JEE Frameworks Overview

- JSF
- RichFaces
- JPA
- Spring
- Struts

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

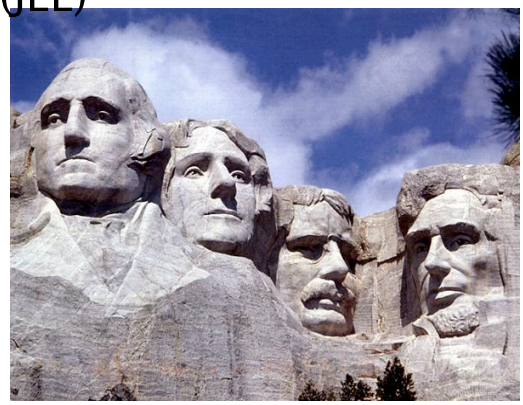
JEE Frameworks

- There are a plethora of standard and third-party frameworks and libraries available to Java developers
- Many of the frameworks are open-source
- This chapter overviews some of the popular frameworks and APIs:
    - JSF
    - RichFaces
    - JPA/Hibernate
    - Spring
    - Struts

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is JavaServer Faces?

- JavaServer Faces is a **component-based** framework for developing Java-based Web applications
- JSF supports a "rapid application development" programming style, especially coupled with a visual development tool
- JSF is a standard part of Java Enterprise Edition (JEE)





JSF borrows inspiration from client-side user-interface systems like Java's Swing, where you also build "screens" from components.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JSF Feature Overview

- A set of pre-written components (e.g. buttons, entry fields, data tables)
- A standard component programming model that encourages third-party component development
- An event-driven programming model similar to that of Java Swing
- Facilities for validating user input
- A page navigation model
- Support for internationalization (I18N)
- "Pluggable" view technologies (e.g. JSP or Facelets)
- An expression language for working with "managed beans"

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JSF Implementations

- The Sun JSF Reference Implementation (a.k.a. Mojarra)
- An implementation provided by a JEE application server
- Apache MyFaces

JSF is a specification, i.e. a document. Here we show a couple of implementations of the JSF specification.

You can find out more about Mojarra at http://javaserverfaces.java.net.

You can find out more about MyFaces at http://myfaces.apache.org.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JSF Hello, World

```
1    <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
2    <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
3    <html>
4      <f:view>
5        <body>
6          <h:form>
7            <h3>JSF Hello, World</h3>
8            <p>
9              Please enter your name:
10             <h:inputText value="#{hello.name}" />
11           </p>
12           <p>
13             <h:commandButton value="Submit"
14                action="#{hello.mymethod}" />
15           </p>
16         </h:form>
17       </body>
18     </f:view>
19   </html>
```

This is a JSP that acts as a JSF page.

In addition to this JSP, a complete JSF Hello, World application would need a faces-config.xml configuration file and the Java class that acts as the "hello" managed bean.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is RichFaces?

- From http://www.jboss.org/jbossrichfaces/:
    - RichFaces is a rich component library for JSF and an advanced framework for easily integrating AJAX capabilities into business application development.
- The RichFaces framework bundles the Ajax4jsf AJAX library
- RichFaces includes the Ajax4jsf project that essentially "AJAX-izes" existing JSF components so you don't need to write any JavaScript



Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

RichFaces Features

- JSF component library
- Ajax4jsf library
- **Skinning** for JEE Web applications
- A **Component Development Kit** (CDK) for creating new RichFaces components

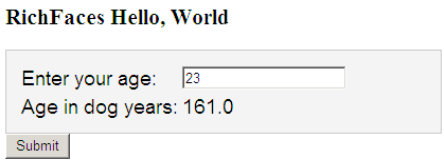
Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

RichFaces and JSF

- RichFaces extends JSF by providing:
    - Additional JSF components
    - Ability to add **Ajax** (Asynchronous JavaScript and XML) capability to existing JSF components

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

RichFaces Hello, World



MyBean.java

first.xhtml

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

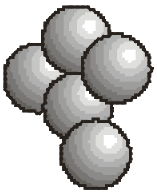
```
1    package mypackage;
2
3    public class MyBean
4    {
5        private int age;
6
7        public int getAge()
8        {
9            return age;
10       }
11
12       public void setAge(int age)
13       {
14           this.age = age;
15       }
16
17       public double getDogAge()
18       {
19           return age * 7.0;
20       }
21
22       public String calc()
23       {
24           return null;
25       }
26   }
```

```
1    <html xmlns="http://www.w3.org/1999/xhtml"
2          xmlns:ui="http://java.sun.com/jsf/facelets"
3          xmlns:f="http://java.sun.com/jsf/core"
4          xmlns:h="http://java.sun.com/jsf/html"
5          xmlns:a4j="http://richfaces.org/a4j"
6          xmlns:rich="http://richfaces.org/rich">
7      <head>
8        <title>RichFaces Hello, World</title>
9      </head>
10
11     <body>
12       <f:view>
13         <h3>RichFaces Hello, World</h3>
14         <h:form id="helloForm">
15           <rich:panel style="width: 50%">
16             <h:panelGrid columns="2">
17               <h:outputLabel for="age" value="Enter your age:" />
18               <h:inputText type="text" id="age"
19                  value="#{myBean.age}"/>
20               <h:outputLabel for="msg" value="Age in dog years: " />
21               <h:outputText id="msg" value="#{myBean.dogAge}" />
22             </h:panelGrid>
23           </rich:panel>
24           <h:commandButton id="submit"
25             action="#{myBean.calc}" value="Submit" />
26         </h:form>
27       </f:view>
28     </body>
29   </html>
```

What is Object-Relational Mapping

- Object/relational mapping is the act of connecting data from a relational database with objects in an object-oriented language like Java

Object Space





Relational Space

STUDENT

STUDENTID	NAME	DATE
5	Jess Settles	1998-12-02
6	George Peeples	1966-05-19
7	Ben McGilmer	1965-03-13

Object orientation is a useful programming paradigm that facilitates reuse and easier maintenance. However, most of the world's data currently resides within relational databases managed by products like Oracle, DB2 and so forth.

O/R mapping lets developers model using object-oriented techniques while accessing/storing data in a relational database.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

O/R Approaches

- Roll your own JDBC
- EJB2 entity beans with container-managed persistence
- EJB3/Java Persistence API entities
- Use a framework/tool:
    - Hibernate
    - iBATIS
    - Others

Java programmers have several options to choose from when considering O/R mapping. One approach is to write the JDBC yourself -- while this gives the most flexibility and potentially best performance, it's code-intensive and tedious.

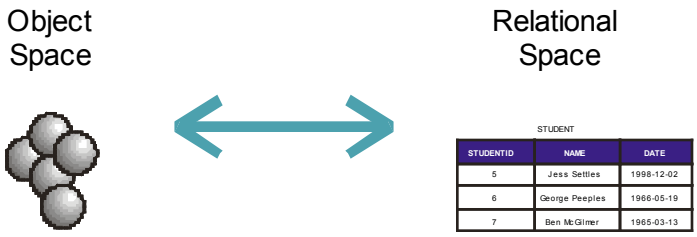
Another approach is to use entity Enterprise JavaBeans. Unfortunately, many architects and developers have decided that in EJB Version 2.0 and prior, entity beans are most often too heavyweight, have a steep learning curve and require too many associated files (e.g. primary-key classes, home interfaces, deployment descriptors and so forth). While this perception may not be entirely accurate (and is mostly addressed in EJB 3.0), the perception has hampered the acceptance of entity beans.

The final choice presented here is to use a non-EJB O/R mapping framework. Such frameworks typically generate the tedious JDBC code, while being easier to code than EJB2 entity beans.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Intro to Java Persistence

- **Java Persistence** is the official Sun API for object/relational mapping
- Originally only for use with EJB3, Java Persistence is now a separate API and can be used in Java SE applications
- JPA is a required part of Java Enterprise Edition, Version 5 or later



Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JPA Implementations

- Since Java Persistence by itself is just a specification, you must obtain an actual implementation
- Current implementations:
    - Any JEE 5 or later application server, including GlassFish and JBoss 5
    - Oracle TopLink and TopLink Essentials
    - Hibernate 3

The GlassFish project from Sun is an open-source implementation of Java Enterprise Edition (Java EE) and contains a Java Persistence implementation provided by Oracle: TopLink Essentials.

TopLink Essentials can also be used outside of the JEE environment, say for a standalone Java application.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is a JPA Entity?

- A Java Persistence entity is a **plain old Java object** (POJO) with data that maps to a relational database
- Persistent object classes don't need to implement any particular interfaces
- Java Persistence apps can use JDK 5 annotations to provide metadata
- Entities must have something that makes them different from all other entities of the same type (primary key)



Column Name	Data Type	Nullable	Default	Primary Key
HELLOID	NUMBER	No	-	1
MSG	VARCHAR2(40)	Yes	-	-

Entities are often written in the form of a JavaBean with get/set methods. They also use annotations to indicate how the entity should be persisted.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Simple JPA Entity

```
1    package hello;
2
3    import javax.persistence.Entity;
4    import javax.persistence.Id;
5
6    @Entity
7    public class Hello
8    {
9        @Id
10       private int helloID;
11       private String msg;
12
13       public void setHelloID(int helloID)
14       {
15           this.helloID = helloID;
16       }
```


Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Simple JPA Entity, cont'd

```
17        public int getHelloID()
18        {
19            return helloID;
20        }
21
22        public void setMsg(String msg)
23        {
24            this.msg = msg;
25        }
26
27        public String getMsg()
28        {
29            return msg;
30        }
31    }
```


Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Simple JPA Main Program

```
1    package client;
2
3    import hello.Hello;
4
5    import javax.persistence.EntityManager;
6    import javax.persistence.EntityManagerFactory;
7    import javax.persistence.Persistence;
8    import javax.persistence.Query;
9
10   public class TestClient
11   {
12       public static void main(String[] args)
13       {
14           EntityManagerFactory emf =
15            Persistence.createEntityManagerFactory(
16            "default");
```


Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Simple JPA Main Program, cont'd

```
17            EntityManager entityManager =
18              (EntityManager)emf.createEntityManager();
19            entityManager.getTransaction().begin();
20
21            Hello h = new Hello();
22            h.setMsg("hi, world");
23            h.setHelloID(33);
24            entityManager.persist(h);
25            entityManager.getTransaction().commit();
```


Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Simple JPA Main Program, cont'd

```
26            entityManager.getTransaction().begin();
27            Query q = entityManager.createQuery(
28    "SELECT h FROM Hello h WHERE h.helloID = :id");
29            q.setParameter("id", 33);
30
31            Hello h1 = (Hello)q.getSingleResult();
32            System.out.println(
33              "Message: " + h1.getMsg());
34            System.out.println(
35              "ID: " + h1.getHelloID());
36            entityManager.getTransaction().commit();
37        }
38    }
```


Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is the Spring Framework?

- The **Spring Framework** is an open-source, Java and JEE framework that aims to make developing applications easier
- Spring applications are based on **plain old Java objects** (POJO) also known as JavaBeans
- Objects in a Spring application should not depend on Spring and should be easy to test



The homepage for the Spring Framework is at:

http://www.springsource.org/

Note that Spring is also available for Microsoft's .NET.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

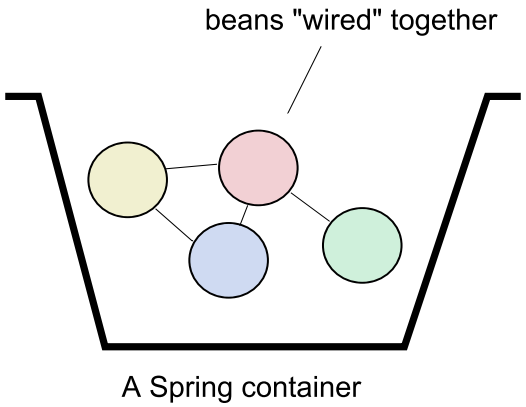
Spring Fundamentals

- Use of Spring beans (POJOs) instead of requiring classes to implement framework interfaces
- Loose coupling between objects
- **Dependency injection** (DI) to give objects resources instead of requiring lookups
- **Aspect-oriented programming** (AOP) to free objects from performing system functions such as logging and transactions
- Spring doesn't re-invent the wheel, but can integrate with other frameworks (e.g. Java Persistence)

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Spring Container

- To manage the lifecycle of the beans, Spring provides a lightweight **container**
- Spring apps create the container, then retrieve bean references from the container
- The container can **wire**, or hook up beans to each other via *dependency injection*



A container is a set of runtime that performs useful tasks. In Spring, one such useful task is ensure that beans have their dependencies satisfied. This is quite different from traditional J2EE, where objects use JNDI to find their own dependencies. Here, the container injects the dependencies so the object doesn't need to do any look ups.

Beans can have references to other beans or to external resources. The container can connect a bean to its referenced objects as part of creating the set of managed objects.

This process, known as "wiring" is typically configured in an XML file that the container reads when it starts.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is Apache Struts?

- Struts is a framework to make it easier to write maintainable Model 2 (MVC) J2EE wWeb applications
- Benefits
    - A more object-oriented approach
    - Concentrate on application logic rather than "plumbing"
- Drawbacks
    - A learning curve
    - Struts is not under much active development

Struts is a set of classes and custom actions that help you write web applications that use the Model-View-Controller architecture. Struts automates some of the more tedious parts of MVC, especially writing the Controller.

In addition, Struts provides a more object-oriented view on a web application, allowing you to decompose complex logic into simple, well defined classes.

You can find out more about Struts from http://jakarta.apache.org/struts.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

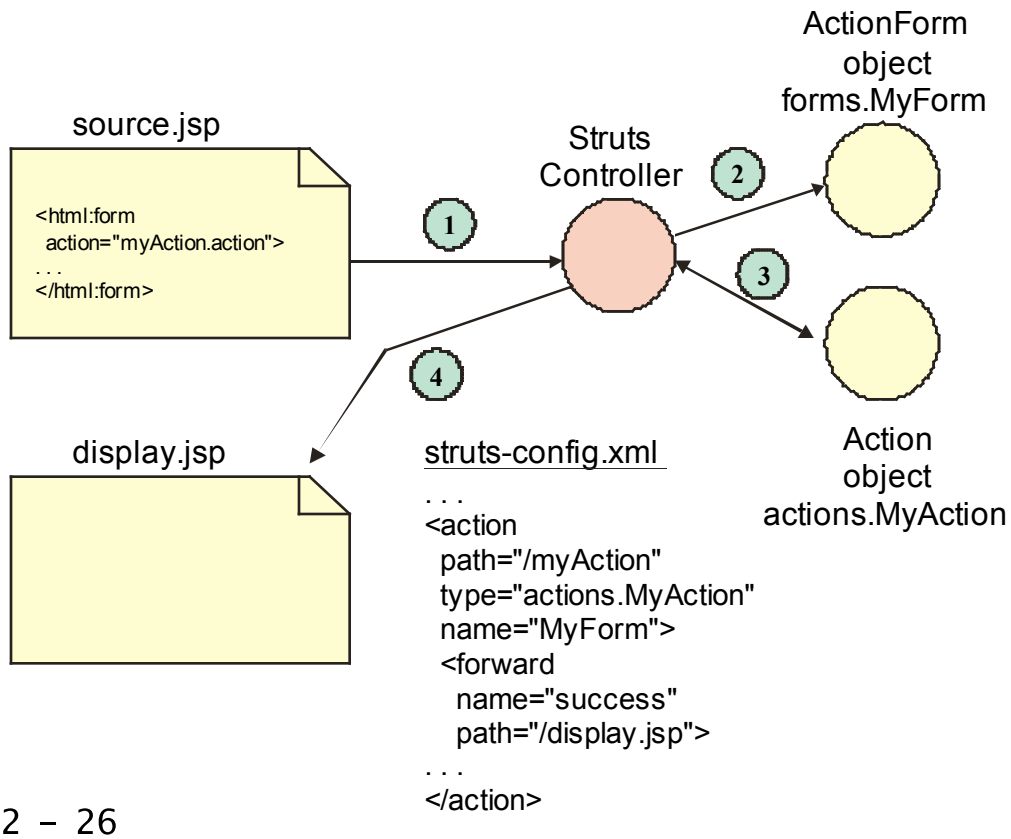
Struts Architecture Highlights

- Struts provides the front controller servlet for you -- no more cumbersome if/else if/else constructs
- Instead of writing a controller, you write **action** classes, each of which handles a particular request
- Struts also provides a set of custom JSP actions so you can create HTML forms that easily participate in Struts

Struts provides a more object-oriented way of handling requests -- it uses the Command pattern from the classic Design Patterns book by Gamma, et. al.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Struts Request Processing



This figure shows the flow of control for a request in a typical Struts-based web application. You can refer back to this page as you go through the rest of the chapter.

The flow starts (1) when the user submits an HTML form that is coded to comply with Struts. The Stuts Controller servlet handles the request by (2) creating and calling an ActionForm object, which is a JavaBean that exposes properties that match the input widgets on the form. The Controller then (3) creates and calls the Action object, which contains the code you write that fulfills the request, using data from the ActionForm. The Controller then (4) transfers control (e.g. forwards the request) to another URL, typically a JSP that displays the results of the request. The JSP can access any request, session or application-scoped variables created by the Action.

The component that ties all of this together is the struts-config.xml file, which you write to configure your Struts application.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned about several popular JEE frameworks and APIs, including:

- JSF
- RichFaces
- JPA
- Struts

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum