The background of the slide is a vibrant blue with a series of thin, white, wavy lines that create a sense of motion and depth. The lines are most prominent on the left side and curve towards the right, creating a dynamic, organic feel. The overall color palette is various shades of blue, from deep navy to light sky blue.

Modern Application Development Featuring Web 2.0 for System z

Rational Developer for System z

Rational. software

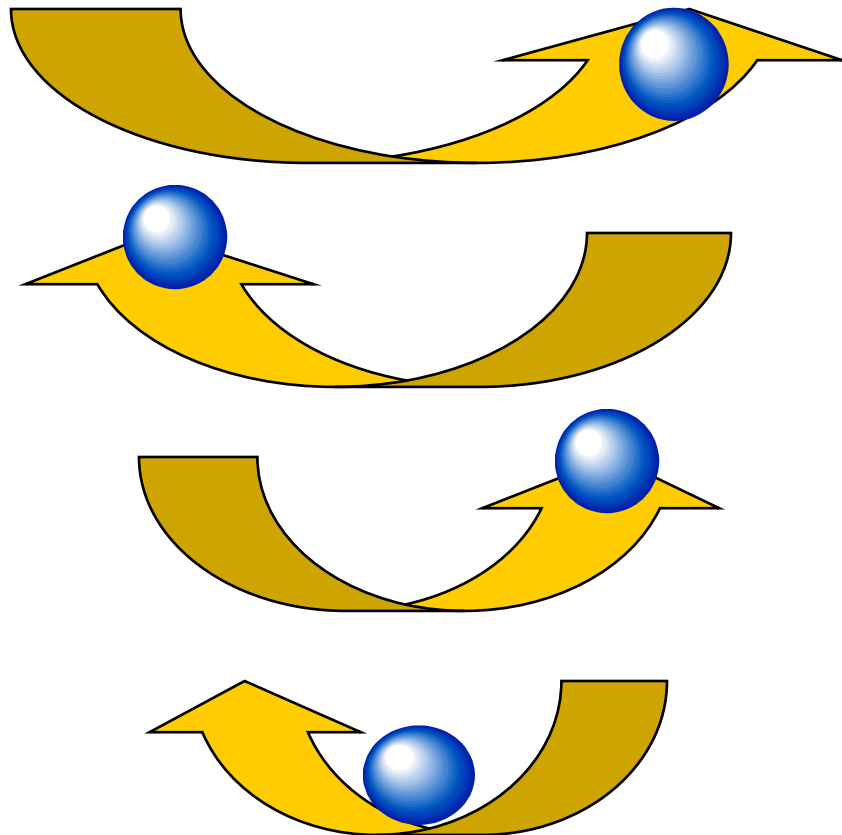
Agenda

- **Introduction**
- **Architectures**
 - Multi-tier Web Architecture
 - Web 2.0 – the web as a platform
- **Client technologies**
 - HTML, CSS, Javascript, AJAX, JSON
- **Connectivity**
 - Web Services, XML, SOAP, REST WSDL, Mashups
- **Business Tier**
 - CICS
 - COBOL

Web 2.0 technologies highlight the next pendulum swing between client and server function.

Client

Server



Mainframe computing

“Dumb” little green screen clients
Omnipotent big mainframe servers

Client-server computing

“Smart” Personal Computer clients
Simple file and database servers

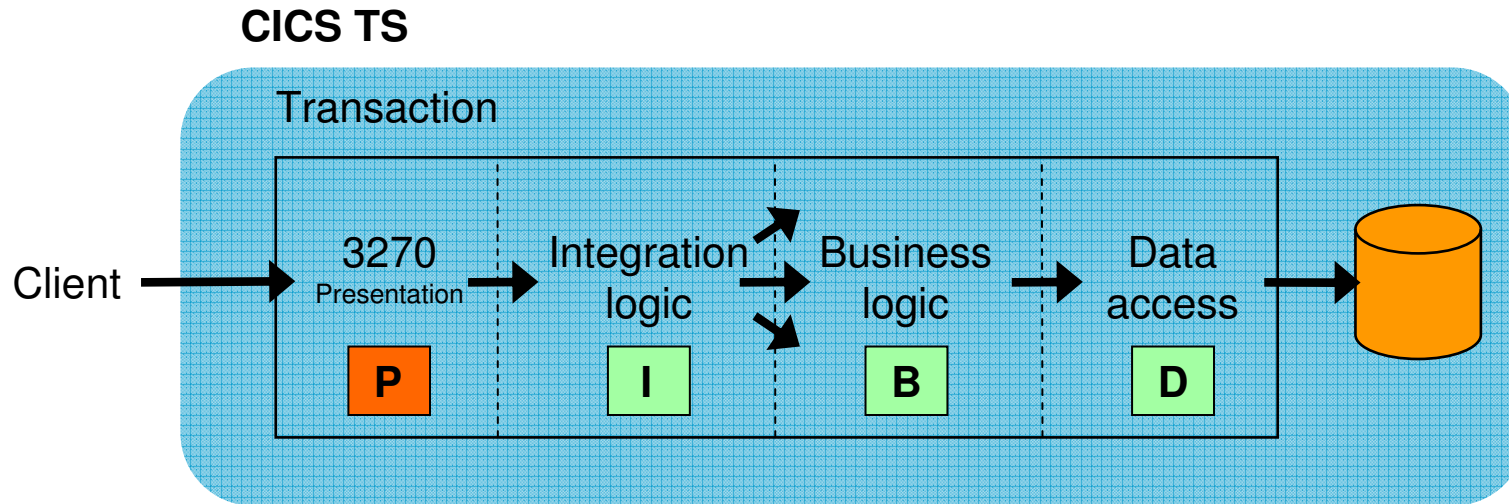
Web (1.0) computing

Light Web Browser clients
Rich application and database servers

Web 2.0 computing

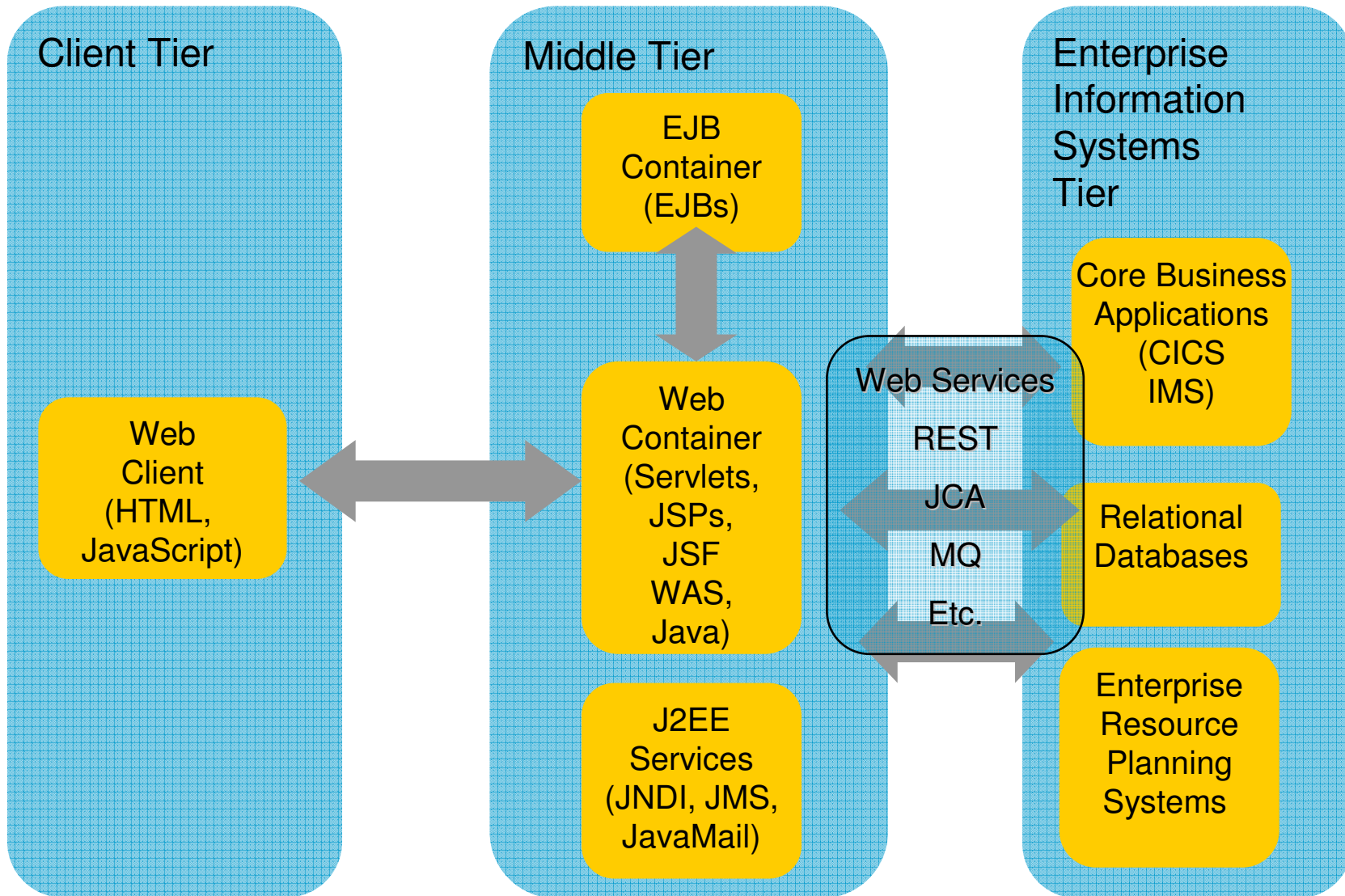
Rich Internet Application clients
Lighter application and database servers

Modern “CICS” architecture



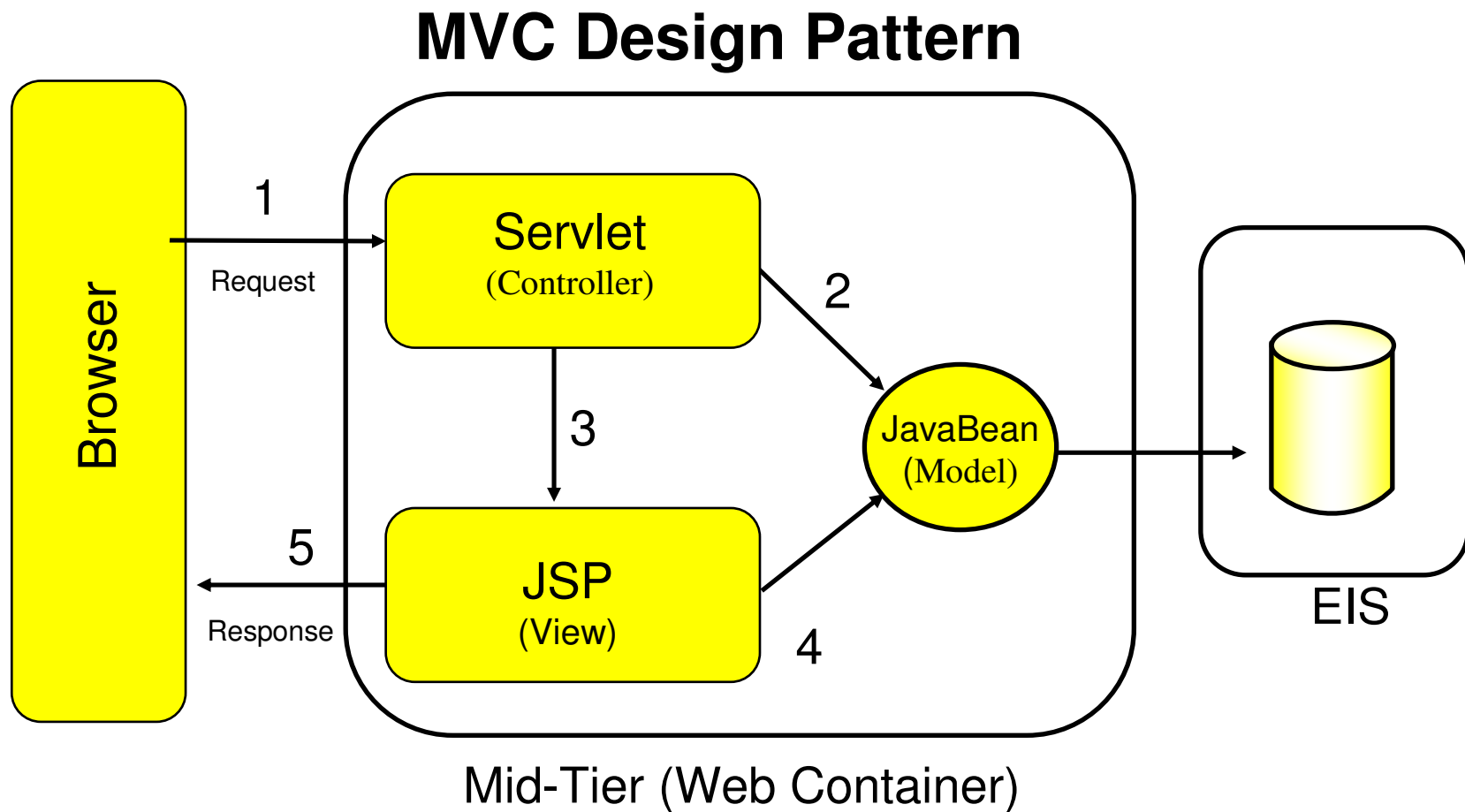
- **Best practice in CICS application design is to separate key elements of the application, in particular:**
 - Presentation logic 3270, HTML, XML
 - Integration or aggregation logic Menu, router, tooling
 - Business logic COBOL, PL/I, Reusable component
 - Data access logic VSAM, DB2, IMS, ...
- **Provides a framework for reuse and facilitates separation of concerns, clear interfaces, ownership, and optimisation**

Web 1.0 Architecture

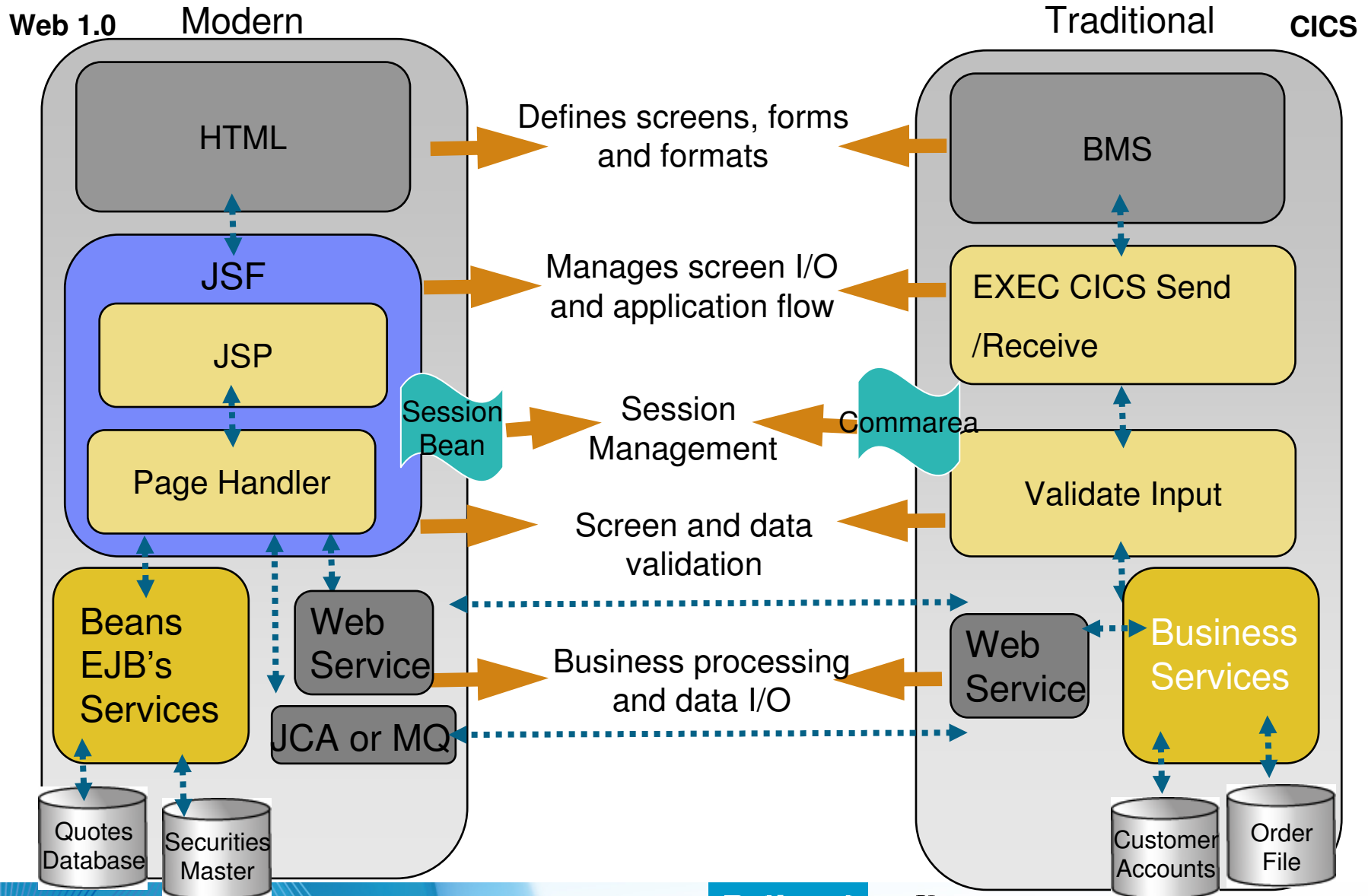


Web 1.0 Application Model

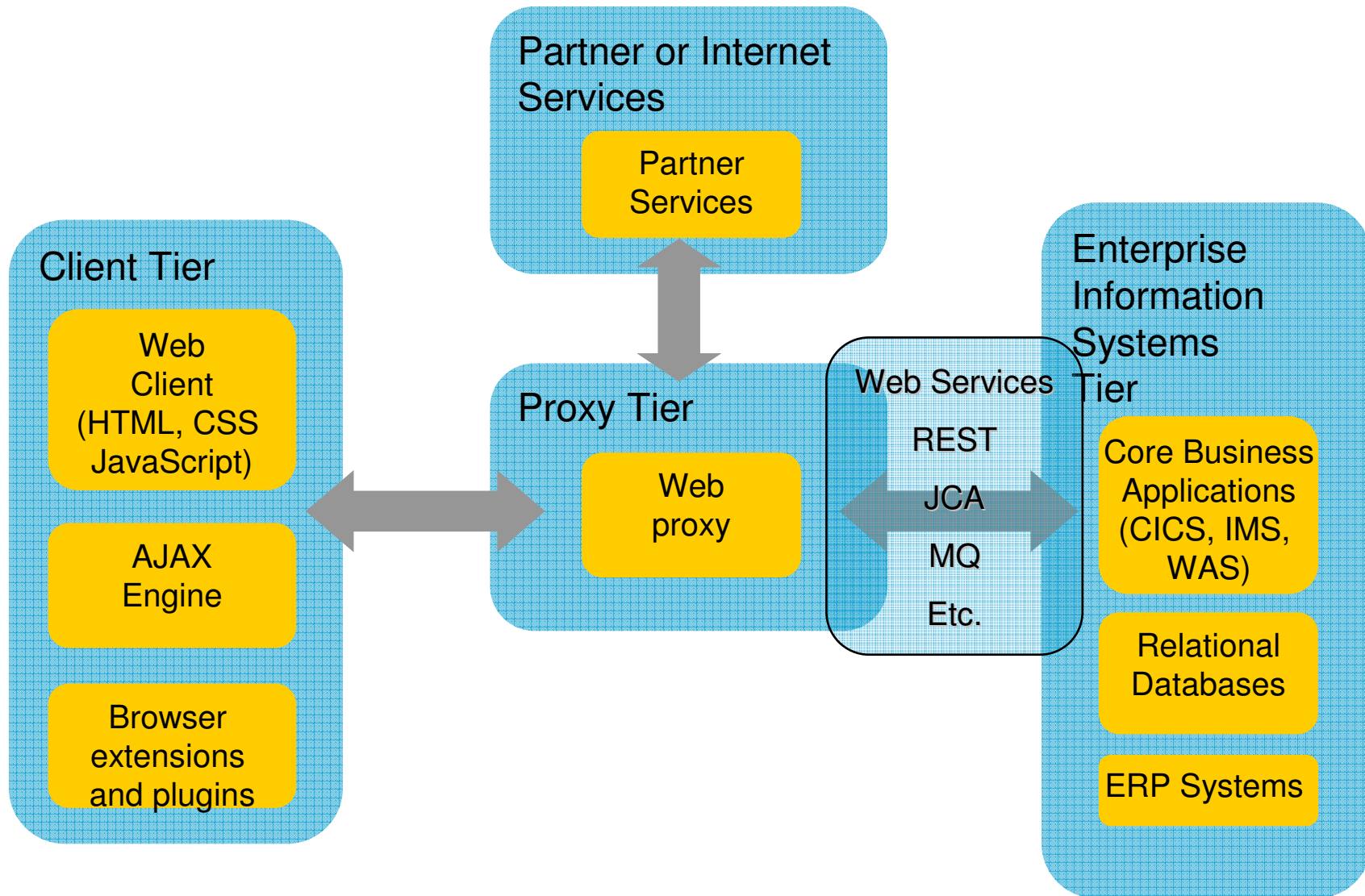
- A request is sent to a servlet that generates dynamic content and calls a JSP page to send the content to the browser, as shown:



It's not that different

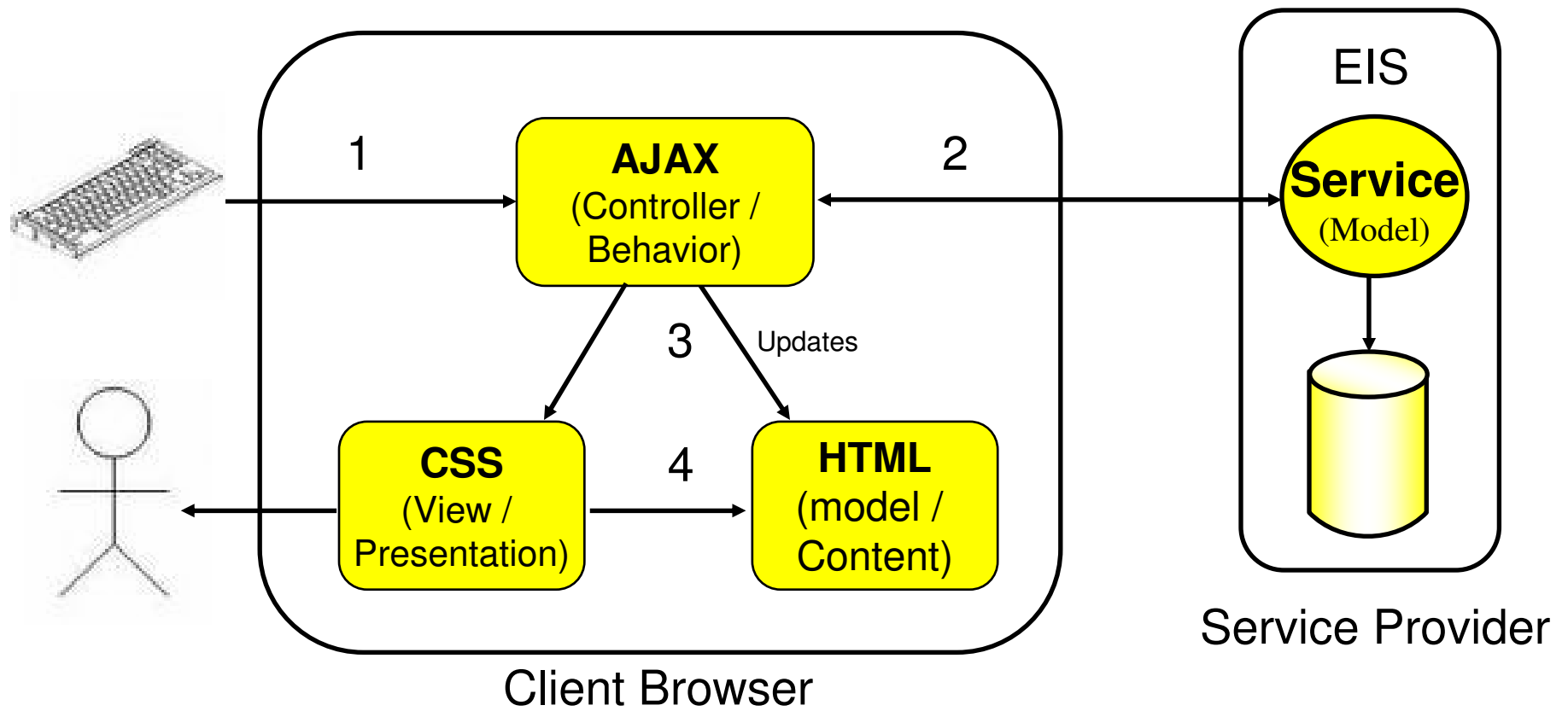


Web 2.0 Architecture

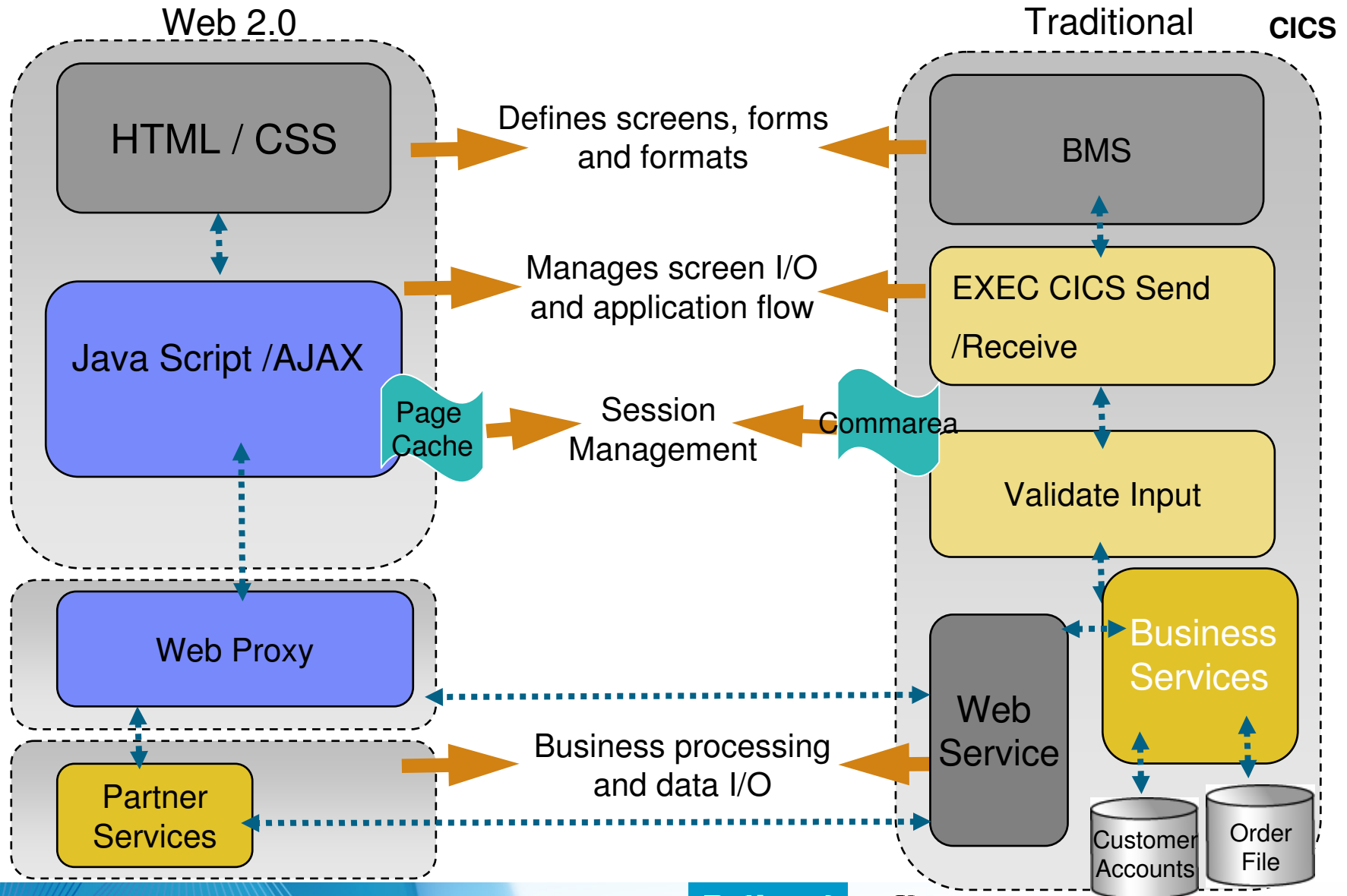


Web 2.0 Application Model



- User enters data on a page. The new data is formatted into a service call, and sent to the service provider. The response is returned to the client and the page is updated.



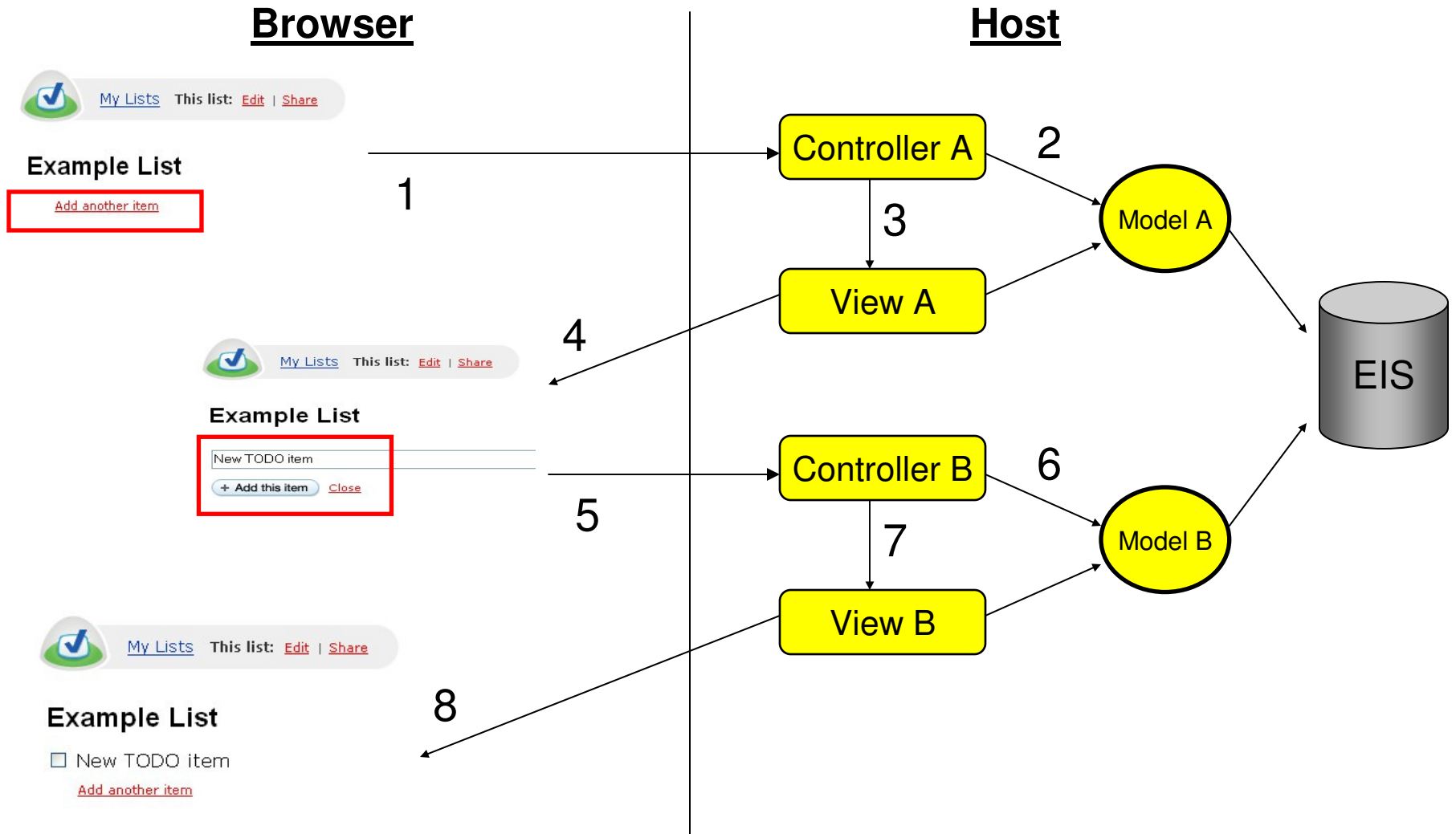
It's not that different



What is RichUI?

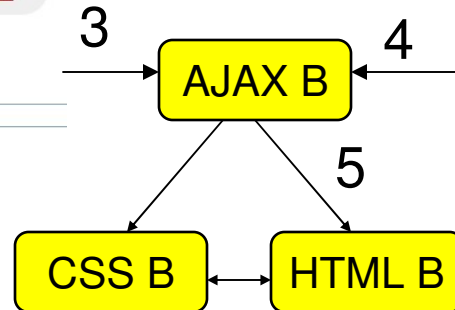
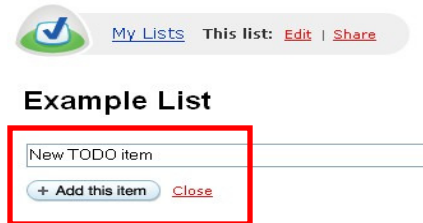
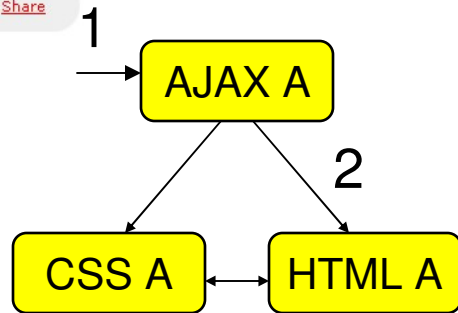
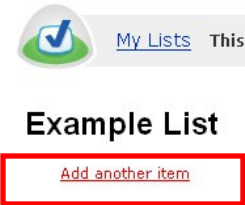
- **RichUI stands for Rich User Interface.**
 - This is a phrase commonly used when talking about an interface that provides dynamic rendering of its individual parts – notably, on the client-browser, as opposed to server-side processing
- **It is a technology that will allow developers of any background to create rich web pages like one would see on a leading-edge, interactive site, such as:**
 - www.digg.com 
 - www.hulu.com 
- **You might also have heard the term: “Rich Internet Application” (RIA)– which is often used synonymously with RichUI.**
- **The benefits of RichUI or RIA include:**
 - Improved user-responsiveness
 - The most successful RichUI implementations can achieve almost a “Windows-desktop” look and feel to users
 - “Rich-er” functionality – beyond the simple rendering of HTML, to include dynamic widgets and components
 - Improved browser/server load-balancing – as more of the business functionality can be distributed to the desktop (browsers)

Web 1.0 Example – TODO list



Web 2.0 (Rich UI) Example – TODO list

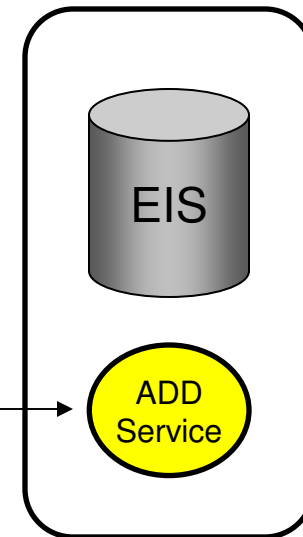
Browser

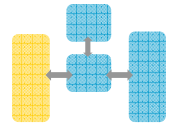


Example List

- New TODO item
- [Add another item](#)

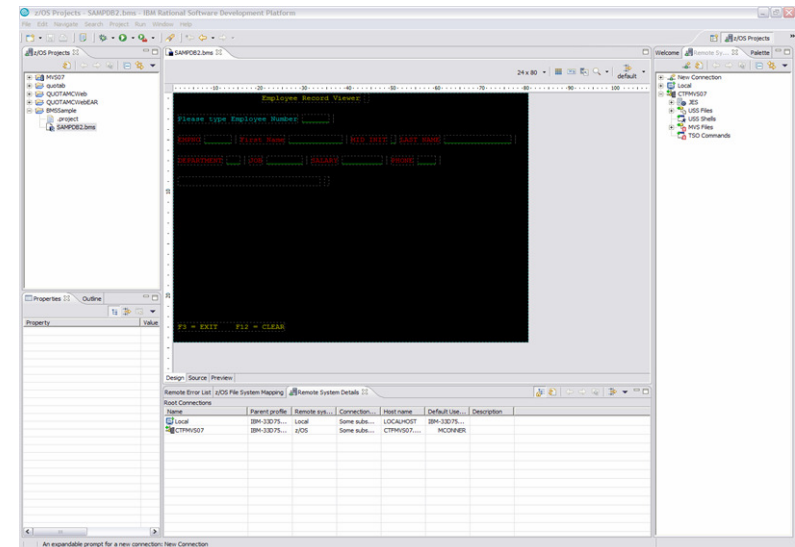
Host



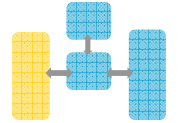


What is HTML?

- **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 RDz 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.
 - RDz provides similar support for HTML screens



HTML vs BMS



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.

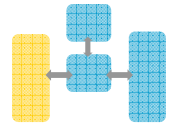
```
<HEAD>
<%@ taglib uri="http://java.sun.com/jsp/core" prefix="f"%>
<%@ page language="java" contentType="text/html; charset=CP1252"
pageEncoding="CP1252"%>
<META http-equiv="Content-Type" content="text/html; charset=CP1252">
<META name="GENERATOR" content="IBM Software Development Platform">
<META 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.emponumininput}" required="false"
style="color: #00ff00" size="6" id="emponumininput"></h:inputText></td>
<td>&nbsp;</td>
<td colspan="44">&nbsp;</td>
<tr><td colspan="80">&nbsp;</td>
<tr><td>&nbsp;</td>
```



What are Cascading Style Sheets (CSS)

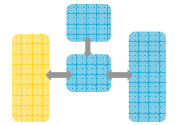
- Define colors, fonts, and formatting in a modular manner
- Able to include on tags, individual (named) elements, or classes of element
 - Flow hierarchically
- Dynamically change style via JavaScript

```
h1 {  
  border: 1px dashed red;  
}  
.clazz {  
  font-family: arial, helvetica;  
  font-size: 12pt;  
  margin-left: 10px;  
  color: green  
}  
#named {  
  font-style: italic;  
  font-family: Courier;  
  background: lightgrey;  
  border: none;  
}
```

Default

using a class

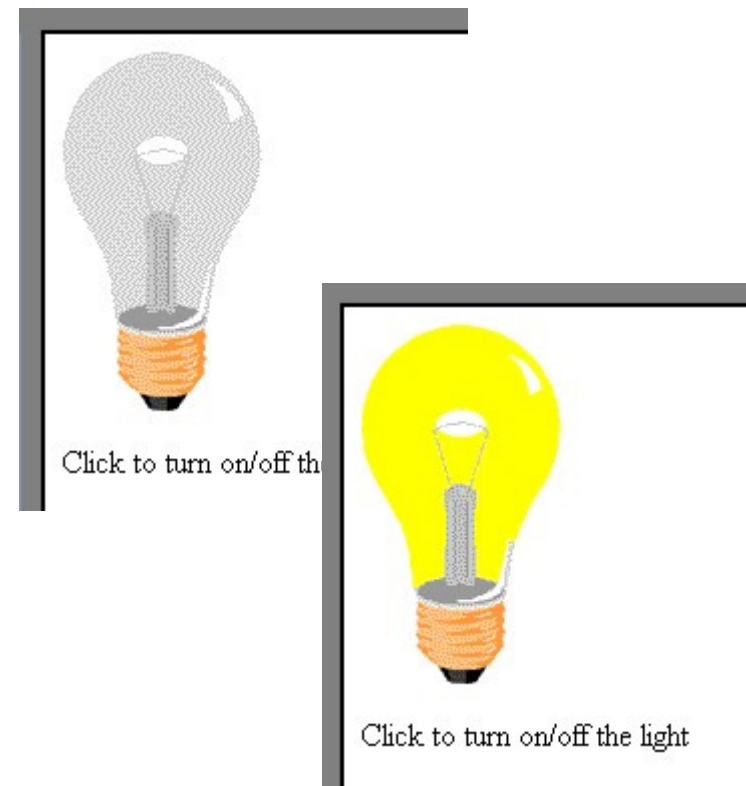
Named Element

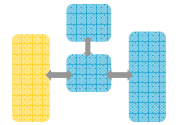


What is JavaScript?

- JavaScript is a scripting language for the web
- JavaScript is a script, not Java
- Small bits of JavaScript interact with HTML and CSS
 - provides interactivity
 - allows dynamic changes
 - completely within the browser

```
function changeimage() {  
  if (cc==0){  
    cc=1;  
    document.getElementById('myimage').src="bulbon.gif";  
  } else {  
    cc=0;  
    document.getElementById('myimage').src="bulboff.gif";  
  }  
}
```





What is AJAX

- **AJAX – Stands for *Asynchronous JavaScript and XML*.** RichUI makes extensive (almost wholesale) use of AJAX, utilizing it whenever it makes a service call. RichUI never executes a traditional HTML Form Submit.
- **JSON – *JavaScript Object Notation* is lightweight format used by JavaScript to exchange data.** JSON is able to serialize structured data, such as arrays, and exchange it among host and client machines.
- **DOJO – An open source JavaScript toolkit.** The DOJO project sets out to create widgets using only JavaScript. RichUI is able to interface with DOJO code in order to pull in some of their widgets.

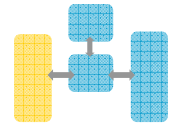
– <http://dojotoolkit.org/>

```

{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    "212 555-1234",
    "646 555-4567"
  ]
}

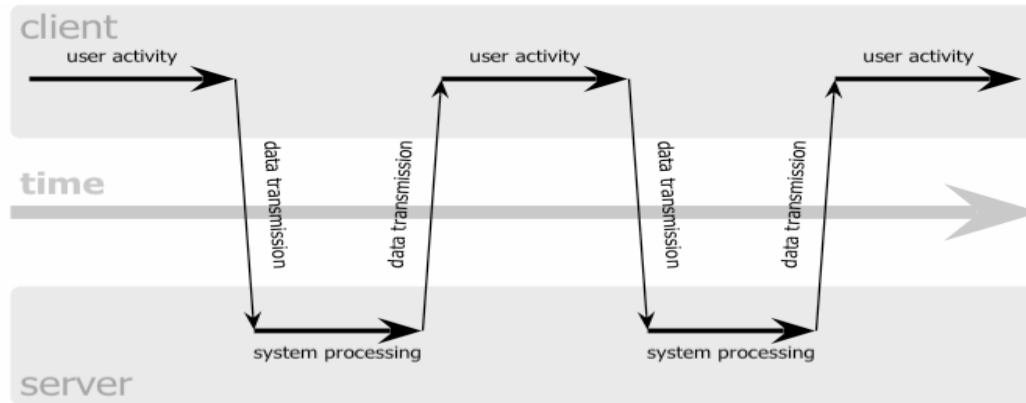
```

Column 0	Column 1	Column 2	Column 3	Column 4
Apples	Oranges	Mangos	Eat lots of fruit	\$2.50
One	Two	Three	Four	Five
Apples	Oranges	Mangos	Eat lots of fruit	\$2.50
One	Two	Three	Four	Five
Apples	Oranges	Mangos	Eat lots of fruit	\$2.50

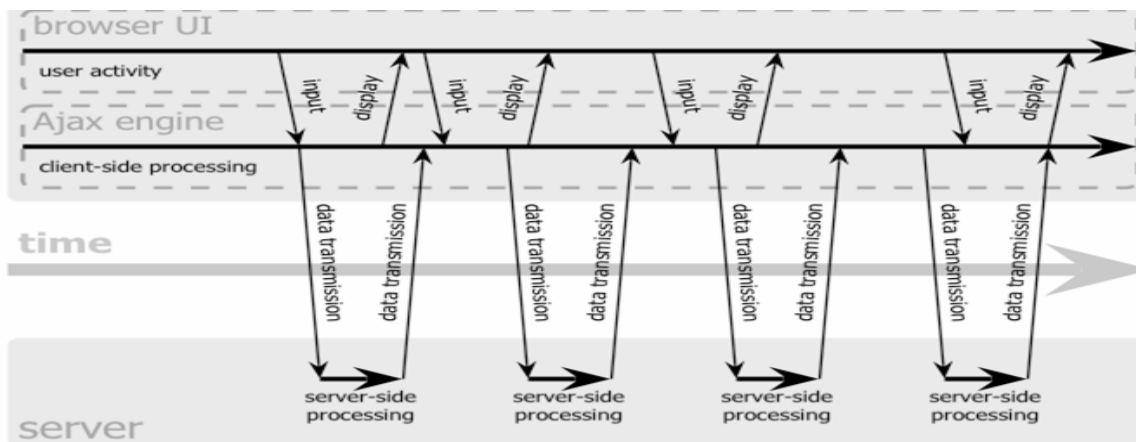


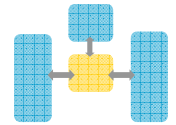
Request Response with Ajax

Classic web application model (synchronous)

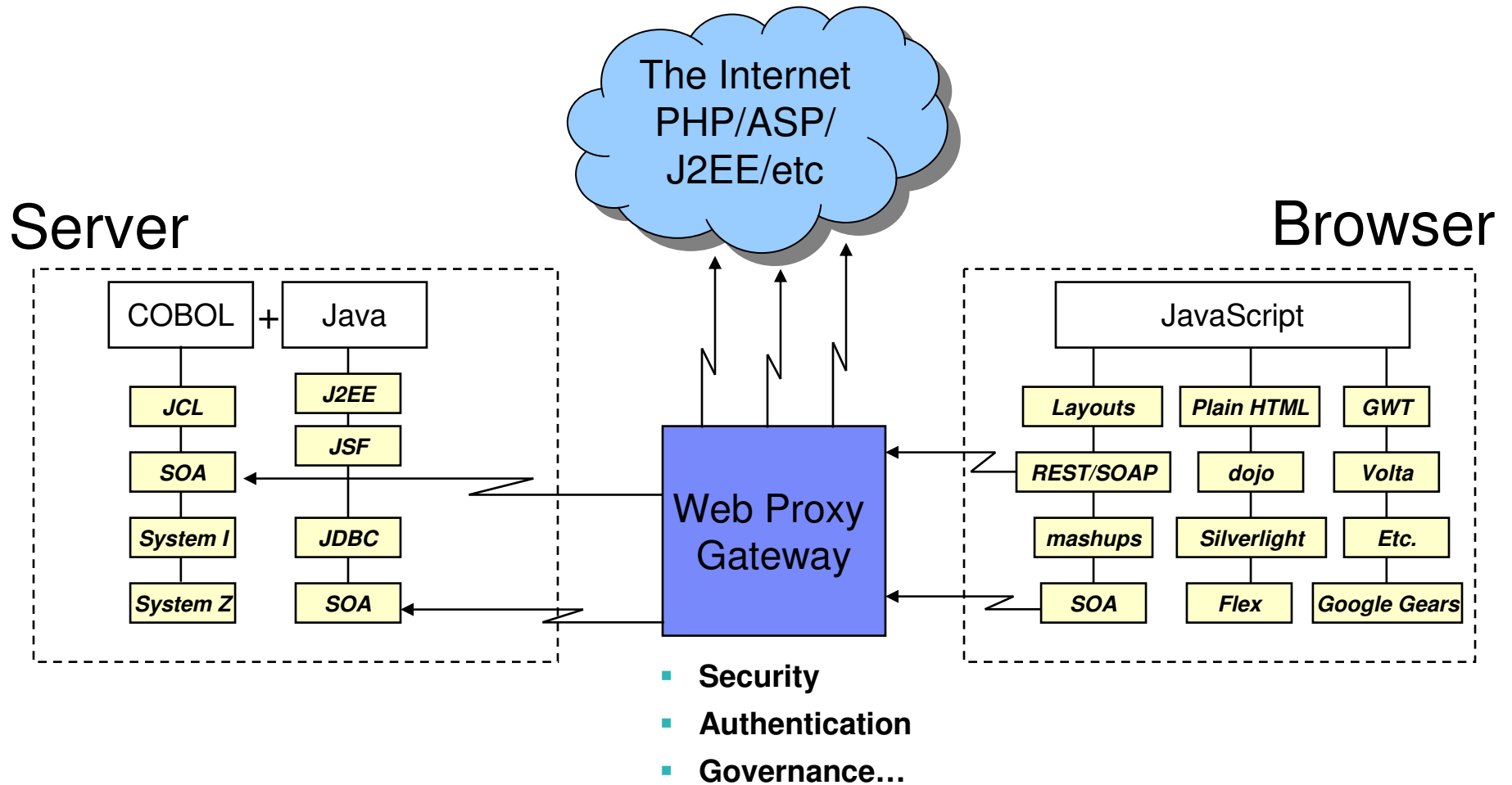


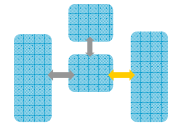
Ajax web application model (asynchronous)





Web Proxy





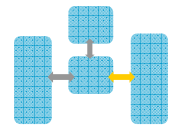
SOA and Web Services

- **Rich U.I. makes extensive use of services, and SOA – Service Oriented Architecture, which is a way to modularize and deploy code so that it can be consumed anywhere in the world using any language.**

- **There are two types of Web Service calls used by Rich U.I.**
 - 1. SOAP service calls** - A type of service call that is more popular in enterprise. It requires the exchange of XML messages between the client and host system.
 - 2. RESTful service calls** – A call made through the HTTP service-interface. Once the call is made, a result is passed back to the requestor in XML or JSON format.

- **By utilizing web services, one can create a truly scalable, flexible, and decoupled system.**

Web Services



Architecture for

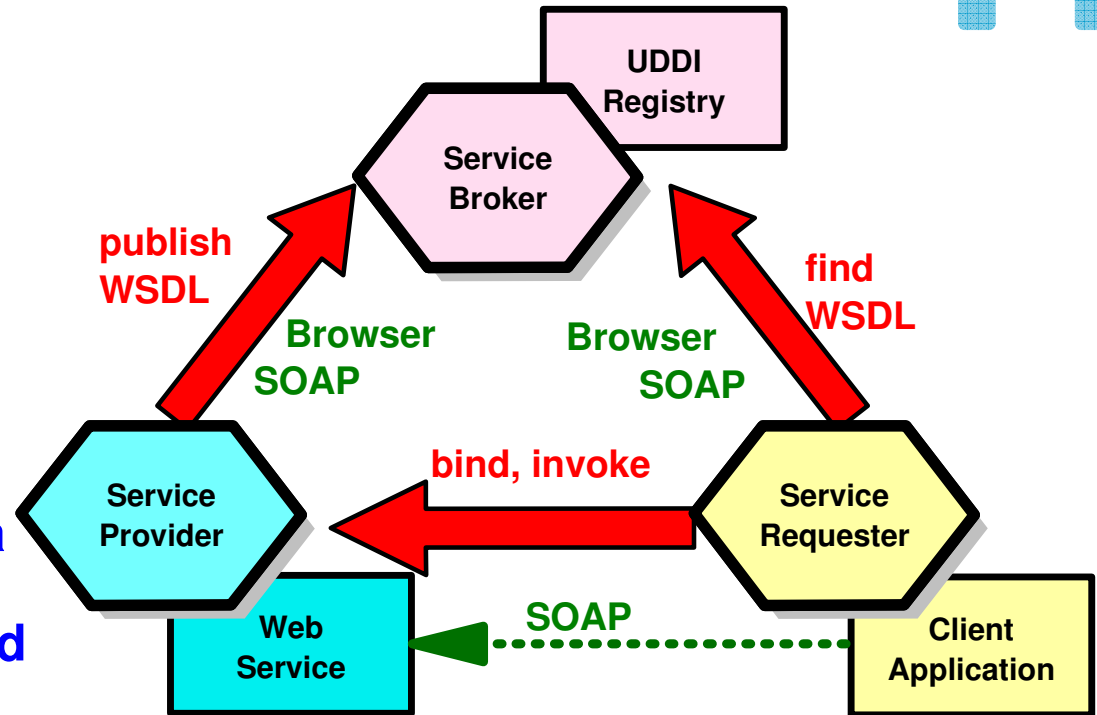
- Application to application
 - Communication
 - Interoperation

Definition:

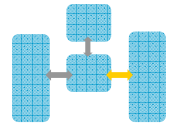
- Web Services are **software components described via WSDL** that are capable of being accessed via **standard** network protocols such as SOAP over HTTP

WS-I.org (Web Services Interoperability Organization)

- Ensure interoperability

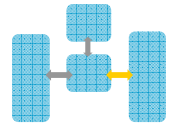


The entire industry is agreeing on one set of standards !!



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

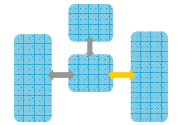


XML Basic Parts

```

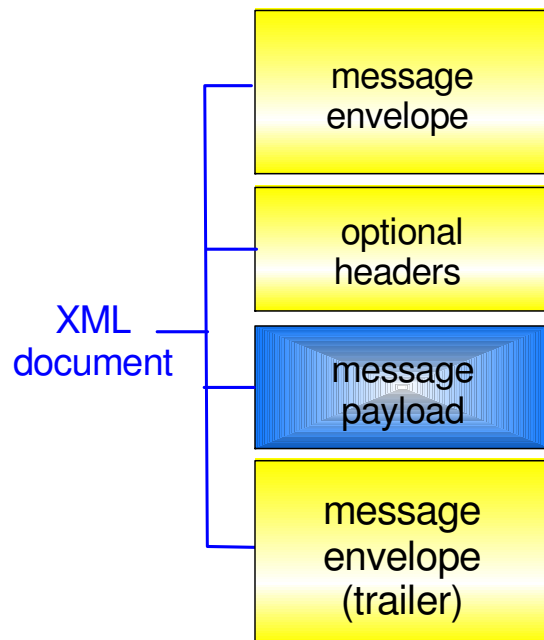
<?xml version="1.0" standalone="no" encoding="UTF-8" ?> ← XML Declaration
<!DOCTYPE shirt SYSTEM "http://shirts.com/xml/dtds/shirt.dtd"> ← Document
<shirt>                                                    ← root element
  <model>CICS Tee</model>                                  ← child of root
  <brand>Tommy 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>

```

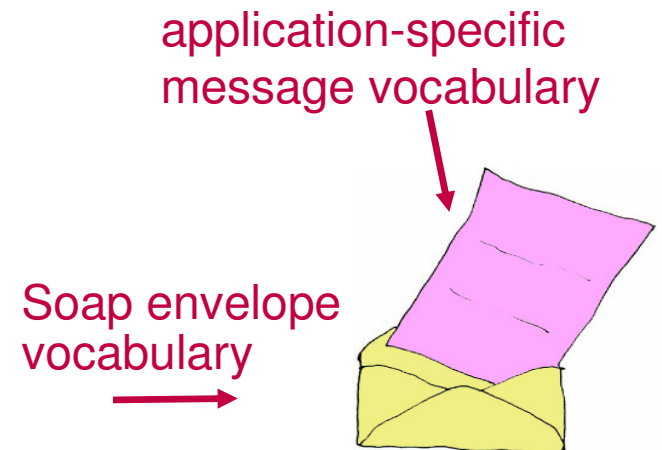
Simple Object Access Protocol (SOAP)

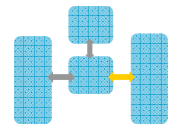
- An XML-based protocol for exchanging of information in a decentralized, distributed environment
- An open standard whose main goal is to facilitate interoperability
- A protocol which is not tied to any operating system, transport protocol, programming language, or component technology



- **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 Example

Request...

```

<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>
    
```

`<m:GetLastTradePrice xmlns:m="Some-URI">
 <symbol>IBM</symbol>
 </m:GetLastTradePrice>` **app-specific message payload**

SOAP envelope

Response...

```

<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"

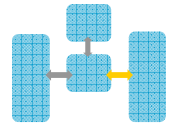
    <SOAP-ENV:Body>
        <m:GetLastTradePriceResponse xmlns:m="Some-URI">
            <Price>134</Price>
        </m:GetLastTradePriceResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
    
```

Result returned in Body

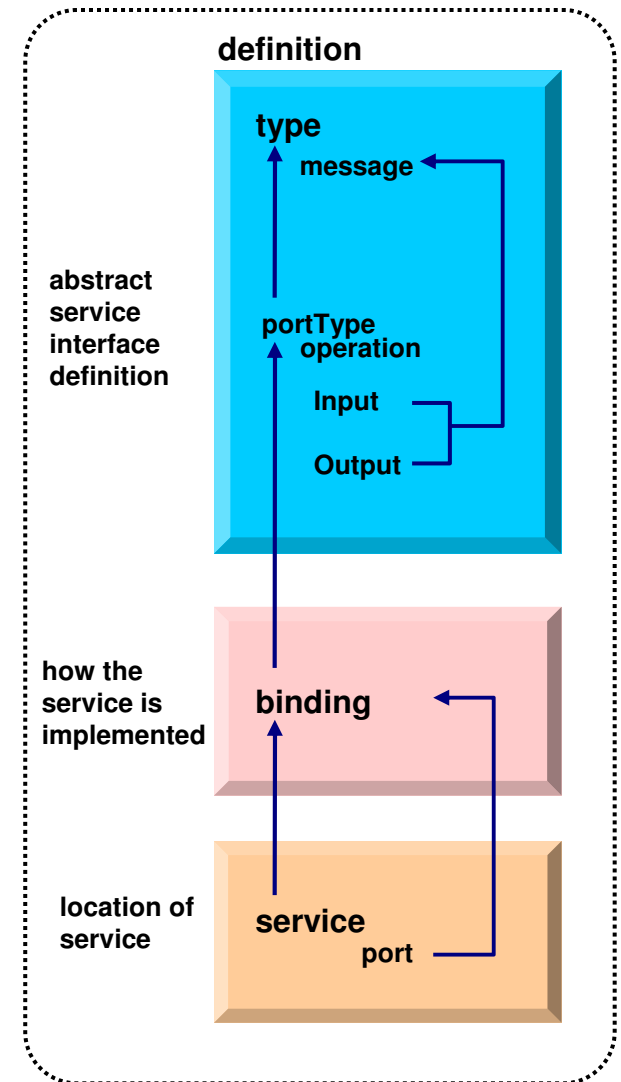
`<m:GetLastTradePriceResponse xmlns:m="Some-URI">
 <Price>134</Price>
 </m:GetLastTradePriceResponse>` **app-specific message payload**

SOAP envelope

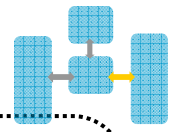
WSDL - Web Service Description Language



- 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**

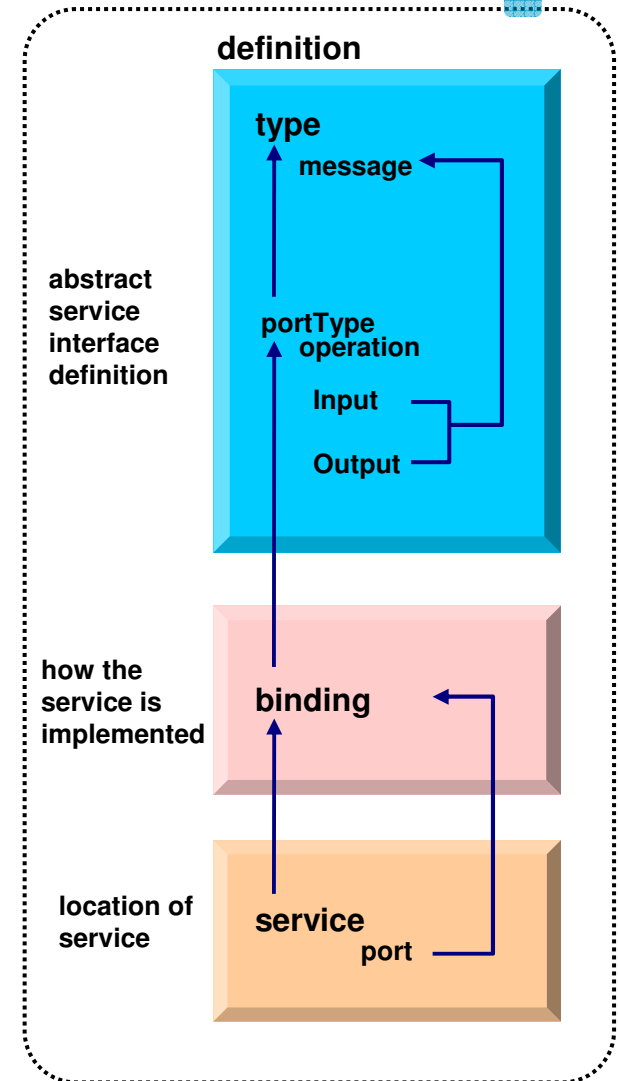
- Operation (business functions)
 - Input Message (0 or 1) and Output Message (0 or 1)
 - 1 or more parts
 - Parts may be simple or complex
 - Complex parts may have multiple elements

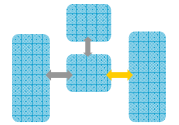
- **Service binding**

- Definition of the physical service interface implementation

- **Service Implementation**

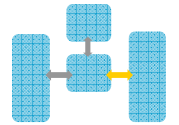
- Location of the service





RESTful services

- **Representational State Transfer**
 - Nouns (URLs) indicate what is being worked on
 - Verbs (GET, PUT, POST, DELETE) indicate the action to perform
- **A lighter-weight, simpler mechanism than Web Services**
 - SSL vs WS-Security
 - HTTP vs. WS-Authentication
 - Unformatted data vs. XML Schema
- **Standards are well defined by IETF**
- ***** Technology stack is in place in the browser**



REST Example

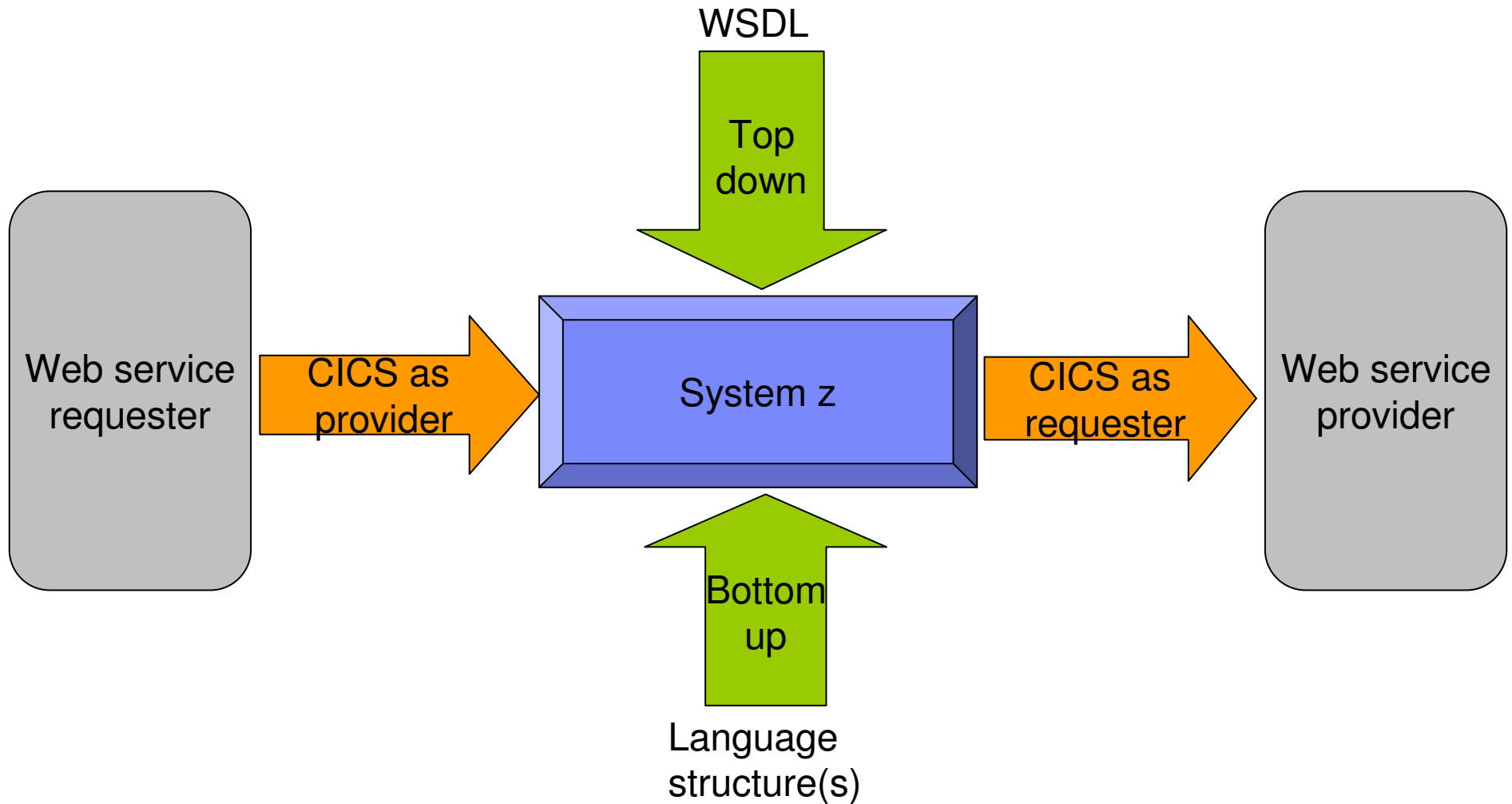
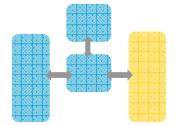
■ Request...

```
GET /mortgage/231677 HTTP/1.1  
Host: www.example.com  
Accept-Language: en  
Charset: UTF-8
```

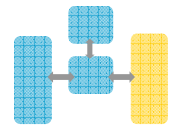
■ Response...

```
HTTP/1.1 200 OK  
Language: en_us  
Charset: UTF-8  
<mortgage>  
  <principle>238000</principle>  
  <rate>3.5</rate>  
  <type>5/1 ARM</type>  
</mortgage>
```

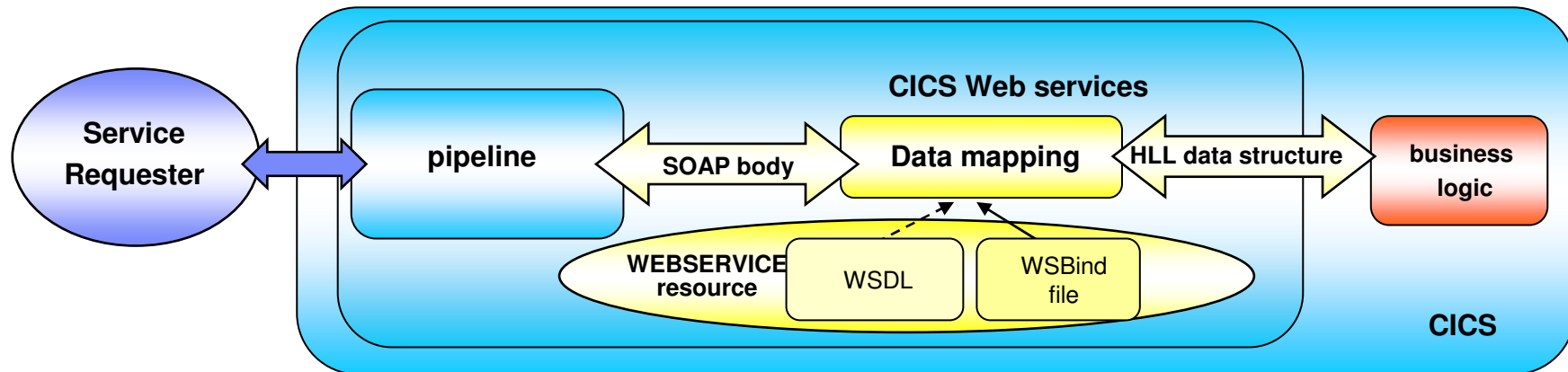
Web Services Enablement Styles



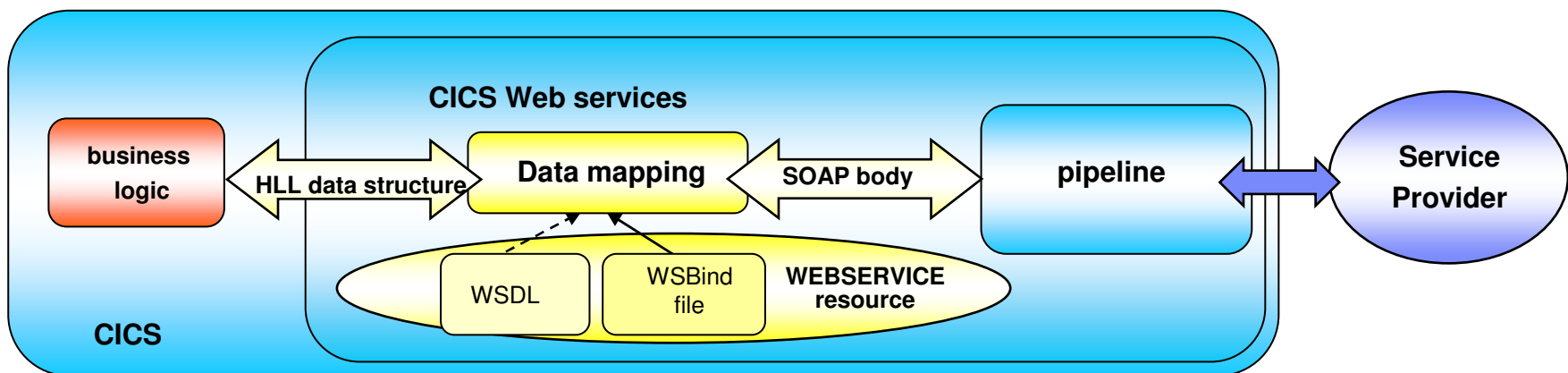
CICS Web Services Implementation



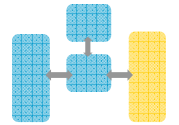
- CICS as a service provider



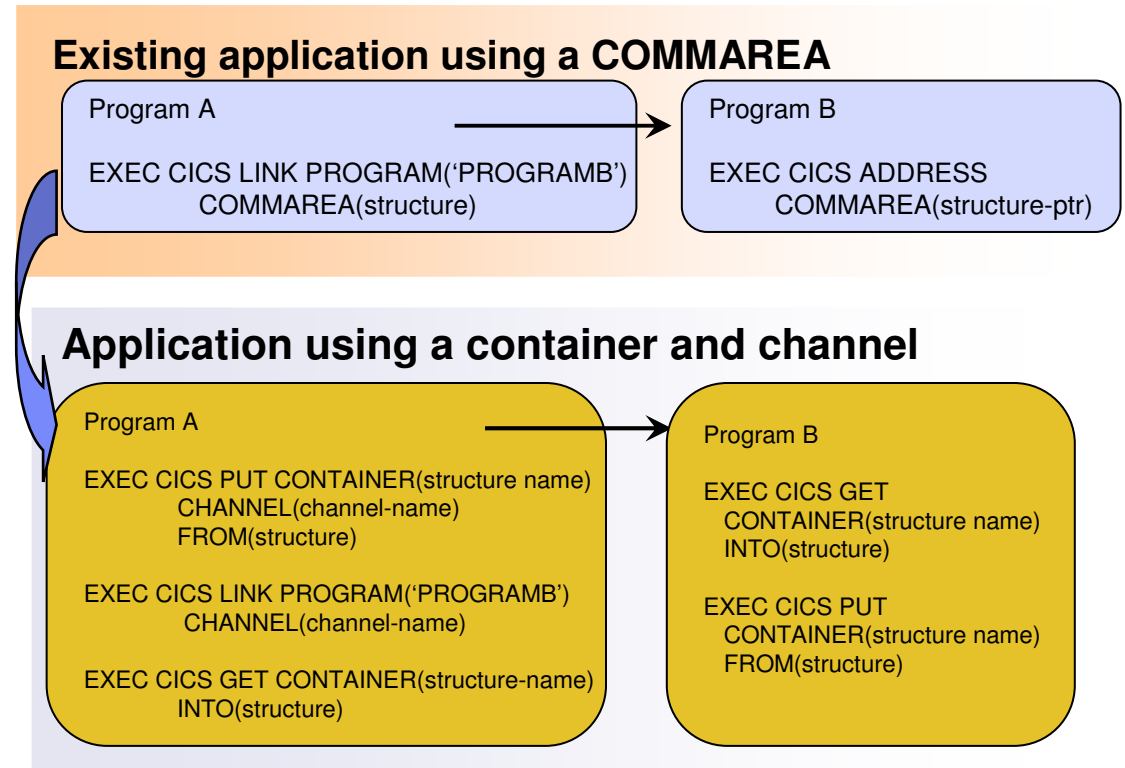
- CICS as a service requester



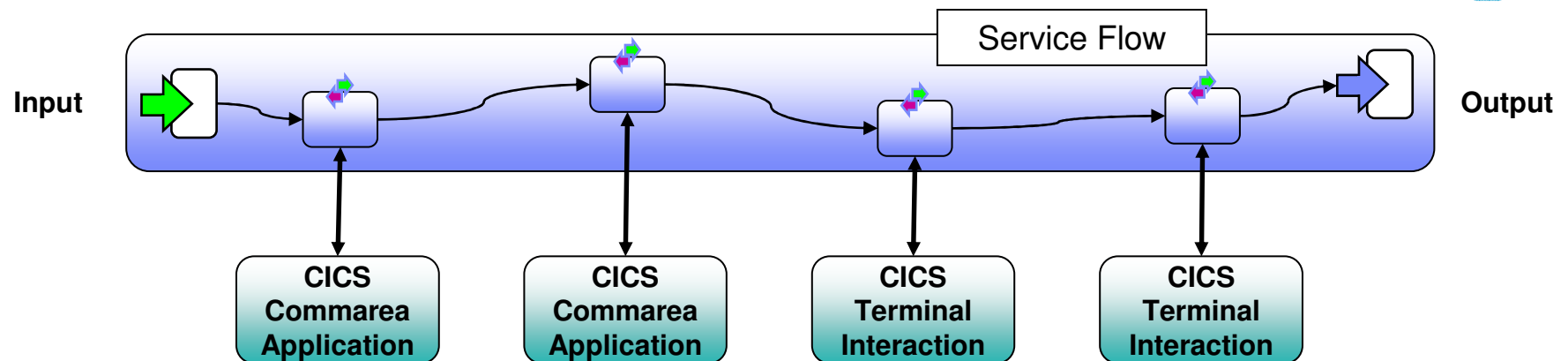
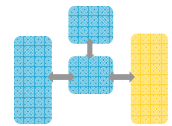
Data Exchange between CICS programs with Containers and Channels



- 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

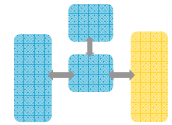


What is a Service Flow?

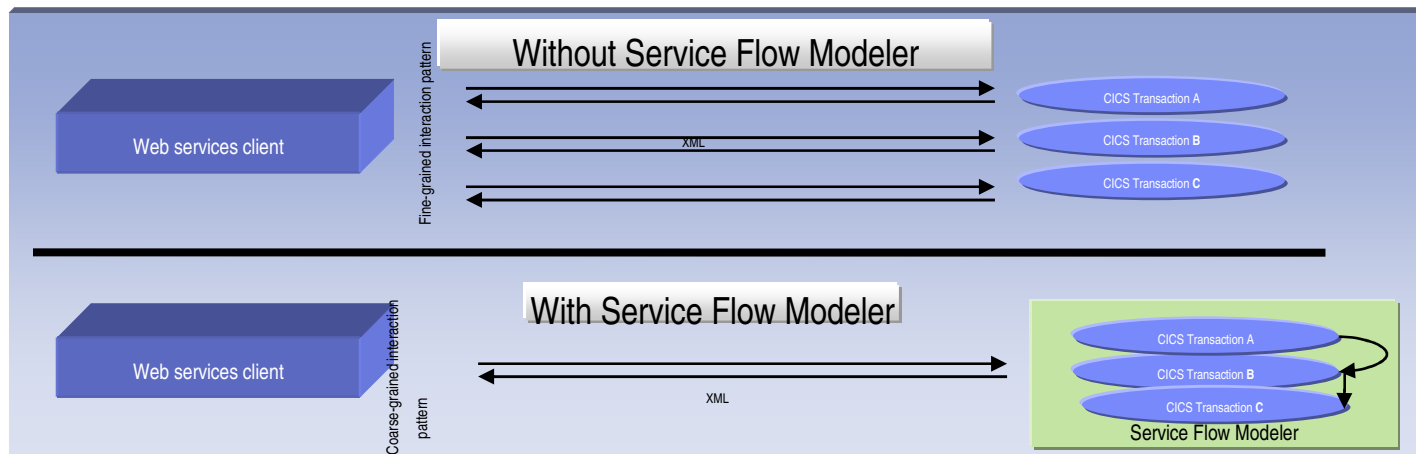
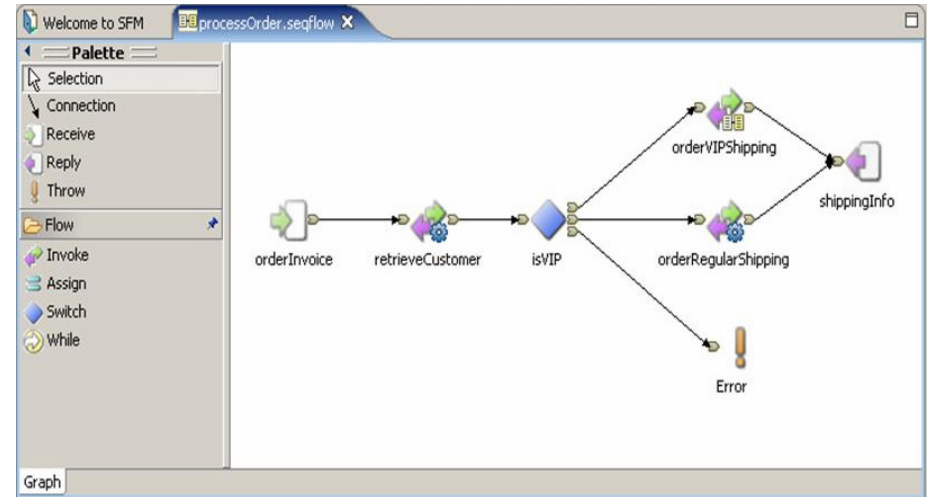


- **A service flow is a non-interruptible micro-flow that is constructed from a collection of nodes**
 - Nodes represent the invocation of CICS resources such as COMMAREA applications or CICS terminal applications
- **The flow describes the navigation path between nodes and allow data mapping between the nodes invocations**
- **A single request may cause the execution of many CICS assets**
- **Allows for the development of reusable aggregated (coarse grained) services from fine grained application resources**
- **Supplied by the CICS Service Flow Runtime – a no-charge, fully supported, orderable feature for CICS TS v3.X**

What is Service Flow Modeler?

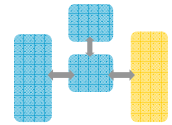


- Service Flow Modeler is the tooling required to develop service flows
- SFM only ships with Rational Developer for System z
- Model, Deploy, and Test Service Flows
 - Aggregates multiple CICS transactions into high-level business processes through visual modeling
 - Supports CICS BMS (terminal-based) applications & CICS commarea/container/channel applications
 - Deploys to the optimized CICS Service Flow Runtime



Enterprise COBOL

CICS/IMS/Batch/DB2 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 TIMES.

XML/
SOAP



XMLParse Document

```
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'
```

XMLGenerate Document

```
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

Summary

- **Web 2.0 applications provide highest levels of visual service**
- **MVC application model provides high levels of flexibility**
- **EGL provides support for all tiers of the Web 2.0 and MVC models**
- **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**