Software Components with JavaBeans

- The JavaBeans API discussion began with a reminder that M. D. McIlroy (1968) made a plea for catalogs of software components more than 30 years ago.
  - JavaBeans, of course, is the standard component architecture for Java technology.
- JavaBeans is particularly well-suited for asynchronous, intra-application communications among software entities.
  - That is, JavaBeans is an intra-JVM (Java1 Virtual Machine) framework.
  - Every target, for example, that registers with a source does so by handing over its object reference, which the source Bean typically maintains internally in a vector.
  - This framework does not allow (in any direct way) for inter-application source-target communication because object references are local to the JVM that houses the running application.

Distributed Component Technologies

- There are several component technologies in the distributed computing arena.
  - Currently, the CORBA-based frameworks are the most popular,
  - newer component technologies such as Enterprise JavaBeans and
  - mobile agents, for example, Aglets, show considerable promise as well.
- The JavaBeans technology has been enhanced in some environments to support location transparency (almost).
  - For example, the Java application server from WebLogic/BEA Systems includes a JavaBeans implementation wherein each Bean is (in effect) wrapped in a network layer.
  - Thus, the source Bean could be running in the application server and the target object could be running on a distant client.

KISS (Keep It Simple, Stupid)

- Classes with many, complex constructors almost always lead to "documentation-itis" during development and result in unreadable source code once the application moves to the maintenance phase.
- The (practical) research on software design has suggested for more than ten years that certain object-oriented programming styles are effective.
- Certain **design patterns** tend to reappear with incredible regularity in most properly designed (large) applications.
  - These design pattern are summarized and cataloged in Design Patterns, by Gamma, et al. (1994).

Declarative vs. Imperative

- Declarative languages allow programmers to describe the state of an application,
  - the application adapts to that description,
  - as opposed to requiring that programmers intricately manipulate (execute) an application to arrive at a certain state.
- The JavaBeans framework has a declarative flavor and facilitates component design that involves plug-and-play, descriptive, declarative assembly of an application from cataloged parts.
  - you build applications by plugging and hooking together smaller components,
  - each of which takes on the responsibility of adapting itself to the declarative specifications that appear in an IDE's property sheet, or customization dialog.

Enterprise JavaBeans Technology

- Enterprise JavaBeans (EJB)
  - defines a model for the development and deployment of reusable Java server components.
- Components
  - pre-developed pieces of application code that can be assembled into working application systems.
- JavaBeans
  - support reusable development components.

- EJB architecture
  - logically extends the JavaBeans component model to support server components.

Server Components and Application Servers

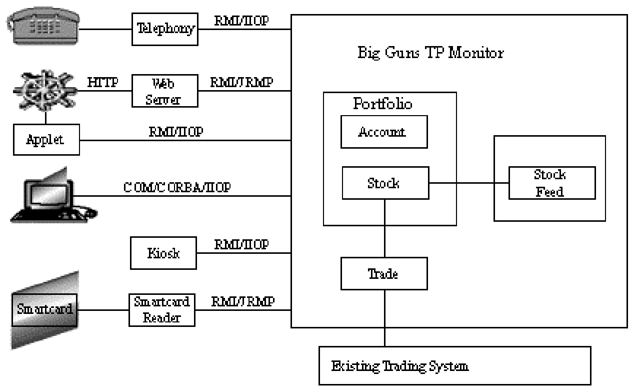
- Server components
  - are application components that run in an application server.
  - Applications are based on a multitier, distributed object architecture
  - Most of an application's logic is moved from the client to the server.
  - The application logic is partitioned into one or more business objects that are deployed in an application server.
- Java application server
  - provides an optimized execution environment for server-side Java application components.
  - high-performance, highly scalable, robust execution environment specifically suited to support Internet-enabled application systems.
  - "Write Once, Run Anywhere" (WORA) portability.

Communication Protocols

- Client applications may use a variety of protocols.
  - Java technology clients invoke the application using the native Java Remote Method Invocation (RMI) interface.
  - RMI requests are transferred using the Java Remote Method Protocol (JRMP) or the Internet InterORB Protocol (IIOP).
  - Native language clients can invoke the application using CORBA IDL running over IIOP or a COM/CORBA internetworking service running over IIOP.
  - The RMI client proxy could also be rendered as an ActiveX control to provide easy integration with any Windows application.
  - Browsers can invoke the application through a servlet running on the HTTP server.
  - The browser communicates with the servlet using HTTP, and the servlet communicates with the application using RMI.

Existing Infrastructure Integration

Money Makers Account Management System



EJB Architecture and their APIs

- EJB
  - The Enterprise JavaBeans API defines a server component model that provides portability across application servers and implements automatic services on behalf of the application components.
- JNDI
  - The Java Naming and Directory Interface API provides access to naming and directory services, such as DNS, NDS, NIS+, LDAP, and COS Naming.
- RMI
  - The Remote Method Invocation API creates remote interfaces for distributed computing on the Java platform.
- Java IDL
  - The Java Interface Definition Language API creates remote interfaces to support CORBA communication in the Java platform. Java IDL includes an IDL compiler and a lightweight, replaceable ORB that supports IIOP.

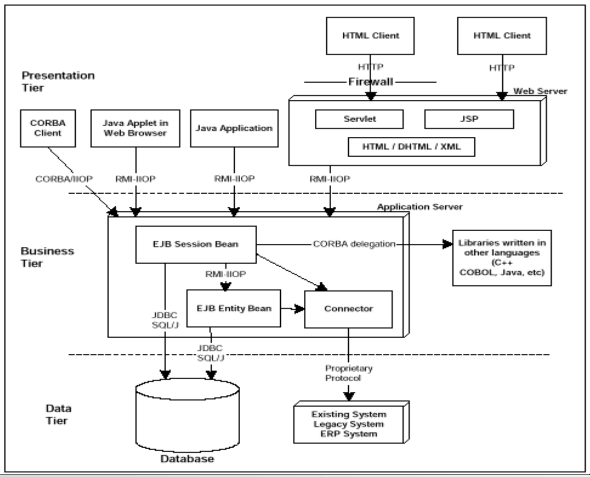
AP 11/01

EJB Architecture and APIs (contd.)

- Servlets and JSP
  - The Java Servlets and Java Server Pages APIs support dynamic HTML generation and session management for browser clients.
- JMS
  - The Java Messaging Service API supports asynchronous communications through various messaging systems, such as reliable queuing and publish-and-subscribe services.
- JTA
  - The Java Transaction API provides a transaction demarcation API.
- JTS
  - The Java Transaction Service API defines a distributed transaction management service based on CORBA Object Transaction Service.
- JDBCTM
  - The JDBC Database Access API provides uniform access to relational databases, such as DB2, Informix, Oracle, SQL Server, and Sybase.

AP 11/01

EJB Object Model



Enterprise JavaBeans Component Model

- The Enterprise JavaBeans component model logically extends the JavaBeans component model to support server components.
- Server components
  - are reusable, prepackaged pieces of application functionality that are designed to run in an application server.
  - They can be combined with other components to create customized application systems.
- Server components
  - are similar to development components, but they are generally larger grained and more complete than development components.
- Enterprise JavaBeans components (enterprise beans)
  - cannot be manipulated by a visual Java IDE in the same way that JavaBeans components can.
  - Instead, they can be assembled and customized at deployment time using tools provided by an EJB-compliant Java application server.

Implicit Services

- The EJB container performs a number of service on behalf of the enterprise beans-
- Lifecycle.
  - Individual enterprise beans do not need to explicitly manage process allocation, thread management, object activation, or object destruction.
- State Management.
  - Individual enterprise beans do not need to explicitly save or restore conversational object state between method calls.
- Security.
  - Individual enterprise beans do not need to explicitly authenticate users or check authorization levels.

Implicit Services (contd.)

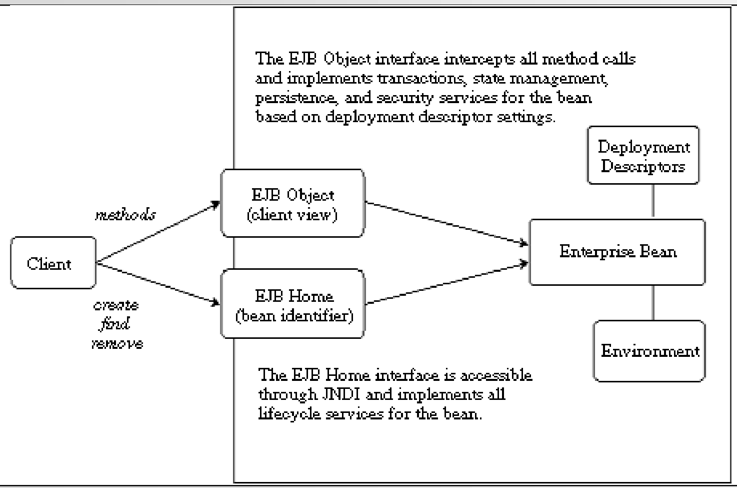
- Transactions.
  - Individual enterprise beans do not need to explicitly specify transaction demarcation code to participate in distributed transactions.
  - The EJB container can automatically manage the start, enrollment, commitment, and rollback of transactions on behalf of the enterprise bean.
- Persistence.
  - Individual enterprise beans do not need to explicitly retrieve or store persistent object data from a database.
  - The EJB container can automatically manage persistent data on behalf of the enterprise bean.

Potential Enterprise JavaBeans Systems

- TP monitors,
  - such as IBM TXSeries and IBM CICS/390
- Component transaction servers,
  - such as Sybase Jaguar CTS
- CORBA systems,
  - BEA Systems M3, IBM WebSphere Advanced Edition, Inprise VisiBroker/ITS
- Relational database systems,
  - such as IBM DB2, Informix, Oracle, and Sybase
- Object database systems,
  - such as GemStone GemStone/J
- Object/relational caching systems,
  - Persistence PowerTier and Secant Extreme
- Web application servers,
  - BEA WebLogic, Bluestone Sapphire, IBM WebSphere, Netscape Application Server, Oracle Application Server, Progress Apptivity, SilverStream Application Server, and Sun NetDynamics.

AP 11/01

EJB Container


The EJB Object interface intercepts all method calls and implements transactions, state management, persistence, and security services for the bean based on deployment descriptor settings.
Deployment Descriptors
methods
EJB Object (client view)
Client
Enterprise Bean
create find remove
EJB Home (bean identifier)
Environment
The EJB Home interface is accessible through JNDI and implements all lifecycle services for the bean.

AP 11/01

Session Beans

- A session bean is created by a client
  - exists only for the duration of a single client/server session.
- Performs operations on behalf of the client
  - such as accessing a database or performing calculations.
- Can be transactional
  - but (normally) they are not recoverable following a system crash.
- Can be stateless
  - or they can maintain conversational state across methods and transactions.
  - The container manages the conversational state of a session bean if it needs to be evicted from memory.
- A session bean must manage its own persistent data.

Entity Beans

- Object representation of persistent data
  - maintained in a permanent data store, such as a database.
  - A primary key identifies each instance of an entity bean.
- Entity beans can be created
  - either by inserting data directly into the database or by
  - creating an object (using an object factory Create method).
  - Entity beans are transactional, and they are recoverable following a system crash.
- Support for session beans is required,
  - but support for entity beans and container-managed persistence is optional.

Enterprise JavaBeans Security

- Automates the use of Java platform security
  - enterprise beans do not need to explicitly code Java security routines.
  - The security rules for each enterprise bean are defined declaratively in a set of AccessControlEntry objects within the deployment descriptor object.
  - An AccessControlEntry object associates a method with a list of users that have rights to invoke the method.
  - The EJB container uses the AccessControlEntry to automatically perform all security checking on behalf of the enterprise bean.

Packaging

- EJB components can be packaged
  - as individual enterprise beans,
  - as a collection of enterprise beans, or
  - as a complete application system.
- EJB components are distributed in a Java Archive File
  - called an ejb-jar file.
  - The ejb-jar file contains a manifest file outlining the contents of the file,
  - plus the enterprise bean class files,
  - the Deployment Descriptor objects, and, optionally,
  - the Environment Properties objects.

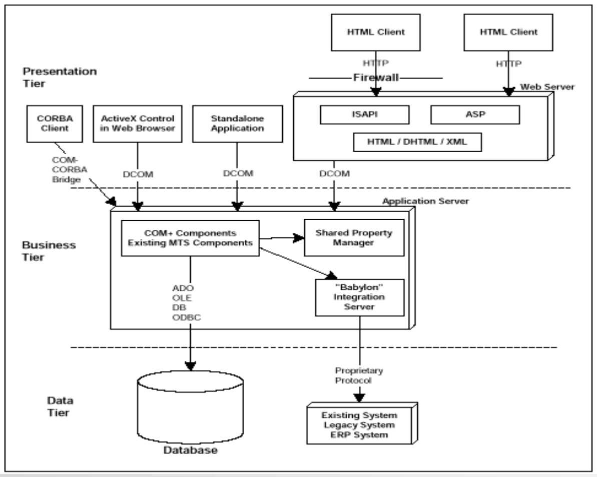
Deployment

- Deployment Descriptor object
  - used to establish the runtime service settings for an enterprise bean.
  - tells the EJB container how to manage and control the enterprise bean.
  - The settings can be set at application assembly or application deployment time.
- Deployment Descriptor defines
  - the enterprise bean class name,
  - the JNDI namespace that represents the container,
  - the Home interface name,
  - the Remote interface name, and
  - the Environment Properties object name.

MTS – EJB Competition

- Although Microsoft Transaction Server (MTS) could be adapted to support Enterprise JavaBeans components, Microsoft is not likely to make the effort.
- MTS provides a container system for COM server components, providing transactional and security services similar to those provided in Enterprise JavaBeans servers.
- COM+, the next generation of MTS, will provide a few additional capabilities, such as dynamic load-balancing and queued request-processing.

Microsoft Windows DNA Object Model



AP 11/01