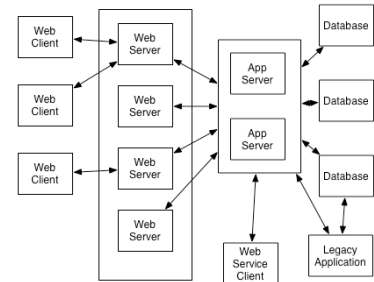


CMSC 433 – Programming Language Technologies and Paradigms Spring 2005

Enterprise Applications and
Enterprise Java Beans
May 5, 2005

Enterprise Applications

- Examples
 - E-Bay
 - Amazon
 - Testudo



2

Features of Enterprise Apps

- Persistent data
 - Databases store essential business information
 - Need to worry about integrity, transactions
- Many interfaces
 - Web clients
 - Web services
 - Legacy applications

3

More Features

- Distribution and scaling
 - Demands on successful app grow over time
 - Should be possible to scale up
 - Multiple machines
 - Geographic distribution
- Resiliency to failure
 - One machine failure cannot shut down app
 - ...app failure = \$\$\$ lost

4

Enterprise Apps and Java

- Extensive support enterprise apps in Java
 - EJB, JSP, JDBC, BMP, CMP, JDO, WSDP, ...
- Focus of this lecture: How do we build these systems?

5

Key Properties of a Database

- A *transaction* is a set of consistent changes to a database
 - Ex: `balance := balance - $100; dispense $100`
- ACID transactions
 - Atomicity -- actions all occur or none occur
 - Consistency -- respect domain invariants
 - Isolation -- no interference with other xactions
 - Durability -- persistent even if system fails

6

Database Possibilities

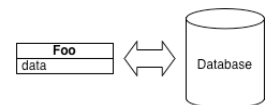
- ACID transactions are well-understood
 - Hard to implement correctly
 - ...but good implementations available
- *Relational databases* are the standard
 - Scale up to very large data sets
 - Handle transactions and failure well
- But don't interact that well with Java
 - SQL for queries
 - Awkward to construct, use correctly

7

Persistent Objects

- Represent persistent state with objects
 - Reads and writes become actions on the database
 - Largely transparent to object client

```
Foo foo = ...  
foo.data = ... // database write  
... = foo.data; // database read
```



8

Enterprise Java Beans (EJB)

- A system for persistent objects
 - ...and many other things
- A *bean* is a class that implements a feature of your application
 - AccountInfo, Order, ShoppingCart, ...
- Beans live in containers
 - Containers often implement security, persistence, etc

9

Kinds of Beans

- Entity beans
 - Represent persistent data
 - BMP - bean-managed persistence
 - Bean contains code to contact database
 - CMP - container-managed persistence
 - AppServer gives mapping to database
 - Beans are more portable
 - Cached in memory
 - Data in database is materialized as the bean

10

Kinds of Beans (cont'd)

- Session beans
 - Represents a task carried out by a user
 - Stateful
 - Tracks state across method calls
 - Useful when actions need to be carried out in sequence
 - Ex: Client logs in to app, so need to track client id
 - Not durable: Go away after timeout or crash
 - Stateless
 - Handles one, isolated request from client
 - Ex: Send an e-mail confirmation, verify credit card¹¹

Kinds of Beans (cont'd)

- Message-driven beans
 - Listens for asynchronous messages (observer)
- ...we won't talk about these

12

Assumptions

- We'll assume all calls to EJBs are remote
 - In practice, people use entity beans locally
- Clients never talk directly to a bean
 - Instead they go through stubs (proxy pattern)
 - Enforces security, transactions

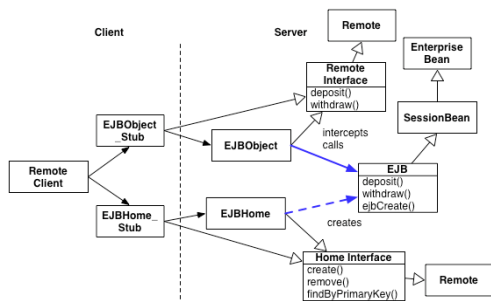
13

Beans Provide Two Interfaces

- *Remote or component interface*
 - Actions for business logic
- *Home interface*
 - Factory design pattern (create, destroy, find)

14

Bean Interfaces



15

You Provide

- Component and home interface
 - How clients interact with bean
- Bean implementation class
 - Does **not** implement component/home interface
 - Client cannot directly access bean; must use container
- *Deployment descriptor* (in XML)
 - Tells container how to manage the bean

16

Container Provides

- Home implementation and stubs
- Component implementation and stubs
- Deployment of beans with features specified by deployment descriptor
 - Reflection used to find bean methods
 - Ex: Will look for deposit() method in EJB

17

Example: Hello World

- Step 1: The Bean class

```
import javax.ejb.*;
public class AdviceBean implements SessionBean {

    private String[] adviceStrings = {"test", "test1", "test2"};

    public String getMessage() {
        System.out.println("in get advice");
        int random = (int) (Math.random() * adviceStrings.length);
        return adviceStrings[random];
    }

    public void ejbCreate() { System.out.println("ejb create"); }
    ...
}
```

Example from *Head-First EJB* 18

Example: Hello World

- Step 2: The interfaces

```
import javax.ejb.*;
import java.rmi.RemoteException;

public interface Advice extends EJBObject {
    public String getMessage() throws RemoteException;
}

public interface AdviceHome extends EJBHome {
    public Advice create() throws CreateException, RemoteException;
}
```

19

Example: Hello World

- Step 3: The deployment descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/j2ee" version="2.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
  <display-name>Ejb1</display-name>
  <enterprise-beans><session>
    <ejb-name>Advisor</ejb-name>
    <home>AdviceHome</home>
    <remote>Advice</remote>
    <ejb-class>AdviceBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <security-identity><use-caller-identity/></security-identity>
  </session></enterprise-beans>
</ejb-jar>
```

20

Example: Hello World

- Step 4: Put it together
 - Package up all files into `ejb-jar.jar`
 - `jar cMf ejb-jar.jar Advice.class AdviceBean.class AdviceHome.class META-INF/ejb-jar.xml`
 - Deploy it
 - `cp ejb-jar.jar ~/java/jboss/server/default/deploy`

21

Example: Hello World

- Step 5a: Write the client

```
public class AdviceClient {  
  
    public static void main(String[] args) throws Exception {  
        new AdviceClient().go();  
    }  
  
    public void go() throws Exception {  
        Context ic = new InitialContext();  
        Object o = ic.lookup("Advisor");  
  
        AdviceHome home = (AdviceHome)  
            PortableRemoteObject.narrow(o, AdviceHome.class);  
  
        Advice advisor = home.create();  
        System.out.println(advisor.getMessage());  
    }  
}
```

22

Example: Hello World

- Step 5b: Specify some extra properties

File `jndi.properties`:

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory  
java.naming.provider.url=localhost:1099  
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
```

- Step 6: Run it
 - `java AdviceClient`

23

CMSC 433 – Programming Language
Technologies and Paradigms
Spring 2005

XML
May 5, 2005

Alphabet Soup

XSL JAX-RPC SOAP
DTD XML XPath
DOM HTML
JAXM JAXB W3C XSLT
UDDI CSS
XPointer SGML
JAXP XLink
XMLP ebXML JAXR
SAX

25

Background

- Applications tend to represent data in proprietary internal formats
 - E.g., Word, Excel, Quicken
- Difficult to share data
 - Between different software versions
 - Between different applications
 - Between different platforms

26

eXtensible Markup Language

- Goal: Provide a universal external format for application data
- Siméon and Wadler:
 - “[T]he essence of XML is this: the problem it solves is not hard, and it does not solve the problem well.”

27

HTML

- You’re probably familiar with HTML:

```
<html>  
<head><title>CMSC 433</title></head>  
<body><p>Hello, world!</p></body>  
</html>
```

- Pretty good for display, but won’t really work as a file format

28

HTML Not Good for Data

- Not extensible
 - Fixed set of tags like <a>, <p>, <h4>, etc.
- No semantic structure
 - Divides document into headings, paragraphs, tables etc.
 - ...which doesn't match a lot of file formats
 - E.g., spreadsheets

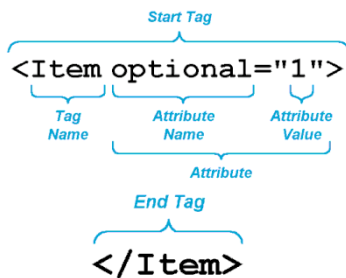
29

XML Example

```
<?xml version="1.0"?>
<reading>
  <book>
    <title>Thinking in Java</title>
    <author>Bruce Eckel</author>
  </book>
  <book>
    <title>Program Development in Java</title>
    <author>Barbara Liskov</author>
    <authorwith>John Guttag</authorwith>
  </book>
</reading>
```

30

Notation



from <http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml-p3.html>

31

Attributes

- Tags can have attributes
 - `<book binding="hardcover">...</book>`
- Attributes don't add any expressiveness
 - Could also have `<hardcoverbook>...</hardcoverbook>`
 - But attrs sometimes make things easier

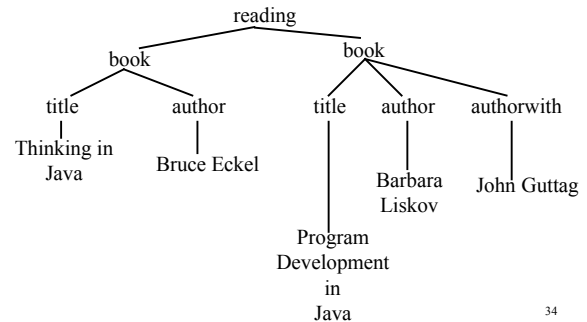
32

Syntactic Notes

- Begin with magic xml incantation
- Must have root element
- All elements need a closing tag
 - Shorthand:
 - Instead of `<Foo attr="value"></Foo>`
 - Can write `<Foo attr="value"/>`
- Elements must be properly nested

33

XML = Tree Structured Data



34

Parsing XML

- XML files are just text files
 - Can write with your favorite text editor
- Easy to parse XML into its tree structure
 - Can tell if a document is well-formed
 - But does it make sense?
 - Can't have two `<title>`s in a `<book>`
 - An `<author>` outside of a `<book>` doesn't make sense
 - Need to *validate* the document

35

Document Type Definition

- Defines a set of legal XML files

```
<!ELEMENT reading (book*)>
<!ELEMENT book (title, author, authorwith?)>
<!ATTLIST book binding CDATA "paperback">
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT authorwith (#PCDATA)>
```
- XML files can specify DTD

```
<?xml version="1.0"?>
<!DOCTYPE reading SYSTEM "reading.dtd">
```

36

XML Schema

- The replacement for DTDs

```
<?xml version="1.0"?>
<xsd:schema - start a schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  - Anything prefixed with xsd: is from ...w3.org... namespace
  targetNamespace="http://www.cs.umd.edu"
  - The tags we're defining are for this namespace
  xmlns="http://www.cs.umd.edu">
  - The default namespace for tags
  ...
</xsd:schema>
```

37

XML Schema Example

```
<xsd:element name="reading">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="book" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="title" type="xsd:string"/>
            <xsd:element name="author" type="xsd:string"/>
            <xsd:element name="authorwith" type="xsd:string" minOccurs="0"/>
          </xsd:all></xsd:complexType></xsd:element></xsd:sequence>
          <xsd:attribute name="binding" type="xsd:string" fixed="paperback"/>
        </xsd:complexType>
      </xsd:element>
```

38

Advantages of Schemas

- Schemas are written in XML
 - So schemas can describe schemas
- Schemas have type information
 - Can introduce new named types

```
<xsd:simpleType name="ssnumber">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-\d{2}-\d{4}"/>
  </xsd:restriction></xsd:simpleType>
```

(Brill, CodeNotes for XML)

39

Design Goals of XML

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

40

Design Goals of XML (cont'd)

6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.