

CS290i - Lecture 8

Beans, beans, beans

Scalable Internet Services and Systems, Winter 2002

Thorsten von Eicken
Department of Computer Science
University of California at Santa Barbara

Beans

Java's Component Software Model

■ a Java Bean is:

- a reusable software component
- Typically used in UIs
- that can be visually manipulated
- in builder tools



slider beans

■ Concepts

- Similar to "VBXs", "OCXs", "controls", "widgets",
- Builder tools query components about their properties and behavior (introspection)
- Visually select components from palettes, panels
- Drop them into a form
- Hook up events to event handlers
- Design-time vs. run-time



calendar bean

2

Defining Features

- **Introspection** allows a builder tool to analyze how a bean works
- **Customization** allows a user to alter the appearance and behavior of a bean
- Beans can fire **events**, and builder tools can find out about the events they can fire and the events they can handle
- **Properties** allow beans to be manipulated programmatically, and support the customization
- **Persistence** allows the state of customized beans to be saved and restored

3

Properties design patterns

■ Instance variable

- value can be manipulated with **set** and **get** methods
- Examples:
 - ◆ Label has a text field that can be set with **setText** and gotten with **getText**
 - ◆ Component.resize has been renamed to **setSize** there is also **getSize**

■ Boolean variables

- instead of a get method, an **is** method is used
- Example:
 - ◆ TextComponent has a property called **editable**
 - ◆ set with **setEditable(boolean)**
 - ◆ tested with **isEditable**

■ **exposed property**: the methods of a property are public

4

Implicit Introspection

■ java.beans.Introspector

- can be used to find out about the properties, events, and methods of a class
- implicit method or explicit method

■ Implicit method

- Uses design patterns
- The Introspector creates a list of all public methods in the bean
- Then searches that list for signatures that match a particular pattern
- Example:
 - ◆ **public void setFlavor(String f)**
- Implies:
 - ◆ Property called "flavor", of type **String**
 - ◆ **setFlavor** is the **write accessor** for the property

5

Implicit Introsp. (cont.)

■ Other design patterns..

- get/set
 - ◆ **public PropertyType getPropertyName()**
 - ◆ **public void setPropertyName(PropertyType v)**
 - ◆ These indicate that the Bean has a property `propertyName` of class `PropertyType`.
- is/get/set
 - ◆ **public boolean isPropertyName()**
 - ◆ **public boolean getPropertyName()**
 - ◆ **public void setPropertyName(boolean v)**
- Indexed Property Pattern
 - ◆ **public PropertyType[] getPropertyName()**
 - ◆ **public PropertyType getPropertyName(int i)**
 - ◆ **public void setPropertyName(PropertyType[] v)**
 - ◆ **public void setPropertyName(int i, PropertyType v)**

6

Implicit Introsp. (cont.)

- Public Method Pattern
 - ◆ The introspector creates method descriptors
 - ◆ for all of the bean's public methods
 - ◆ including all of the public methods of the bean's superclasses .
- Multicast Event Set Pattern
 - ◆ **public void addEventListener(EventNameListener l)**
 - ◆ **public void removeEventListener(EventNameListener l)**
 - ◆ an event set called `eventName`
 - ◆ Example:
 - ! **public void addActionListener(ActionListener listener)**
 - ! **public void removeActionListener(ActionListener listener)**
 - ! the bean fires action event sets that will be processed by `ActionListeners`

7

Explicit Introspection

■ Write explicit code to provide info

■ Example

- Bean `b` is class `Sample`
- Find class `SampleBeanInfo`
- Call `SampleBeanInfo.getBeanInfo(b)`, returns `BeanInfo` object
- `BeanInfo` object has get methods to provide info
 - ◆ E.g. `getEventSetDescriptors` returns information about the kinds of events fired by this bean

8

Enterprise Java Beans

Architecture for multi-tiered apps

- Allow app writers to concentrate on “business logic”
- And write portable applications
- Standard for app servers to implement

3-tier & multi-tier server architectures

- Instead of 2-tier client-server
- User interface (view)
- Business logic (model)
- System services (DB, thread pool, ...)

Middleware services

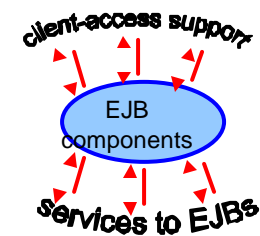
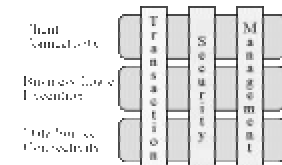
- Transaction monitors, state management
- Messaging, resource pooling
- Object request brokers, more...

9

Java App Servers

Java Application Server

- Provides services for web apps
- “Web server OS”
 - (not only web)
- Client connectivity
 - Protocol efficiency
 - Dispatch efficiency
 - Server clustering
- Business logic execution
 - Standardized by EJB (uses platform JVM)
 - Reliability
- Data source connectivity
 - Connection pooling
 - Request multiplexing
 - Caching



10

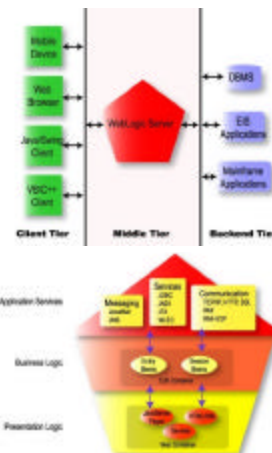
BEA WebLogic

Overview

- Leading Java application server
- Competes with IBM, Oracle, Sybase, etc.

Features

- Includes web server, or integrates with Apache, IIS, Netscape
- Clustering: load balancing and failover
- Implements J2EE (servlets, JSP, EJB, Java Messaging, ...)
- Distributed transactions, 2pc commit
- JDBC connection pooling



1

Business logic concerns

Things to consider when building a large server app...

- Lifecycle
 - process allocation, thread management, object activation, object destruction
 - thread-pooling, concurrency control, and resource management
- State Management
 - save or restore conversational object state between requests
- Security
 - authenticate users or check authorization levels
- Transactions
 - specify transaction demarcation code to participate in distributed transactions
 - manage the start, enrollment, commitment, and rollback of transactions

12

Business logic (cont.)

- Persistence
 - ◆ retrieve or store persistent object data from a database
 - ◆ DB performance optimization, replication
- Scaling
 - ◆ pool sizes, cache sizes, clustering, coherence
- Fault-tolerance
 - ◆ replication, fail-over

13

EJB system

3 pieces:

- EJB components ("EJBs")
 - ◆ Implement business logic
- EJB server
 - ◆ Actual process(es) in which EJBs run
 - ◆ May contain non-EJB code, e.g. JSP, Servlets, HTTP protocol, etc...
- EJB container
 - ◆ Environment in which EJBs execute, "EJB Operating System"
 - ◆ Loads EJBs on demand and configures them from specs
 - ◆ Shuffles EJB state to/from store (DB)
 - ◆ Provides interfaces to clients to access EJBs
 - ◆ Provide services to EJB components
 - Security, transactions, caching, ...

14

Types of EJB components

Session beans

- Transient bean instance that serves a single client
- Fields contain state of client's "session"
- Transient: gone when client ends session
 - ◆ Typically created & destroyed by container
- Stateful vs. state-less

Entity beans

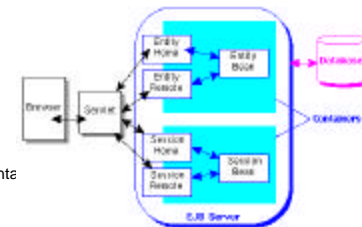
- Represents objects that contain data
 - ◆ Table of employees -> e-bean = 1 row = 1 employee
 - ◆ Customer, account, inventory item, credit card, discount, price quote, shipping address ...
- Transactional & long-lived
 - ◆ Typically mapped to objects in DB
 - ◆ Persistence either managed by the bean, or by container

15

EJB interfaces

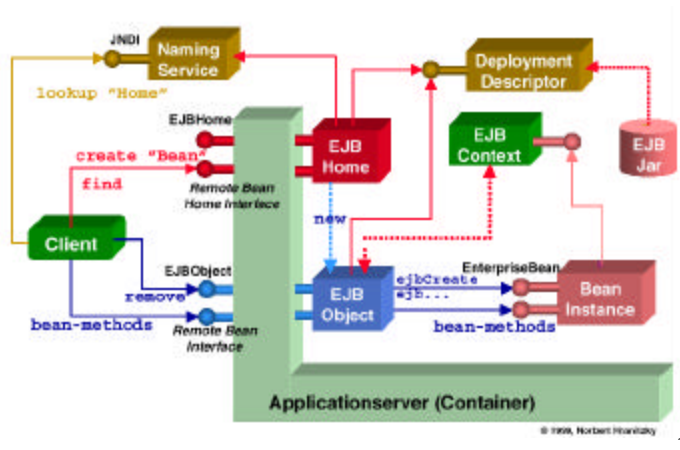
Wrapper interfaces:

- ◆ Remote interface
 - What the bean does
 - Used to invoke business logic
- ◆ Home interface
 - How to manage the bean
 - Create, delete, remove from container



16

EJB Architecture



17

EJB Example

“Write a session EJB”, M. Shoffner

• http://www.javaworld.com/javaworld/jw-07-1998/jw-07-step_p.html

Example: Discussion Forum Server

- Hashtable loadAllThreads ()
 - ◆ Keys: threads
 - ◆ Values: empty vectors
- Vector loadThreadArticles (String t)
 - ◆ Return vector with all articles of thread
- boolean postArticle (String art, String t)
 - ◆ Post article to thread t

18

To EJB or not to EJB?

“To EJB, or not to EJB?”, Humphrey Sheil

• <http://www.javaworld.com/javaworld/jw-12-2001/jw-1207-yesnoejb.html>

EJB Advantages

- The EJB Specification
 - ◆ Big, but complete
 - ◆ Defines roles, etc., Developer know what to do
- Integration with J2EE
 - ◆ Servlets, JMS, JSP, JCA, JDBC, ...
- Almost transparent scalability
 - ◆ Scalability largely a matter of the container & services

Since J2EE's earliest days, EJB technology has caused much controversy, with developers and architects constantly struggling to make best use of EJB in their projects.

Consequently, our industry has spawned folklore and rules of thumb to guide us how best to use EJB -- some true, some out of date, and some pure fabrication.

Javaworld article summary

19

EJB pros & cons

- Free access to and usage of complex resources
 - ◆ Transactions, security, pooling, naming, ...
- Strong & vibrant industry
 - ◆ Continuous improvement, long-term commitments
 - ◆ Multiple choices

EJB Disadvantages

- Large complex spec
- Increased development time
 - ◆ Due to spec & code complexity
- Added code complexity
 - ◆ Everything requires umpteen pieces
 - ◆ 3 classes per session bean, 4 per entity, deployment descr, ...
- Potential for more complex solution than necessary
- Continual spec revisions

20