JavaBeans and Software Architecture

Nikunj R. Mehta

Outline

- Java and JavaBeans
- JavaBean characteristics
- Enterprise JavaBeans
- JavaBeans & Java technologies
- Relation to software architecture
- Conclusion

Java and software development

- Developing large complex systems
  - Component-based development standards
  - Middleware platforms
  - Software architecture
- Java programming language
  - Object oriented PL with native multi-threading support
  - Reduces common programming mistakes
  - Provides greater control over software organization
- Java platform
  - Portable, secure, standards based

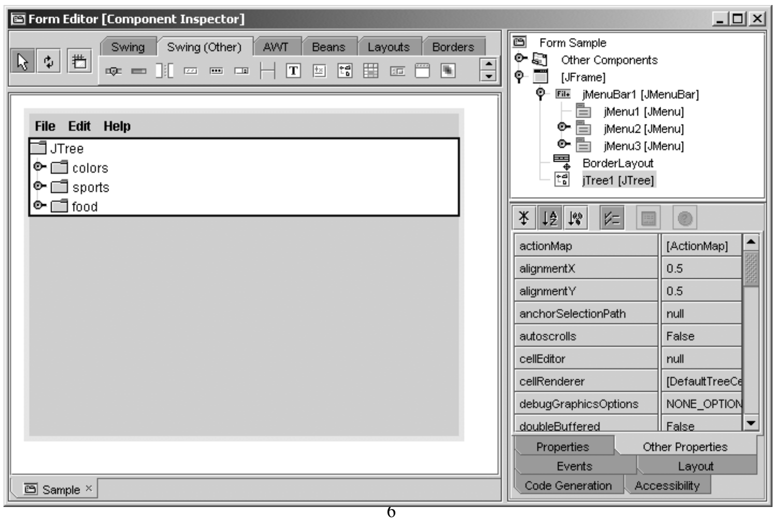
JavaBeans

- Original software component model for Java
- Flexible, loosely coupled architecture
- Enables large-scale, coarse-grained reuse
  - Large number of JavaBeans created, sold, and used
  - Used in many application domains
- Intuitive programming paradigm
  - No special class inheritance needed
  - Drag-and-drop visual editing of programs
  - API, programming conventions, and naming patterns
- Basis for many Java component technologies

Usage And Applications

- JavaBeans
  - A reusable software component that can be manipulated visually in a builder tool
- Separation of interfaces
  - Compile time and run time capabilities
  - Configuration and customization support
- GUI and non-graphical JavaBean components
- An important constituent of Java IDEs
  - Mostly used in developing *bigger* widgets and GUI applications
  - Visual editing as well as manual programming

NetBeans IDE: A Visual Builder



Form Sample
- Other Components
- [JFrame]
  - jMenuBar1 [JMenuBar]
    - jMenu1 [JMenu]
    - jMenu2 [JMenu]
    - jMenu3 [JMenu]
  - BorderLayout
  - jTree1 [JTree]

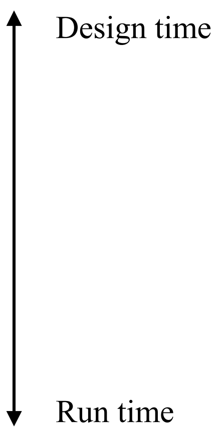
actionMap	[ActionMap]
alignmentX	0.5
alignmentY	0.5
anchorSelectionPath	null
autoscrolls	False
cellEditor	null
cellRenderer	[DefaultTreeCe
debugGraphicsOptions	NONE_OPTION
doubleBuffered	False

Outline

- Java and JavaBeans
- JavaBean characteristics
- Enterprise JavaBeans
- JavaBeans & Java technologies
- Relation to software architecture
- Conclusion

JavaBean Characteristics

- Introspection
- Customization
- Properties
- Events
- Persistence
- Packaging
- Methods

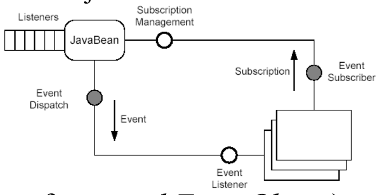


Methods

- Used at run time
  - Exposes services provided by JavaBean
  - Synchronous communication mechanism
- Same as regular Java methods
  - Lexical and syntactic rules, always *public*
  - Invocation semantics
- Excludes certain name prefixes
  - *is*, *get*, *set*, *add*, *remove*

Events

- Central to JavaBeans
  - Represent the occurrence of an incident of interest
  - Loosely coupled, publish-subscribe interaction
    - Event listener – any interested Java object
    - Event producer – a JavaBean
- Subscription management
  - Add, remove listeners
- Event dispatch
  - Send event message (subclass of *java.util.EventObject*) to all listeners
  - Uses regular Java method calls



Properties

- Provide access to JavaBean state
  - Have name and type
  - Read only, read-write, or write-only
  - For customization at design time
  - To modify behavior at run time
- Defined as a set of methods
  - Naming patterns
  - Separate class behavior from interface
- Many types
  - single valued, multivalued, indexed, bound, constrained

Example JavaBean

```
public class Circle extends java.awt.Component implements java.io.Serializable {
      ...
      // Paint the circle on the screen
      public void paint(Graphics g)
      ...
      // Get the boolean Shown property
      public boolean isShown()
      ...
      // Get the Radius property
      public int getRadius()
      ...
      // Set the Radius property
      public void setRadius(int radius) throws PropertyVetoException
      ...
      // Get the Border Size property
      public int getBorderSize()
      ...
      // Set the Border Size property
      public void setBorderSize(int size)
```

Persistence

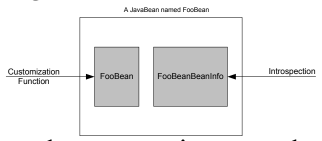
- For recreating JavaBean instances
  - Communicate design customization for execution
- Three mechanisms
  - Serialization
    - Virtual machine level, special logic
  - Externalization
    - Self management of internal state by JavaBean
  - Encoding
    - Standardized encoding of properties by a utility
- Portability of persistent information
- Efficiency of storage and retrieval

Packaging

- Package related Java classes, persistent state, documentation, resources, etc.
- Special archive format
  - Java archive (Jar) file
  - Reduce clutter
  - Platform-independent
  - Uses ZIP compression
  - Additional manifest file to describe contents
- Deployment can produce previously customized JavaBean instances

Introspection

- Acquire design information about bean type
  - To compose applications, or composite bean
  - Automatic discovery of configuration/customization capabilities
- Introspection architecture
  - Fault-tolerant
  - Separate introspection info. reduces run time overhead
  - Based on standard naming patterns of JavaBeans
- High-level design information
  - Unlike reflection only gives implementation details



15

Customization

- Only used at design time
  - Visual application design using JavaBeans
- Important capability afforded by visual editors
  - Construct visual editing interfaces through introspection
  - Connect different JavaBeans and events
  - Specify custom properties through property editors
  - Visualize designs
  - Generate Java source code
- JavaBeans can provide custom property editors

16

Outline

- Java and JavaBeans
- JavaBean characteristics
- Enterprise JavaBeans
- JavaBeans & Java technologies
- Relation to software architecture
- Conclusion

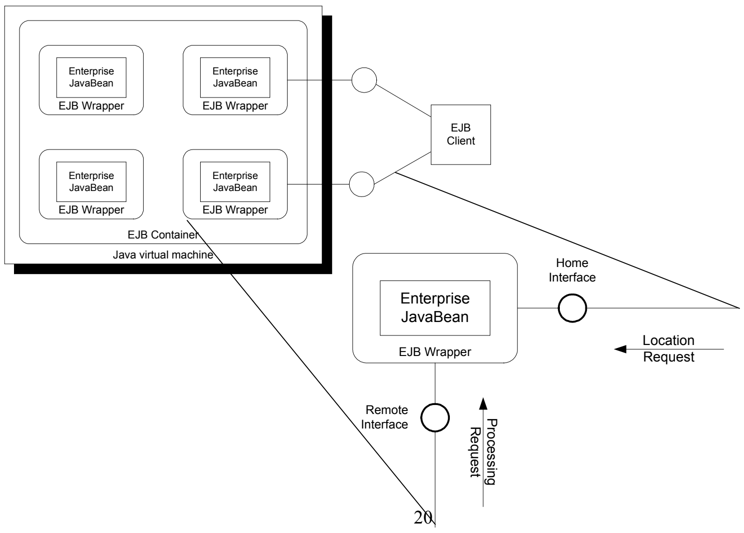
Enterprise JavaBeans

- Distributing JavaBeans
  – Network communication using RMI
  – Transparent invocation semantics
  – Serialization of parameters, return values
  – Using JNDI for object location
- Enterprise JavaBeans
  – Aimed at reducing TCO of business applications
  – Concurrently accessed by large number of users
  – Perform data access and information processing
  – Provide greater availability, security, resource mgmt.
  – Infrastructure services provided by EJB middleware

EJB Middleware Facilities

- Deployment
- Memory and instance management
- Thread management
- Communication management
- Security
- Location
- Messaging
- Invocation
- Persistence
- Transaction management

EJB Architecture



Outline

- Java and JavaBeans
- JavaBean characteristics
- Enterprise JavaBeans
- JavaBeans & Java technologies
- Relation to software architecture
- Conclusion

JavaBeans & Java Technologies

- JavaBeans
  - A generic component model
  - Applicable in both consumer and business applications
  - Most suited to the client environment
- Java technologies influenced by JavaBeans
  - Abstract windowing toolkit
  - Java database connectivity
  - JavaMail
  - Java management extensions

Abstract Windowing Toolkit

- Standard API for producing GUI applications in Java
- AWT beans are JavaBeans
  - Reusable software components
  - Can be visually manipulated in a builder tool
  - Can be customized for use in an application
  - Use methods, properties, events
- Java foundation classes
  - Lightweight visual components
  - Richer set of widgets

Java Database Connectivity

- JavaBean persistence often inadequate
  - Serialization, externalization, encoding
  - Insufficient durability and integrity
  - Memory inefficient
- Bridge JavaBeans with database technology
  - ACID transactions
  - Establish connections, perform queries, manipulate result sets
  - Events and properties on the above
  - Enables use of components for data access
  - Simplifies application development

JavaMail

- Used in mail applications
  - Based on standard mail protocols
    - SMTP, POP3, IMAP, NNTP
  - Enable componentization of mail clients
- Non-graphical JavaBeans
  - Design time specification of connection properties
  - Run time management of connections and mail stores
  - Events in response to user actions and data received from mail servers

Java Management Extensions

- For managing applications and networks
  - Fine-grained instrumentation – components
  - Coarse-grained instrumentation – application
  - Obtain information about running components
  - Control run-time behavior
- MBeans
  - Follow special naming patterns
  - Implement management interfaces
  - Contain properties, methods, and notifications
  - Define meta data as introspection information
  - Register with MBean servers to provide remote access

Outline

- Java and JavaBeans
- JavaBean characteristics
- Enterprise JavaBeans
- JavaBeans & Java technologies
- Relation to software architecture
- Conclusion

Relation to Software Architecture

- Components
  - EJB – session, entity, message-driven beans
  - AWT – GUI widgets
  - JDBC – connections, queries, result sets
  - JavaMail – folder, store, message, transport
  - JMX – MBean
- Connectors
  - EJB container provides facilities through EJB *wrapper*
  - RMI, asynchronous messages in EJB
  - JavaBeans, AWT, JDBC, JavaMail use events, properties and methods for interaction

Support for Architectural Styles

- JavaBeans support publish-subscribe style
  - Interaction through events
  - Components are producers and consumers of events
- EJBs are in the distributed object style
  - Synchronous and asynchronous communication
- JavaBeans and C2 style
  - Rosenblum & Natarajan
  - C2 style-aware visual builder tool for JavaBeans
  - Components are JavaBeans
  - Interaction through events
  - Configurations adhere to C2 style

Outline

- Java and JavaBeans
- JavaBean characteristics
- Enterprise JavaBeans
- JavaBeans & Java technologies
- Relation to software architecture
- Conclusion

Conclusion

- Component-based software
  - Promising for object technologies
  - Address challenges facing large, complex software systems
- JavaBeans
  - Loosely coupled architectural style
  - Basis for Java component technologies
- Java and the Internet
  - Best of programming and delivery worlds