



Oracle E-Business Suite Release 11i

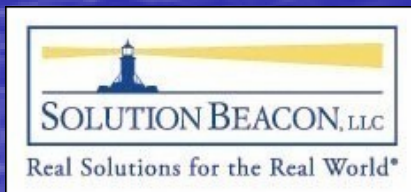
Java in the Oracle Frameworks

Lance Reedy
Solution Architect

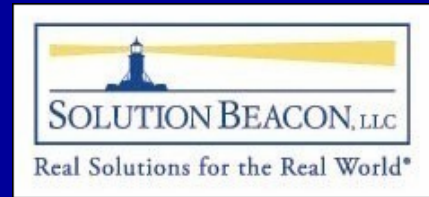
Encore Series-Worcester, MA

Release 11i/12 Workshops

Dallas, TX • Raleigh, NC • San Ramon, CA
Cincinnati, OH • Denver, CO • Atlanta, GA
Detroit, MI • Las Vegas, NV • Worcester, MA



Are you an OAUG Member?



Global Users. Global Solutions.

Member Benefits include:

- ◆ Advocacy opportunities to influence Oracle on product enhancements, usability, new features, Oracle support, pricing and quality
- ◆ Knowledge that showcases the latest trends and techniques used by industry leaders through our national and regional events and our publications, such as OAUG Insight magazine
- ◆ Communication with other OAUG members worldwide through participation in OAUG committees, leadership positions, interaction with Oracle Corporation's user initiatives, frequent member surveys, and Oracle management briefings
- ◆ Education through the hundreds of career-enhancing presentations in our conference paper database archive, as well as discounts to conferences and Oracle education
- ◆ Networking with Oracle customers, industry experts, third-party software firms, and other Oracle Applications specialists through our Member Database and Online Vendor Directory



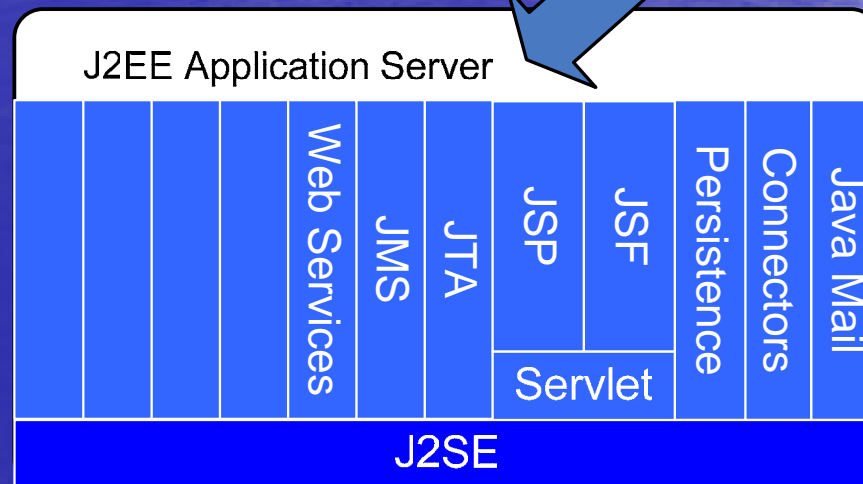
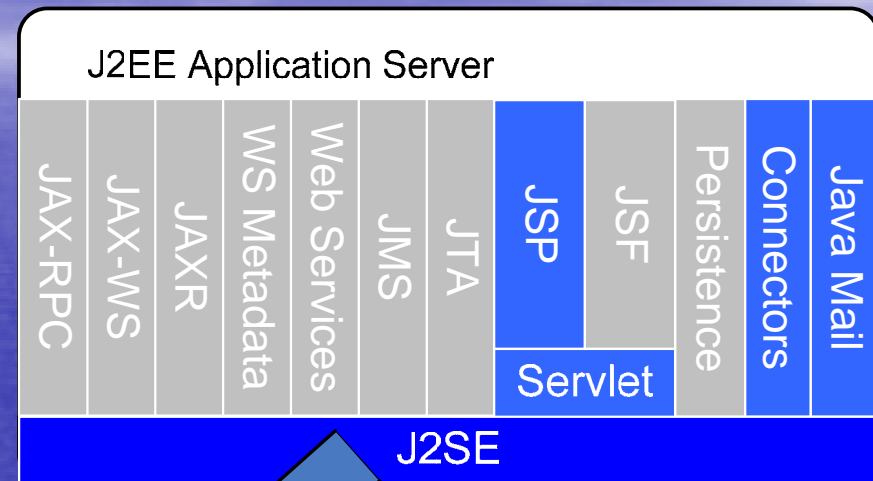
Agenda



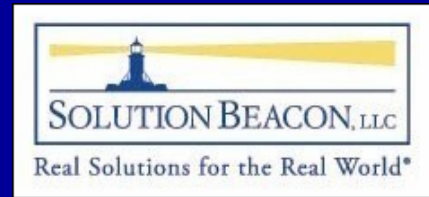
◆ Explore the use of Java technologies in

- OAF
- ADF
- OC4J

◆ Wrap-up



Why Frameworks



◆ Why use a framework?

- Structures code
- Offers a common set of tools
- Promotes a common look and feel
- Facilitates maintenance
- Speeds development

◆ Why J2EE centered frameworks?

- Common skillsets
- Well documented programming models

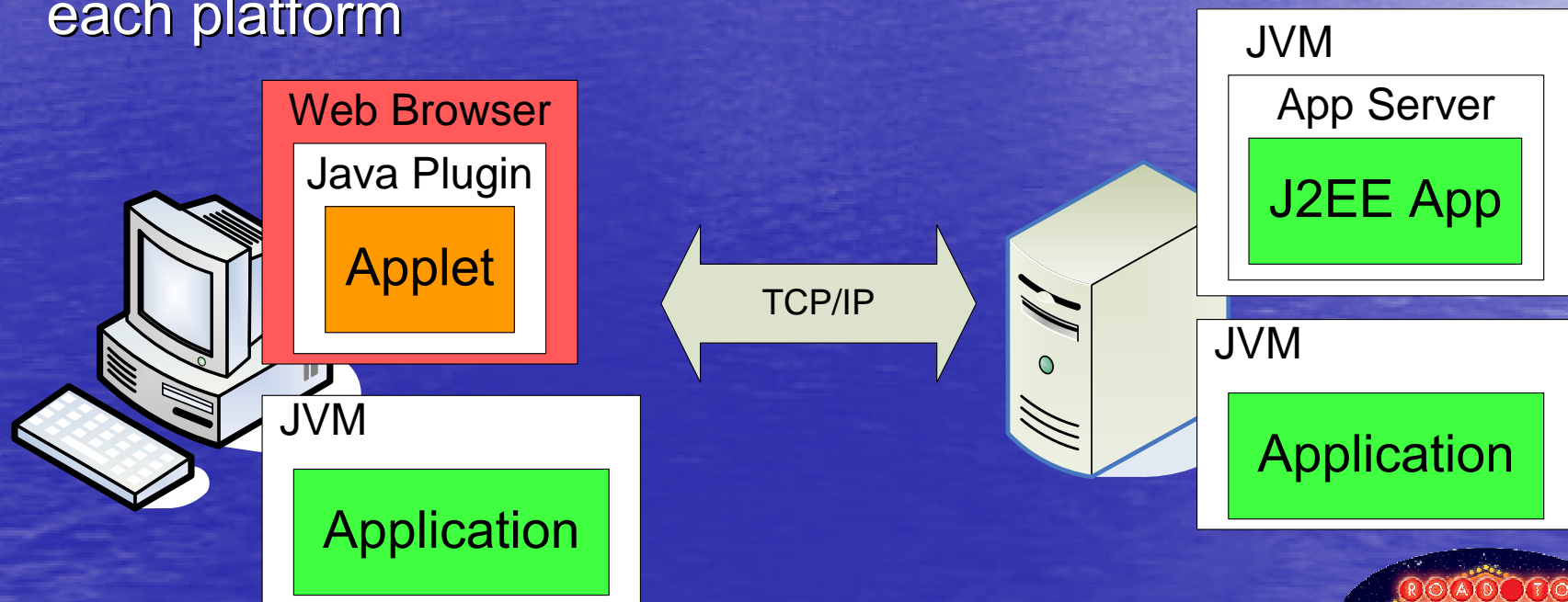
◆ Why Java?



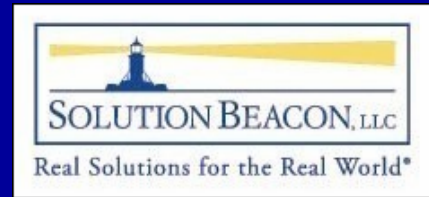
Why Java



- ◆ As a language Java has the flexibility of running on both sides of the client / server model
- ◆ Uses different programming models most appropriate to each platform



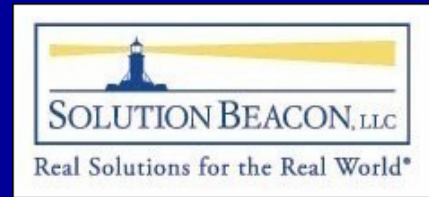
What is Oracle Applications Framework?



- ◆ Based on early web standards and technologies
- ◆ An attempt to introduce abstraction into application design
 - Separation of components into layers
 - Connections only between adjoining layers
- ◆ By separating the display from the business logic change becomes easier
- ◆ Mechanism for personalization, design pattern for extension



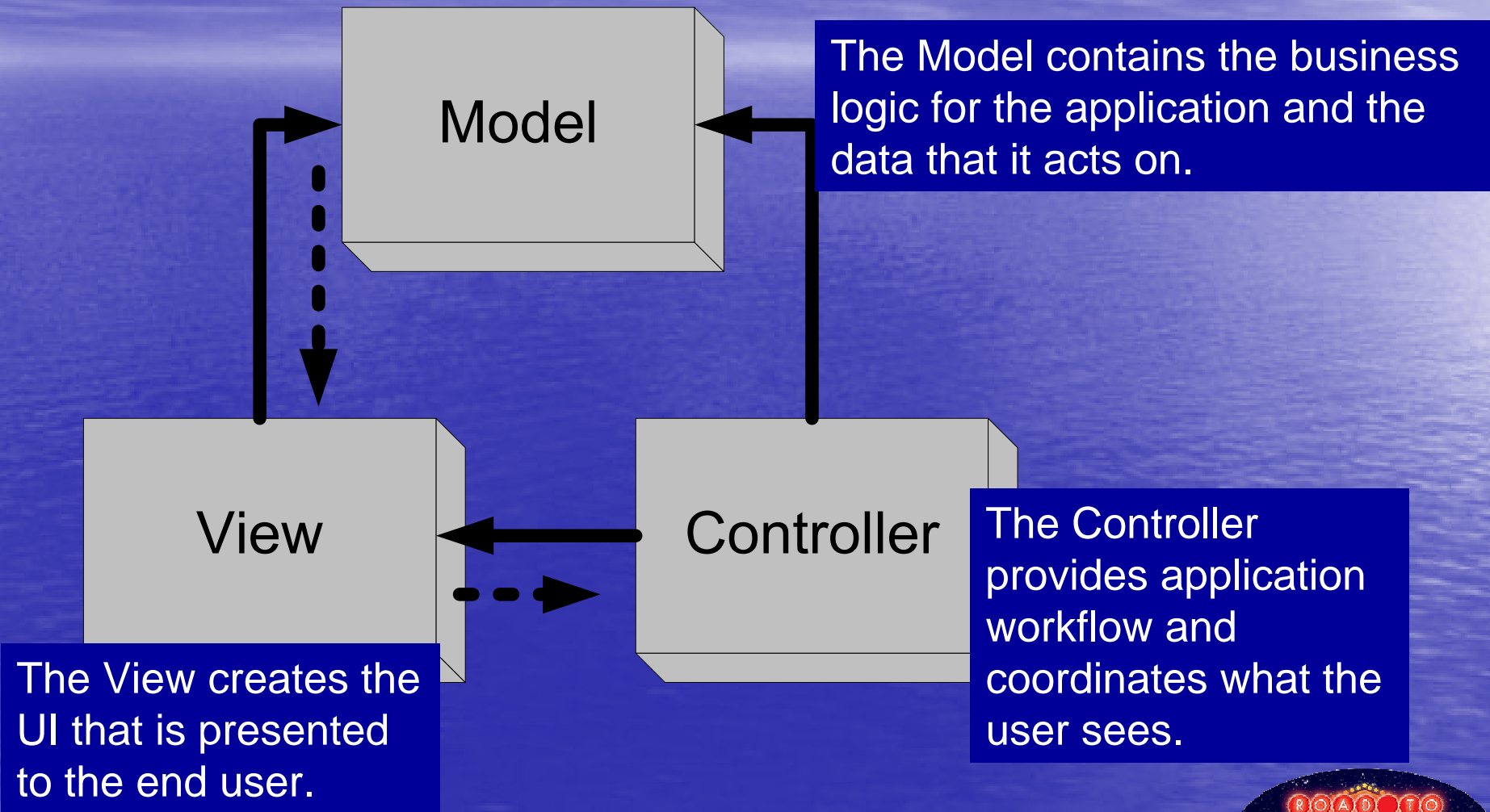
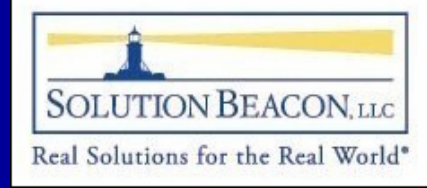
Abstraction



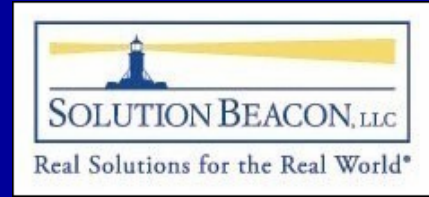
- ◆ The idea of breaking an algorithm into discrete layers, each of which is opaque
 - All calls go from one layer to the adjacent one, never bypassing a layer
- ◆ Common in Object Oriented programming
- ◆ Simplifies code while complicating solutions?!



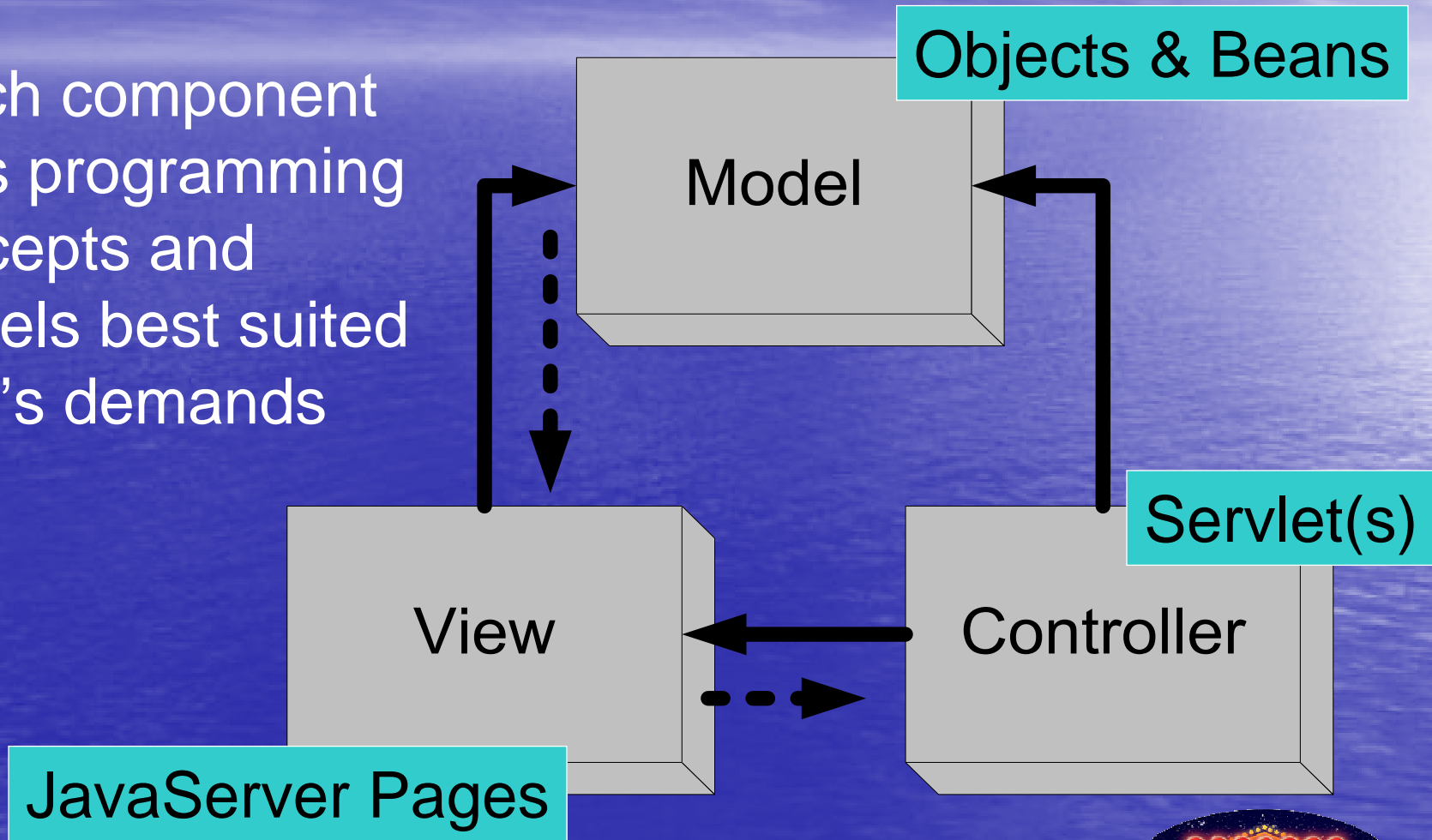
Model View Controller Design Pattern



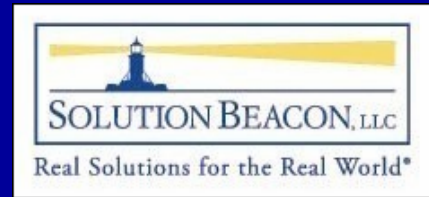
MVC Technologies



- Each component uses programming concepts and models best suited for its demands



Model Details



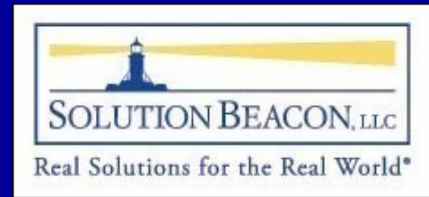
- ◆ A Java class (commonly a JavaBean) is used to represent a business entity that the application acts on

Model

```
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
  
public class NameInputBean {  
    public NameInputBean() {  
    }  
}
```



Creating the View



- ◆ Custom tags are used in the JSP page to reference data presented by, or sent to, the Model.

View

```
<html><table>
  <tr>
    <td align="left">
      <bean:write name="employee"
        property="id"/>
    </td>
```



Controller Details



Controller

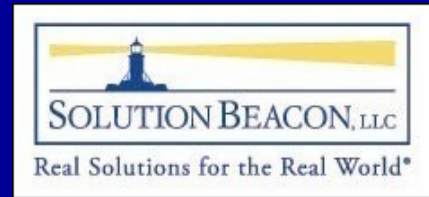
- ◆ A XML config file is used to control behavior of the application
- ◆ A group of Java Classes implement the controller's API

```
<action
  path="/logon"
  type="com.example.helloworld.LogonAction"
  name="userLogonForm"
  scope="request"
  input="/userLogon.jsp"
  >
</action>
```

```
<form-bean
  name="userLogonForm"
  type="com.example.helloworld.LogonAction"/>
```



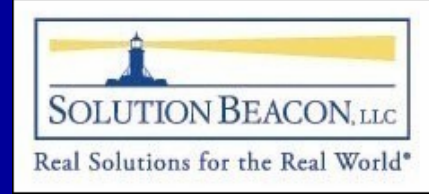
Personalizations



- ◆ Changes to page layout or content visibility
- ◆ Can be done by administrators or by end users
- ◆ The personalizations are applied between the HTML page being generated and it being sent to the client
- ◆ Personalizations are stored via data, not code changes

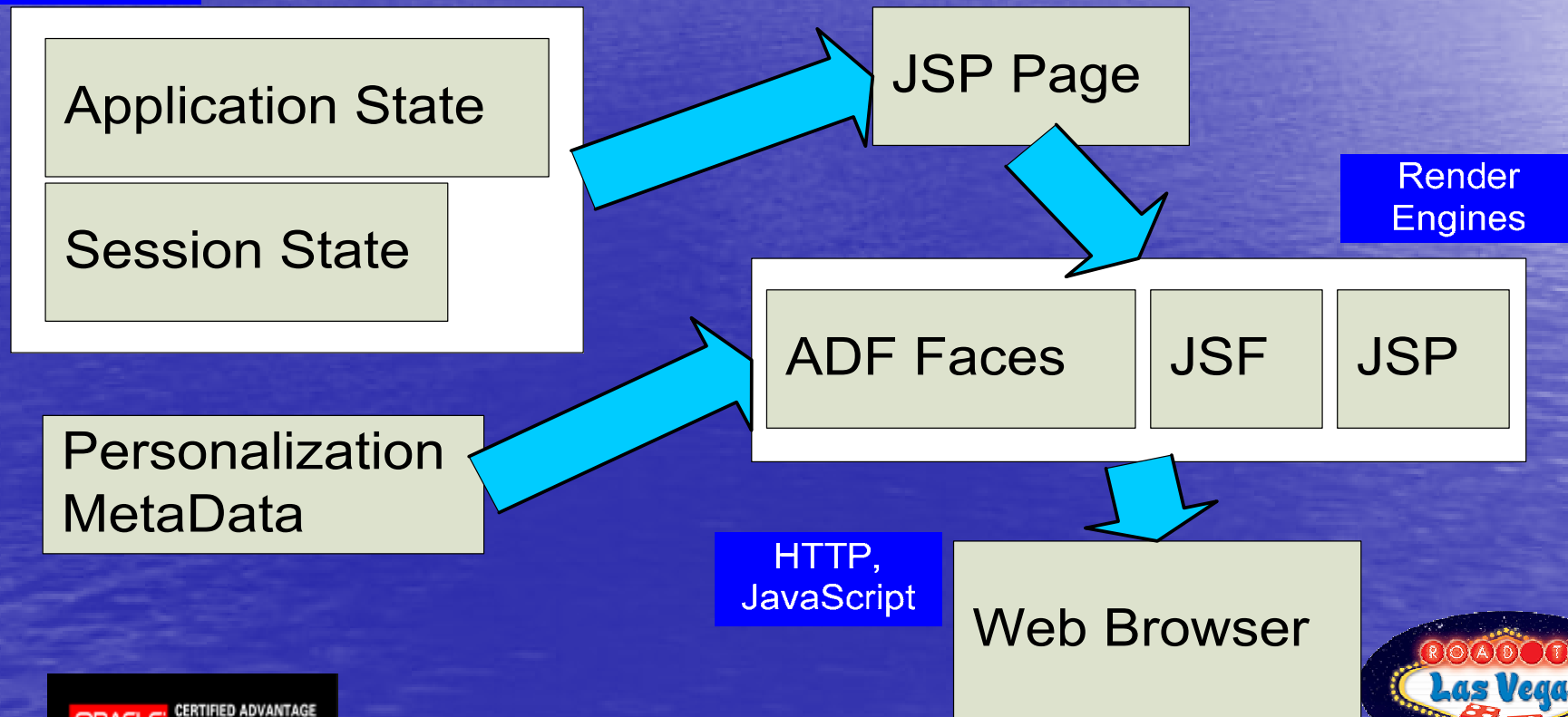


OAF Page Rendering



- ◆ A hierarchy of operations, many of which are driven purely by data

JavaBeans



HTML User Interface

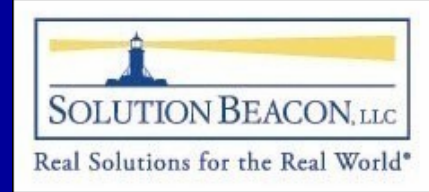


- ◆ Strength is platform independence
- ◆ Weaknesses include
 - Tags and properties used can become too complex and interdependent to manage easily
 - Dependent upon the browser's interpretation of the HTML language

```
<body>  
<p>Example table</p>  
<table cellpadding=1 ...  
<caption>Table 1</caption>  
<tr>  
  <td><font ...>  
    Homer</font></td>  
  <td>cell2</td>  
</tr>  
</table>  
</body>
```



Java Server Page

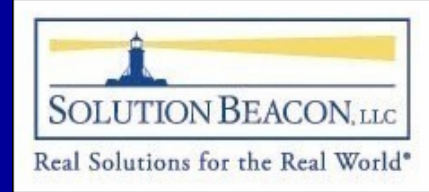


- ◆ A way to create HTML with programmatic content.
- ◆ Embeds Java code (scriptlets) into HTML
- ◆ Strength is creating a maintainable HTML UI
- ◆ Weakness is doing complex code
- ◆ Gets compiled into a Servlet at runtime

```
<body><p>
    A table drawn via a JSP page.
</p><p>
<table cellspacing=1 ...
<caption>Table 1</caption>
<tr>
<td><font ...>
    <%= visitor.getName() %>
</font></td>
<td>cell2</td></tr>
</table>
</body>
```



Servlet

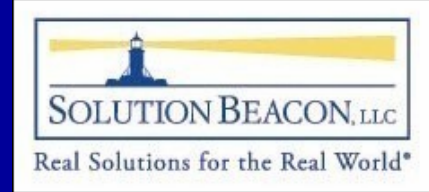


- ◆ Java code that produces HTML as it's output
- ◆ Distributed as compiled Class files
- ◆ Strength is the ability to perform real work
- ◆ Weakness is producing a maintainable HTML UI

```
public class _test1 ... {  
    public void _jspService(... {  
        response.setContentType...  
        __ojsp_s_out.write( "<body>...");  
        __ojsp_s_out.write( visitor.getname() );  
    }  
}
```



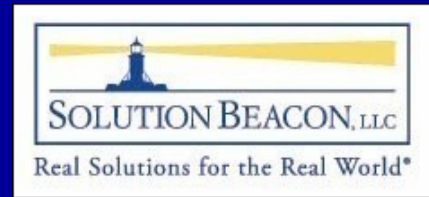
What is Application Developer Framework?



- ◆ The next generation after OAF
- ◆ Builds on OAF, adds
 - More dynamic page refreshing
 - Data Binding
 - Ability to build content for limited devices
 - JavaServer Faces
- ◆ In short, it moves further into the J2EE architecture



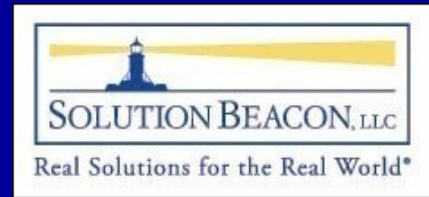
Standards Used by ADF



- ◆ *MVC*
- ◆ *JSP, Servlets*
- ◆ JavaBeans
- ◆ Enterprise JavaBeans 3.0
 - Including new persistence API
- ◆ JavaServer Faces



Oracle Technologies Used



◆ ADF Faces

- UIX widgets

◆ TopLink

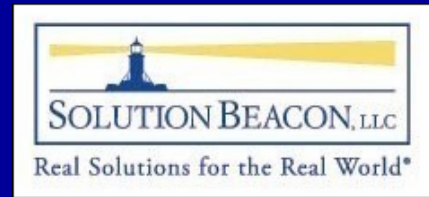
- Data persistence

◆ OC4J

- J2EE application server



JavaBeans and EJB



- ◆ JavaBeans conform to particular coding rules
 - Beans have data fields, each field must be accessed via a Get or a Set method with a matching name
 - They can be loaded dynamically
- ◆ They are used to store data that needs to be used programmatically

JavaBean

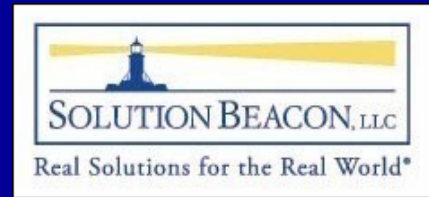
```
Public class Employee {  
    Private String name;  
    Private long id;  
    Private Date hireDate;  
  
    public String GetName()...  
    public void setName(String s)...
```

Enterprise JavaBean

```
public Object persistEntity(Object entity)...
```



Enterprise JavaBeans



- ◆ EJB expands the idea of the basic JavaBean
 - Persistence in a database
 - Remote access
- ◆ Java carries the C/UNIX idea of Remote Procedure Call forward in the form of Remote Method Invocation
 - An EJB container must facilitate the use RMI to allow remote client's to access Enterprise JavaBeans

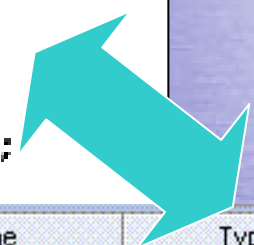


Java / SQL



- ◆ A query mechanism must exist to allow objects to be retrieved from the DB as needed
- ◆ Datatypes must be mapped between languages:
 - String vs varchar
 - Address vs a table

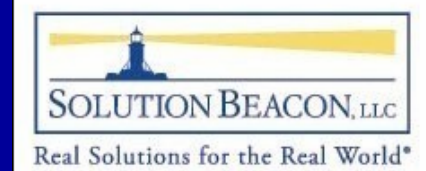
```
6 public class Users {
7     /**...*/
10    private List expertiseAreasColle
11    private Integer userId;
12    private String userRole;
13    private String email;
14    private String firstName;
15    private String lastName;
16    private String
17    private String
18    private String
19    private String
20    private String
```



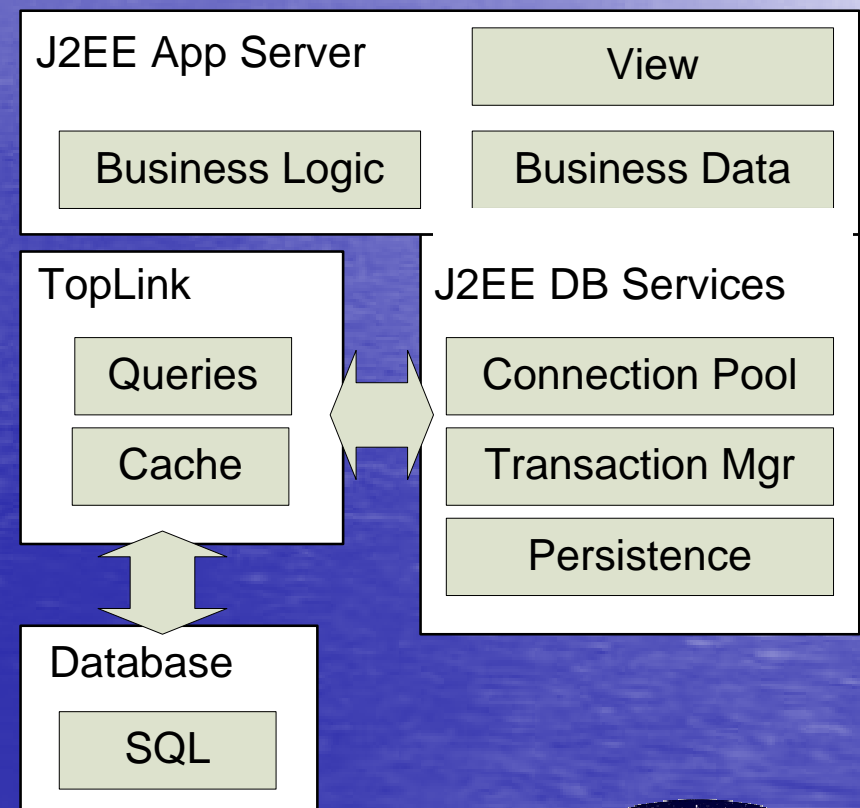
PK	Name	Type
✓	USER_ID	NUMBER(8, 0)
	USER_ROLE	VARCHAR2(10)
✓	EMAIL	VARCHAR2(50)
	FIRST_NAME	VARCHAR2(30)
	LAST_NAME	VARCHAR2(30)
	STREET_ADDRESS	VARCHAR2(40)
	CITY	VARCHAR2(30)
	STATE_PROVINCE	VARCHAR2(25)
	POSTAL_CODE	VARCHAR2(12)
	COUNTRY_ID	CHAR(2)



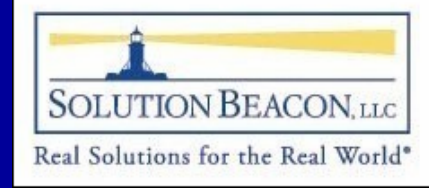
Object Persistence



- ◆ The desire in a J2EE environment is to prevent application code from interacting directly with the database. Embedding SQL in code is evil.
- ◆ This is achieved by using the J2EE App Server to access common DB services



EJB 3.0 Persistence

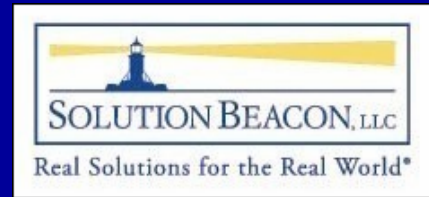


- ◆ Based on annotation
- ◆ Uses a tool to generate the code's skeleton and annotations
- ◆ Uses other tools at compile and runtime to turn the annotations into actual code

```
66 @Entity
67 @Table(name = "EMP")
68 public class Employee implements java.io.Serializable {
69     private int empNo;
70
71     @Id
72     @Column(name = "EMPNO")
73     public int getEmpNo() {
74         return empNo;
75     }
76
77     public void setEmpNo(int empNo) {
78         this.empNo = empNo;
79     }
}
```



TopLink versus EJB 3.0



- ◆ TopLink is an established Oracle standard
- ◆ EJB 3.0's Java Persistence API is an emerging industry wide standard...
 - ... Java Persistence API was added in latest J2EE spec
 - Can be used outside of EJB, by J2SE apps
 - Based on ideas submitted by various J2EE vendors including Oracle



- ◆ JavaServer Faces simplifies the design of HTML UI
 - By providing tags to represent common data elements
 - By providing components which can help to validate data as it's being entered
- ◆ ADF Faces adds tags representing common Oracle UI elements (UIX Widgets)

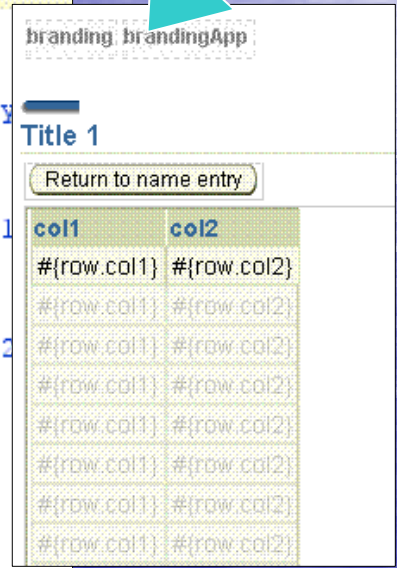
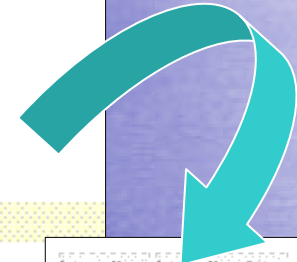


ADF Faces – Declarative UI

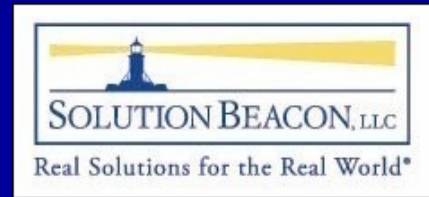


- ◆ Extends the JSP concept to expose Oracle's own UIX widgets
- ◆ Replaces even simple HTML tags with abstract elements

```
<h:form>
  <af:panelPage title="Title 1">
    <f:facet name="menu1"/>
    <f:facet name="menuGlobal"/>
    <f:facet name="branding"/>
    <f:facet name="brandingApp"/>
    <f:facet name="appCopyright"/>
    <f:facet name="appPrivacy"/>
    <f:facet name="appAbout"/>
  </af:panelPage>
  <af:iterator/>
  <af:panelButtonBar>
    <af:commandButton text="Return to name entry"/>
  </af:panelButtonBar>
  <af:table emptyText="No items were found">
    <af:column sortable="false" headerText="col1">
      <af:outputText value="#{row.col1}"/>
    </af:column>
    <af:column sortable="false" headerText="col2">
      <af:outputText value="#{row.col2}"/>
    </af:column>
  </af:table>
</af:panelPage>
</h:form>
</afh:body>
```



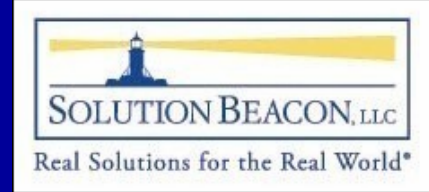
Oracle Containers for Java – J2EE Server



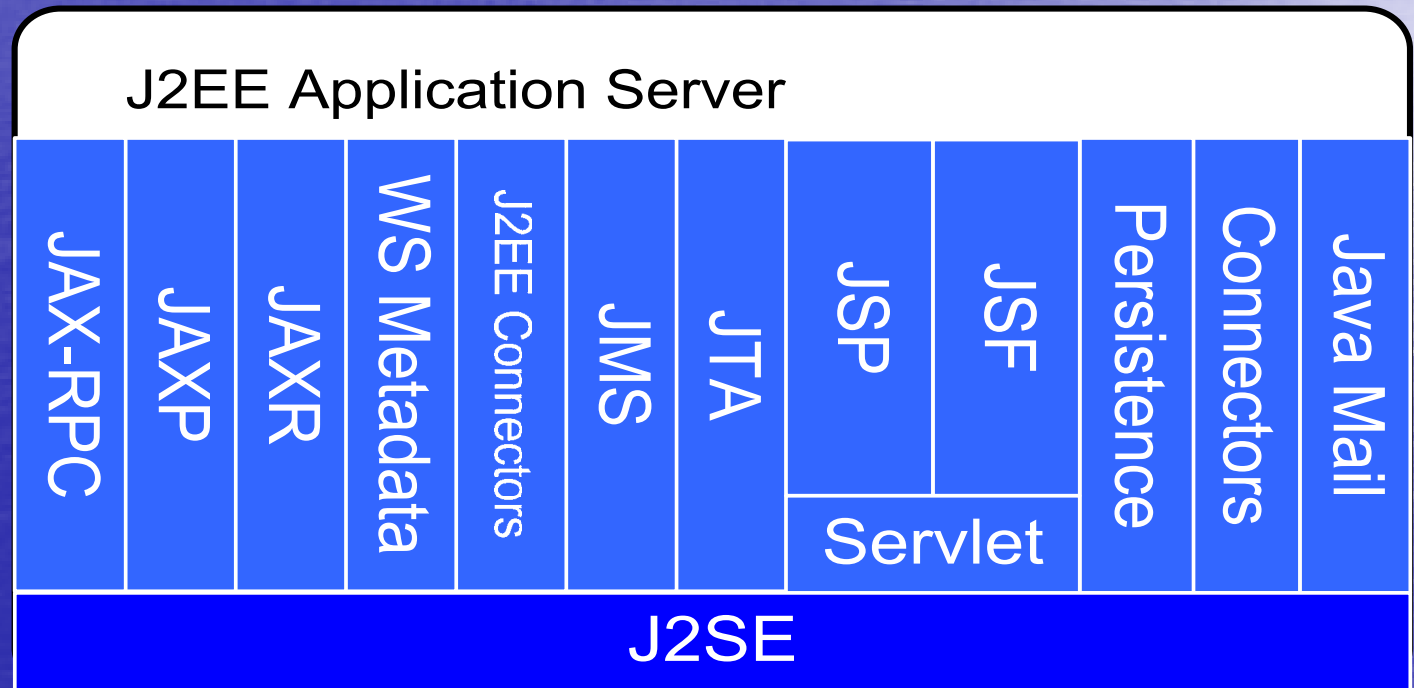
- ◆ J2EE defines a specification for the applications server
- ◆ Defines clear interfaces and mechanisms for interacting with other infrastructure, such as Identity Management (LDAP)
- ◆ Individual vendors select portions of the spec that they will implement
- ◆ Focuses on server side technologies
 - Depends upon HTML or Swing for User Interface



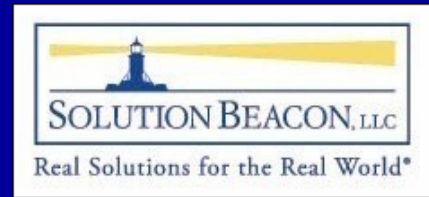
J2EE Technologies



- ◆ A rich suite of APIs built on a common core



Development in J2EE



- ◆ Java is rich with OpenSource solutions
 - J2EE app servers: JBoss, Tomcat, GlassFish, Orion
 - Developer tools: Eclipse, NetBeans
 - Frameworks: Struts, Tapestry, Turbine
 - Persistence: Hibernate, OJB
- ◆ Some of these are sold as commercial applications with vendor support and consulting
- ◆ None of the developer tools offer tight integration into proprietary application servers
- ◆ Few of them offer a complete development solution

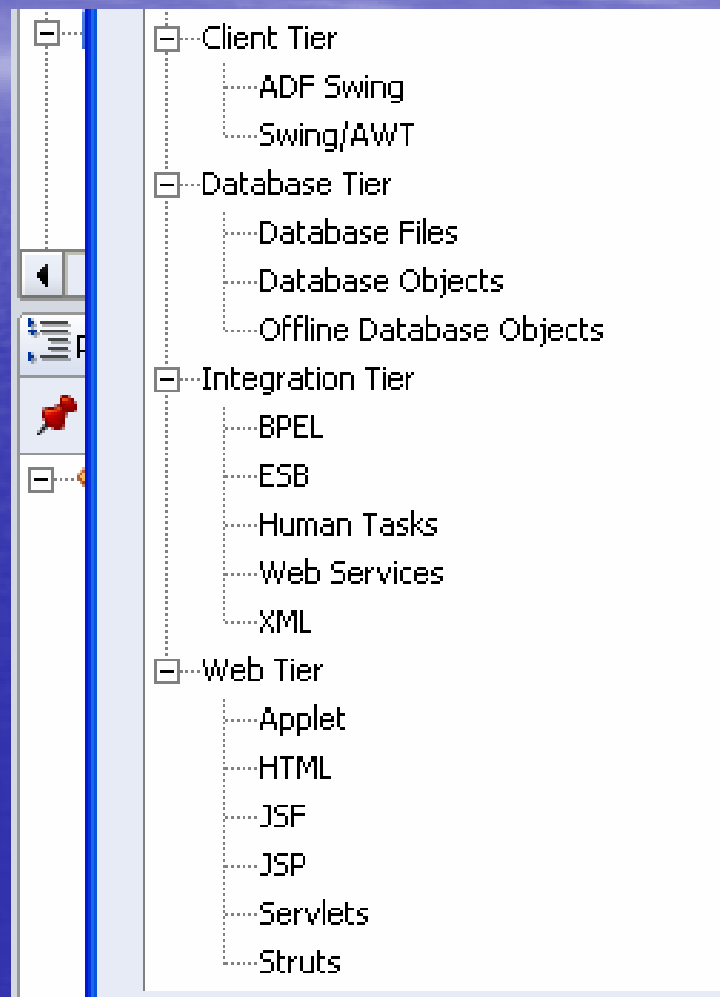
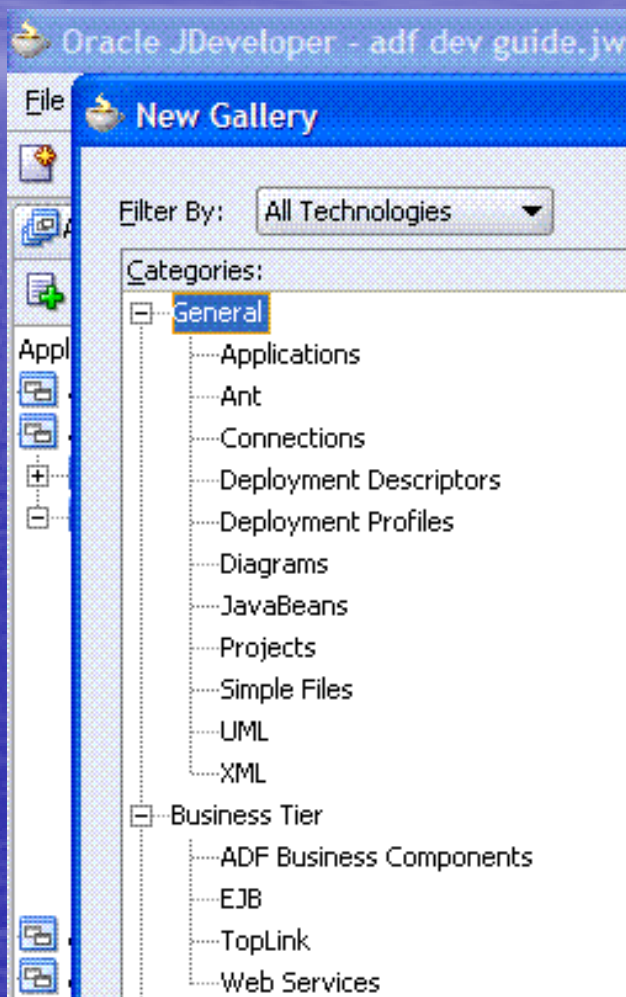


JDeveloper

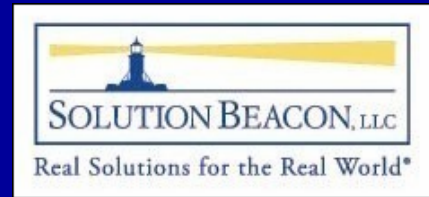


- ◆ Oracle's full featured developer's toolkit
- ◆ Very feature rich for plain Java, J2EE, and Oracle specific development
- ◆ Supports design, coding, testing, packaging, and deployment





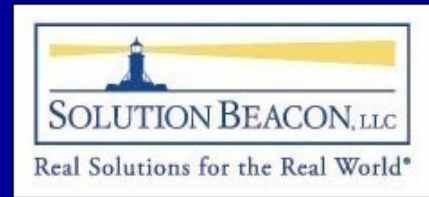
J2EE Application Deployment



- ◆ J2EE applications are composed of a number of components
 - Classes, Servlets
 - HTML, JSP, and images
 - XML configuration files
- ◆ Logical groups of Classes become Java Application Archives (jar) files
- ◆ All content is organized into a specific directory structure
- ◆ Results are combined into a special jar file



WAR File Contents



◆ Web Application Archive

/

/META-INF

/css (custom)

style sheets

/images (custom)

static images

/html (custom)

static HTML pages

/WEB-INF

web.xml, struts.xml

/WEB-INF/classes

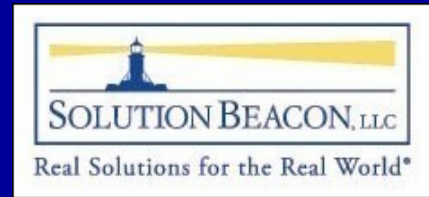
individual Class files

/WEB-INF/lib

Jar files of Classes



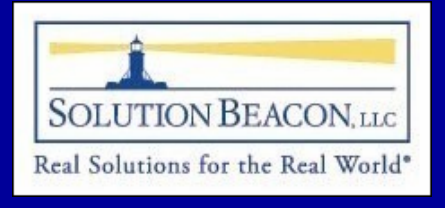
The Future



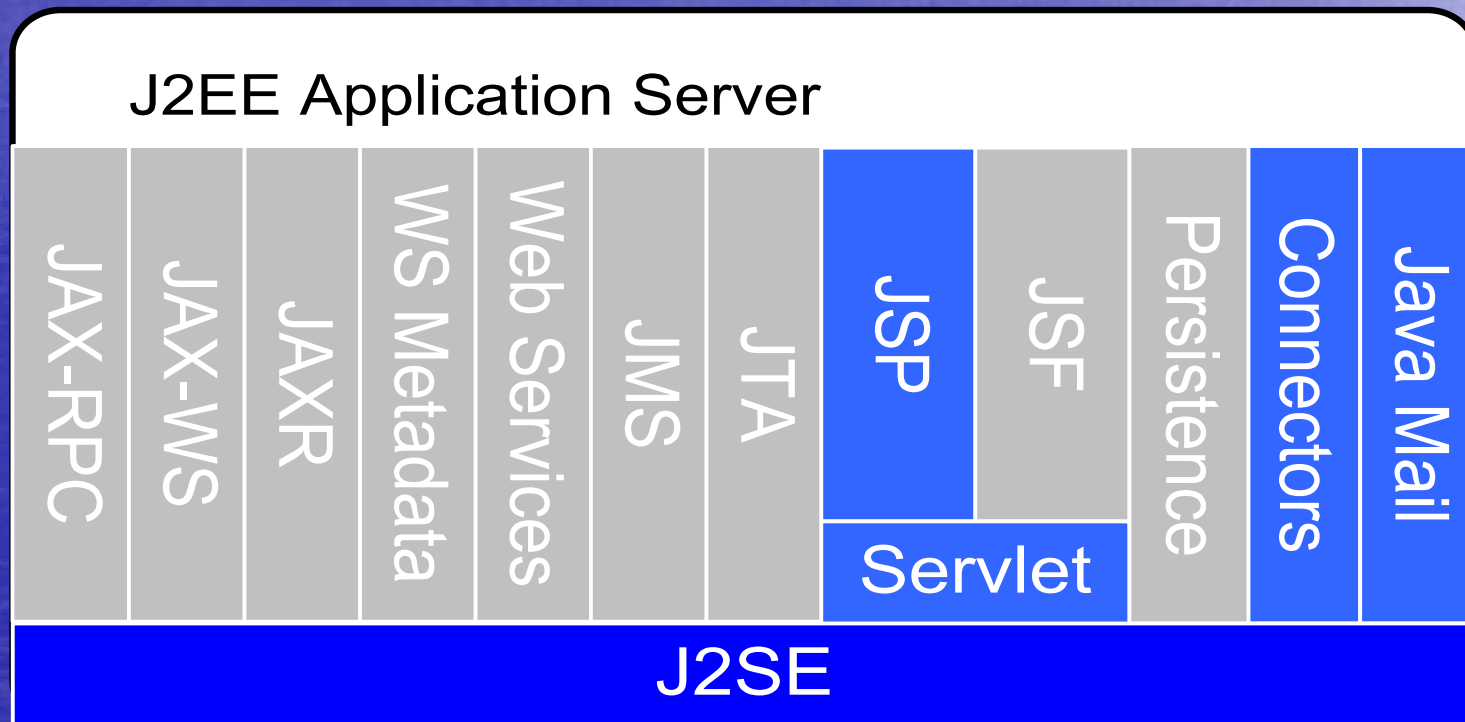
- ◆ Driven by industry standards
 - Release of the Java EE 5 spec drives the use of EJB
 - J2EE is driving towards web services and SOA
- ◆ Fusion makes the move to a Service Oriented Architecture (SOA) based environment
 - Same Java syntax, new programming models
 - Event driven processing
 - Focus on business modeling and processes
 - Distributed applications communicating via Enterprise Service Bus and Web Services



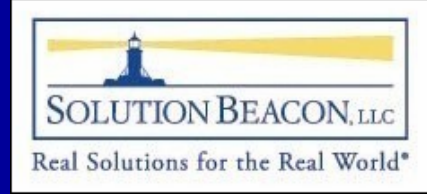
Past Environment



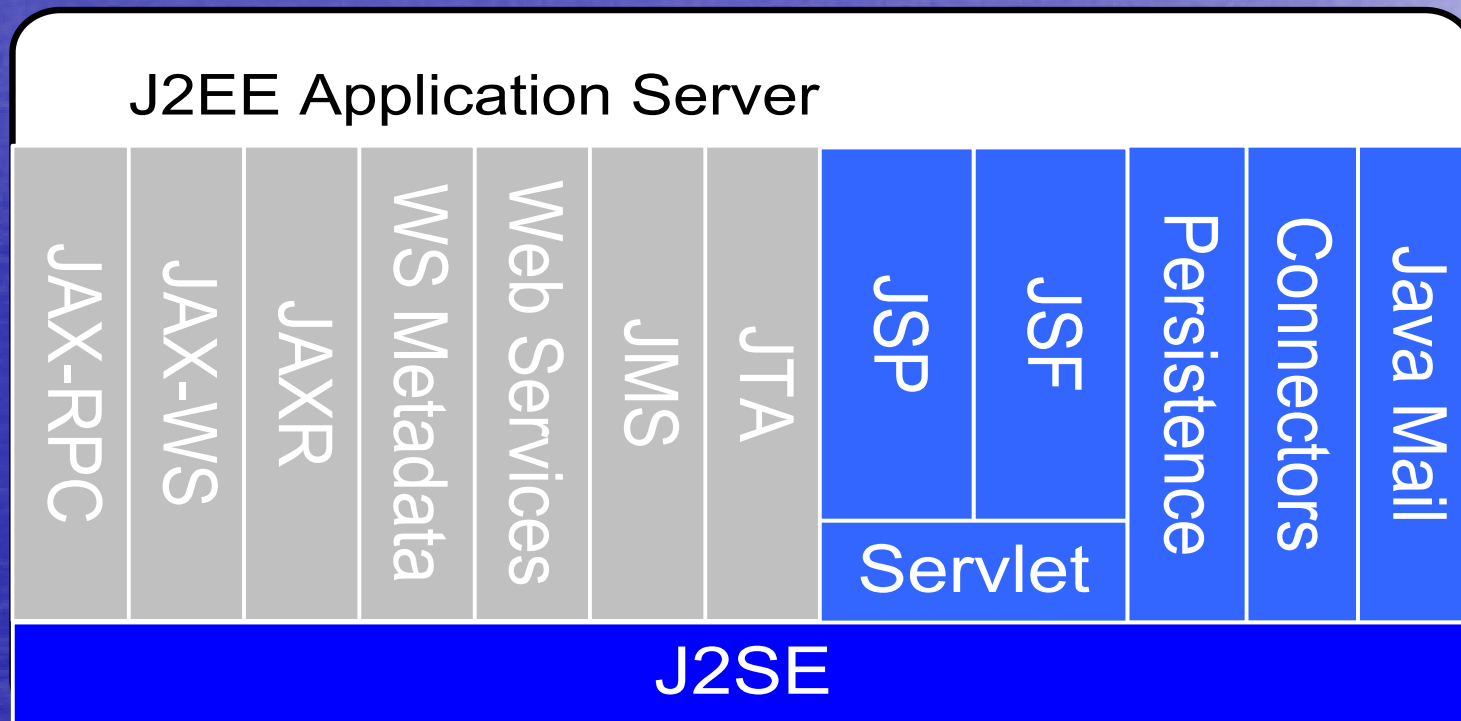
◆ Components used in OAF



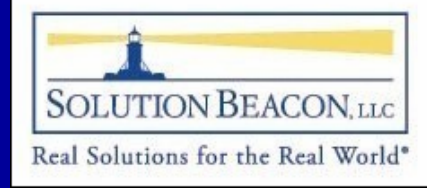
Present Environment



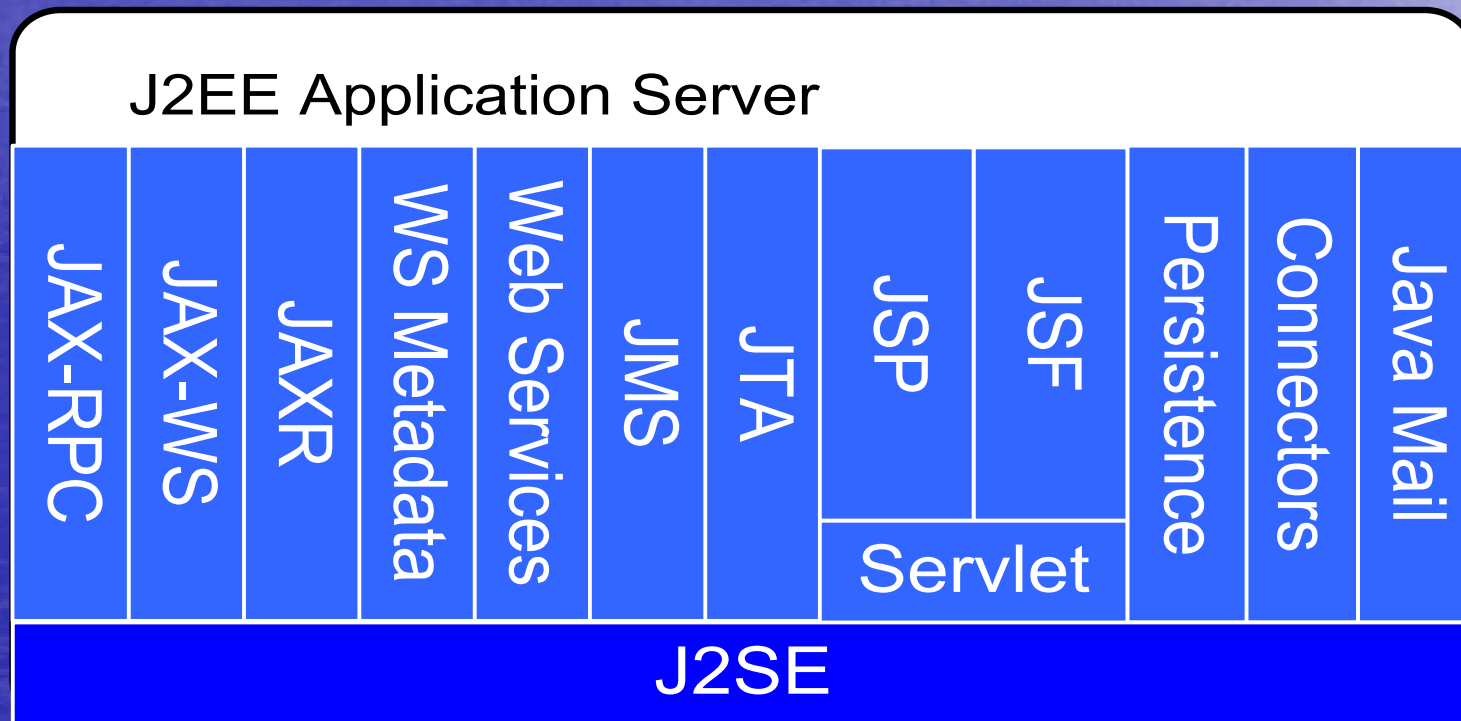
◆ Components used in ADF



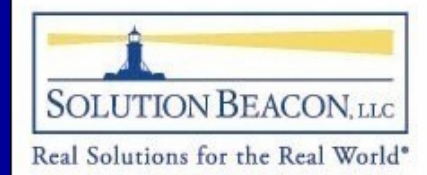
Future State



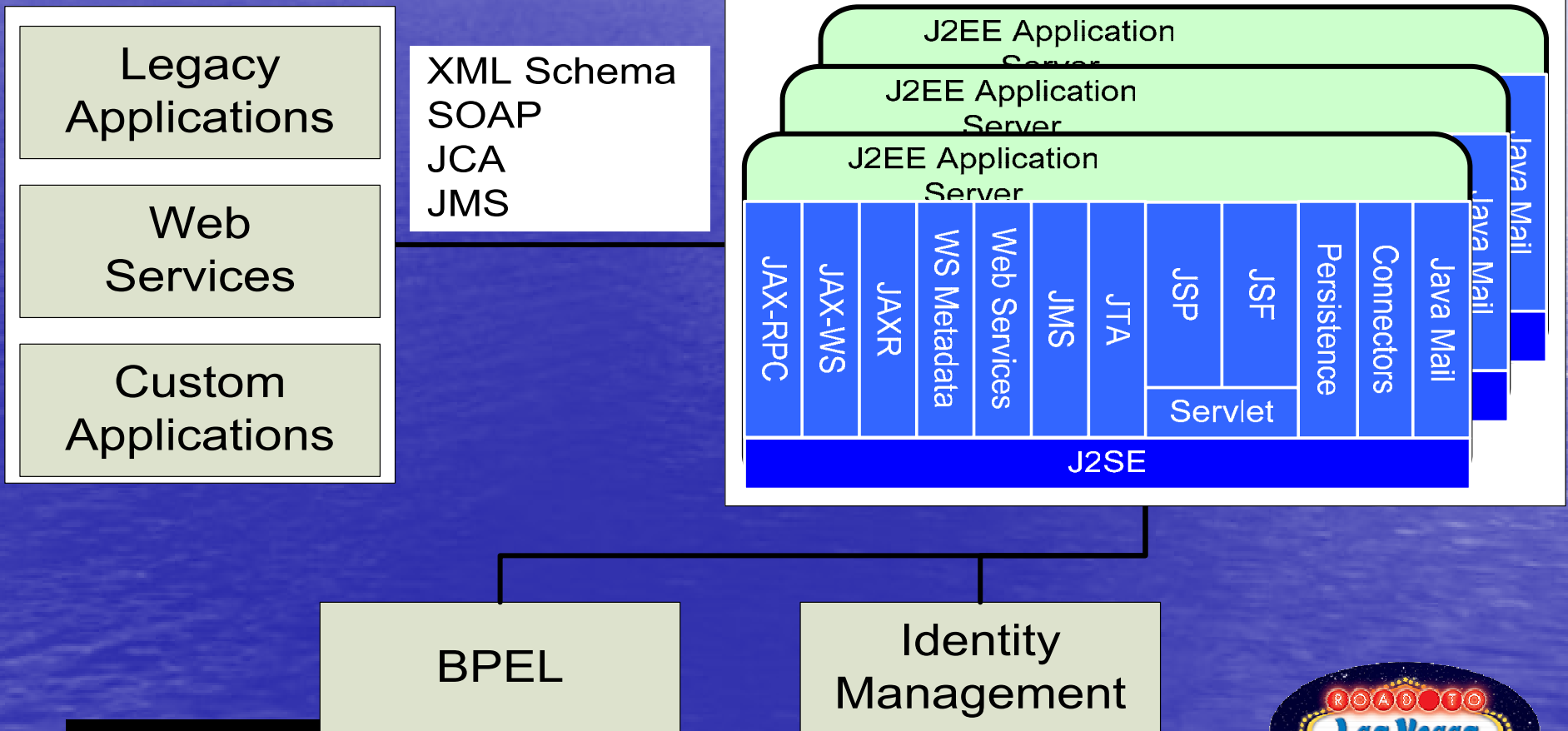
◆ Components needed for SOA and web services



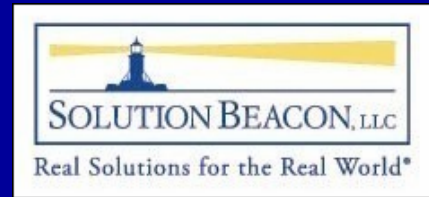
J2EE in the Architecture



Cluster



Questions and Answers



Thank you!

Lance Reedy

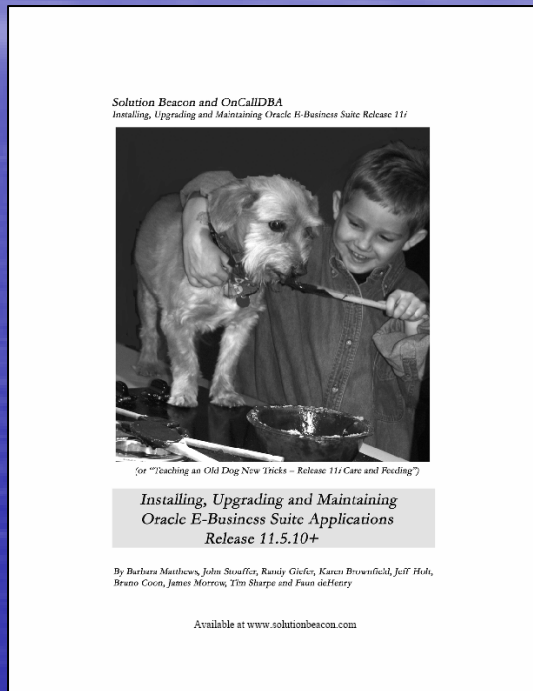
lreedy@solutionbeacon.com

www.solutionbeacon.com

Real Solutions for the Real World.®



Installing, Upgrading and Maintaining Oracle E- Business Suite Applications 11.5.10.2+



Got Oracle? Get the Books!

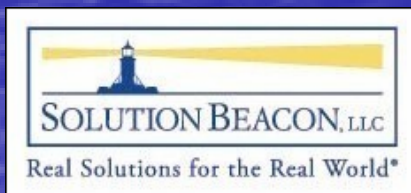
Need more info?
Visit our website

www.solutionbeacon.com



The Release 12 Primer –
Shining a Light on the
Release 12 World

Available
October 2007



Release 11/12 Workshops
Dallas, TX • Raleigh, NC • San Ramon, CA
Cincinnati, OH • Denver, CO • Atlanta, GA
Detroit, MI • Las Vegas, NV • Worcester, MA

