Lecture 19: Intro to O/O Components
O/O Components

• A computational unit that exposes a well defined, contractual interface
  – constructed from a set of one or more classes

• Sample components
  – Applets, Servlets, Java Beans, Enterprise JavaBeans
O/O World Dream

• Goal: objects everywhere
• To get something done, send a message to the appropriate kind of object
• Problems
  – how to find objects, how to communicate
  – sharing, persistency, load balancing, transactions
Distributed O/O Objects – Simple Early Models

• RMI (Java), Corba, Com

• RMI
  – register remote objects with a public name
  – create stub and skeleton proxies that are network savvy

• Limited facilities for persistency, sharing, transactions, remote creation
Bean Components

• Java Beans
  – “A reusable software component based on the JB’s specification that can be manipulated visually in a builder tool.”

• Enterprise Java Beans
  – “A set of contracts between the component developer and the system that hosts the component.”
Additional Component Aspects

• Deployable
  – loaded into a container in which they can run
  – linked together to build a system

• Runs in a “container” that supplies services need for running and managing the components

• Distributable, with remote invocation via a specified interface
Servlet Components

• compiled Java classes with special methods
• run on a request/response oriented server
  – request (possibly with data) arrives at server
  – servlet is run to generate response that is returned
• Used for programming server side presentation logic, e.g.
  – servlet code constructs HTML specified page and returns it
  – results in a thin “client”, i.e. browser
Servlet Methods

• init(): called one time when servlet engine brings servlet into memory

• doget(): used for processing page hits from browser
  – has a request and a response argument
  – response can be used to redirect browser to new servlet/URL

• service(): general form of request made to servlet
Servlet Container

• Web server
• Container receives servlet request from Browser and invokes appropriate servlet service method
• First time servlet is loaded, its init() method is called
• Container maintains session objects – a kind of global object that servlets can use to store state
DS Servlets

• StartUp
• Login
• AdminActions
• MemberActions
Enterprise Java Beans

• Special Java classes
• “run” on a special server/container
• used for programming server side business logic
• 4 kinds
  – Session
    • stateful, stateless
  – Entity
    • container, bean managed persistence
Sample EJB Container/Server Functions

- Create and destroy beans
- Handle network communications for remote method execution
- Maintain bean pools – multiple instances for load balancing
- Instantiate beans with data from database
- Perform transaction synchronization
Session Java Beans

- Correspond to business logic functionality
- Are not permanent, only exist for use of system
- DS e.g.
  - LogIn: called by LogIn servlet when user logs on (stateful – saves name, user type, etc.)
  - GetDate: called by MemberActions servlet if user has indicated get a date, and entered desired dater properties (stateless)
Entity Java Beans

- Correspond to business logic permanent data
- DS e.g.
  - MemberData (we will choose container managed persistence)
Enterprise Java Bean Classes

• Actual class, implements Session/Entity Bean
• Remote Interface – shows accessible methods
  – extends EJBObject
• EJB Object generated by container from Remote Interface
• Home Interface
  – extends EJBHome
• EJB Home Object generated by container from Home interface
(Remote) Object Creation

Client

JNDI

home = Find Home Object

Invoke Method

EJB Container/Server

home: Home Object

obj: EJB Object

Invoke Method

Enterprise Bean(s)

obj = create

Invoke Method
Home Object Functions

- Create an instance of an EJB object, or use existing instance from a pool
- Find existing EJB objects (entity beans)
- Remove EJB objects
EJB Object Functions

• Creates or finds an instance of the associated actual bean to service the request
  • e.g. an existing stateless session bean
  • e.g. creates a new entity bean and fills in appropriate data from data base
Session Bean Callback Methods

• Created by programmer
• Are called by system/container when it is about to do something to your bean
• ejbPassivate(): writing bean to temporary storage if there are too many beans (Stateful)
• ejbActivate(): called after bean is brought back in
• ejbRemove(): cleanup before destruction
Entity Bean Callback Methods

- `ejbLoad()` and `ejbStore()`: used by container to keep a bean consistent with data base
- `ejbActivate()` and `ejbPassivate()`: used by container to allow entity bean to be reused for different data, i.e. transition into and out of an instance pool
Other Entity Bean Methods

• `ejbCreate()`: creates a bean plus data in the underlying data base. Could also created data base entry separately

• `ejbFindxxxx()`: methods for loading some data base data. e.g.
  – `ejbFindByPrimaryKey()`
  – defined automatically (container managed) or by user (bean managed)
Deployment

• Use a deployment wizard
• Deploy using an Ejb-jar file
  – Enterprise beans
  – Remote interfaces
  – Home interfaces
  – Deployment descriptors
• Container tool generates home and EJB objects for you
Deployment Descriptors

• Typical contents
  – Bean Home Name
  – bean class names
  – stateful or stateless
  – session timeout period
Bean Design Issues

• GetDate session bean needs to look through the MemberData entity beans
  – use the get all finder to return an enumeration and then index through them?
  – have a more sophisticated finder that just gets the ones that match?

• Consistency problems
  – two different clients have different session objects that are altering the data base at the same time that getDate and LogIn might be reading from it.
Sample DS Design Scenario

• Scenario 1
  – user logs on, is found to be a member
  – system displays member options web page

• Scenario 2
  – user enters desired properties on get date part of member options page, and hits enter
  – system finds a date and prints properties