
```

Program B

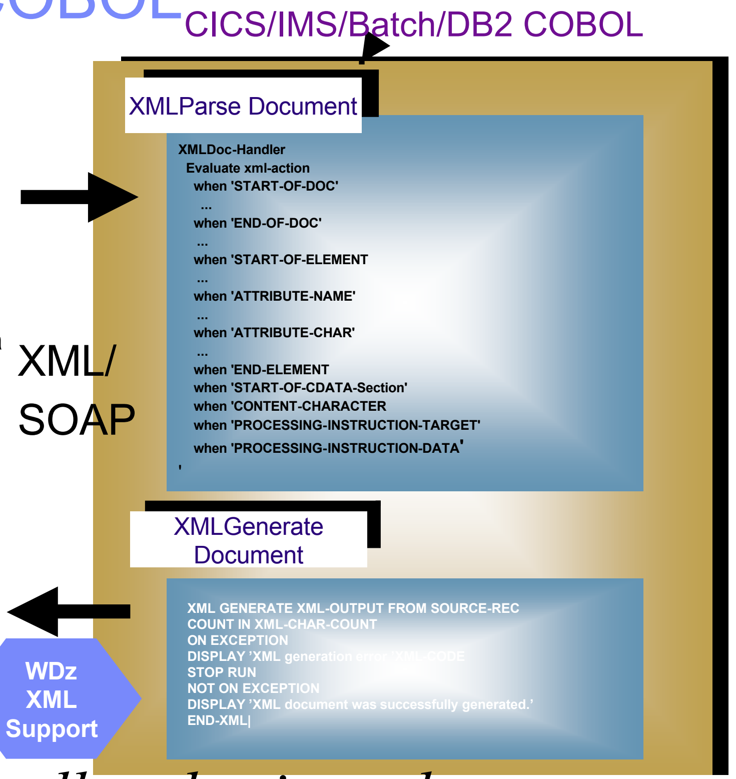
```
EXEC CICS GET
  CONTAINER(structure
name)
  INTO(structure)

EXEC CICS PUT
  CONTAINER(structure
name)
  FROM(structure)
```

IBM Enterprise COBOL

- **XML Language based generation from COBOL data structure**
  - XMLGenerate Verb
  - WebSphere EJB support
  - DB2 V8
- **High speed XML Sax based parsing**
- **Object Oriented Support for Java COBOL Interoperability**
- **Unicode support**
- **CICS and DB2 integrated preprocessor**
- **Raise 16Mb COBOL data size limit**
  - Picture clause replication:

    01 A PIC X(134217727).
  - OCCURS::

    05 V PIC X OCCURS 134217727 TIME



```
XMLDoc-Handler
 Evaluate xml-action
  when 'START-OF-DOC'
   ...
  when 'END-OF-DOC'
   ...
  when 'START-OF-ELEMENT
   ...
  when 'ATTRIBUTE-NAME'
   ...
  when 'ATTRIBUTE-CHAR'
   ...
  when 'END-ELEMENT
  when 'START-OF-CDATA-Section'
  when 'CONTENT-CHARACTER
  when 'PROCESSING-INSTRUCTION-TARGET'
  when 'PROCESSING-INSTRUCTION-DATA'
'
```

```
XML GENERATE XML-OUTPUT FROM SOURCE-REC
COUNT IN XML-CHAR-COUNT
ON EXCEPTION
DISPLAY 'XML generation error 'XML-CODE
STOP RUN
NOT ON EXCEPTION
DISPLAY 'XML document was successfully generated.'
END-XML|
```

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**
- **CICS 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**
- **CICS and WebSphere Application Server are strategic middleware products that together…**
  - Interoperate - Web services, JCA, Enterprise JavaBeans
  - Exploit and complement z/OS qualities of service
  - Have high qualities of service, low cost per transaction, excellent security.

Demo

Modern Application Architecture – Building and testing a JSF/COBOL process.

- **Demo of WDz used to create a simple, understandable visual and business application process for deployment.**
- **The session shows how to build and deploy composite CICS and WebSphere applications using the IBM WebSphere Studio tooling and the Enterprise Compilers. Composite applications are applications which are assembled from independent component parts, using Web and Web Services standards.**