Reflection and JavaBeans

Topics

- Reflection
- JavaBeans

Objectives

- Be exposed to the Java Reflection system
- Learn about the JavaBeans component technology, and how to use it in non-visual programming

Reflection

- **Java run-time system maintains run-time type information**
  - **Fields**
  - **Methods**
  - **Constructors**
- **Type information is available through the** Class **class**

The Class Class

- Obtained through the getClass method of the Object class
- Offers methods to obtain; fields, methods, constructors
  - Each represented by their own class

Fields

- **Obtained using** Class **methods:**

  getDeclaredField(String name)

  getDeclaredFields()

  getField(String name)

  getFields()

Field Class

- **Provides access to field details**

  getDeclaringClass()

  getModifiers()

  getType()

  getName()

  get(Object obj)

  get<Type>(Object obj)

  set(Object obj, Object value)

  set<Type>(Object obj, <type> value)

Methods

- **Obtained using** Class **methods:**

```
getDeclaredMethods()
getMethods()
getDeclaredMethod( String name,
                   Class[] paramTypes )
getMethod( String name,
           Class[] paramTypes )
```

Method Class

- **Provides access to field details**

  getDeclaringClass()

  getExceptionTypes()

  getModifiers()

  getName()

  getParameterTypes()

  getReturnType()

  invoke(Object obj, Object[] args)

Constructors

- **Obtained using** Class **methods:**

  getConstructor(Class[] paramTypes)

  getConstructors()

  getDeclaredConstructor(Class[] paramTypes)

  getDeclaredConstructors()

Constructor Class

- **Provides access to field details**

  getDeclaringClass()

  getExceptionTypes()

  getModifiers()

  getName()

  getParameterTypes()

  newInstance(Object[] args)

More Class Methods

- **Additional useful** Class **methods:**

  forName(String className)

  getModifiers()

  getInterfaces()

  getPackage()

  isArray()

  isInstance(Object obj)

  isInterface()

  isPrimative()

  newInstance()

Package Class

- Not particularly useful, primarily concerned with implementation and specification versions
- The name of the package may be obtained

```
getName()
```

Resources

- Ancillary files used by an application
  - Loaded from the class path, including jar files
- Methods for loading resources
  - Two Class methods:

    getResource(String name)

    getResourceAsStream(String name)

    - Paths are relative to the class' package
  - Two ClassLoader methods

    getSystemResource(String name)

    getSystemResourceAsStream(String name)

Example

<ImageLoader.java>

JavaBeans

JavaBeans

- Java’s component object model
- Programming Legos
  - JavaBeans may be “plugged” together to create programs
  - All AWT and Swing components
  - Non-UI classes
- Supporting class libraries

What are JavaBeans

- "… reusable software components that can be manipulated visually in a builder tool."
- In practical terms, beans support:
  - Introspection
    - Design patterns – really signature conventions
  - Customization
  - Events
  - Properties for customization and programmatic use
  - Persistence
- Visual beans extend java.awt.Component

Bean Features

- Properties
  - Named attributes associated with a bean
  - By default identified by design pattern
- Methods
  - Normal Java methods
  - Callable from other beans or the hosting environment
- Events
  - A means for one bean to notify another that something interesting has occurred

Design v. Run-time

- Design-time
  - Bean runs inside IDE
  - Provides design info about the bean
  - Allows customizations to be performed
- Run-time
  - Not as concerned with design info or customization
  - Code supporting Design and run-time may be implemented in separate classes

Bean Developer Assumptions

- Have access to introspection and reflection API's
- Beans may be serialized and de-serialized
- Multithreaded environment

Using JavaBeans

- Assemble beans
  - Create instantiations of the bean
- Customize beans
  - Set properties
- Connect the beans
  - Implement event listeners which invoke methods on other beans

Properties

- Named attributes of object
- May be accessed programmatically
- May be set as part of bean customization
- Will be persistent
- By default identified according to design pattern
- Always accessed by parent objects methods
- Indexed properties support multi-value

Property Design Patterns

- Naming conventions for getter and setter
  - Simple properties

```
public <PropertyType> get<PropertyName>();
public void set<PropertyName>( <PropertyType> value );
```

  - Optional, alternative getter for boolean properties

```
public boolean is<PropertyName>();
```

  - Indexed properties have additional methods for accessing the individual array elements

```
public <PropertyElement> get<PropertyName>(int a);
public void set<PropertyName>(int a,
                    <PropertyElement> value);
```

Event Model

- Event notifications are propagated to listeners by invocation of their methods
- Each event notification is defined as a distinct method
  - Grouped in event listener interfaces
- Event sources define registration methods
  - Methods accept references to a specific event listener interface

Events

- Event object encapsulates event state info
  - Derived from java.util.EventObject
  - By convention class name ends with “Event”

Example Event

```
public class PriceChangeEvent extends java.util.EventObject {
   protected String mTicker;
   protected int mPrice;
   PriceChangeEvent(Object source, String ticker, int price) {
      super( source );
      mTicker = ticker;
      mPrice = price;
   }

   public String getTicker() {
      return mTicker;
   }

   public int getPrice() {
      return mPrice;
   }
}
```

Event Listeners

- Event listener interfaces define interface for delivering events to interested objects
  - Extend java.util.EventListener
  - By convention class name ends with “Listener”
  - Methods should conform to design pattern

    void <*eventOccuranceMethodName*>(<*EventobjectType*> event);
  - Methods may throw checked exceptions
- An example

```
public interface PriceChangeListener
extends java.util.EventListener {
  void priceChanged( PriceChangeEvent event );
}
```

Event Sources

- Must implement interfaces for registering and de-registering listener
  - Registration methods for multicast event sources should conform to design pattern

    public void add<*ListenerType*>(<*ListenerType*> listener);

    public void remove<*ListenerType*>(<*ListenerType*> listener);
  - Methods should be synchronized
  - Unicast event source's add method may throw java.util.TooManyListenersException

EventListenerList

- Provides a list for managing listeners
  - Associates listeners with the listener interface they are registered with
  - May be used to manage any number of listeners and listener interfaces

EventListenerList Usage

```
private EventListenerList listenerList = new EventListenerList();
.
.
.
public void addPriceChangeListener(PriceChangeListener listener) {
    listenerList.add(PriceChangeListener.class, listener);
}

public void removePriceChangeListener(PriceChangeListener listener) {
    listenerList.remove(PriceChangeListener.class, listener);
}

private void firePriceChangeEvent(PriceChangeEvent evnt) {
    PriceChangeListener [] listeners;
    listeners = listenerList.getListeners(PriceChangeListener.class);
    for (PriceChangeListener listener : listeners) {
        listener.priceChanged(evnt);
    }
}
```

Bound Properties

- **Provide change event notification through** java.beans.PropertyChangeListener
- **A single event notification method**

```
void propertyChange( PropertyChangeEvent evt );
```

- **Bean must have proper listener registration methods**

```
void addPropertyChangeListener(PropertyChangeListener l);
void removePropertyChangeListener(PropertyChangeListener l);
```

- **Bean must generate proper change events**

```
PropertyChangeEvent(Object source, String property,
                    Object oldValue, Object newValue);
```

Constrained Properties

- **Allow interested objects to veto a property change**
  - **Set methods of constrained properties throw** java.beans.PropertyVetoException
  - **Set methods of constrained properties notify** java.beans.VetoableChangeListeners **of impending property change**
  - **Objecting listener throws** java.beans.PropertyVetoException

Constrained Properties

- **The** VetoableChangeListeners **has a single event notification method**

  void vetoableChange(PropertyChangeEvent evt);

- **Bean must have listener registration methods**

  void addVetoableChangeListener(VetoableChangeListener l);

  void removeVetoableChangeListener(VetoableChangeListener l);

Monitoring Specific Properties

- Specific properties may be bound or constrained
- Use property specific listener registration methods

```
void addPropertyChangeListener( String propertyName,
                               PropertyChangeListener l );
void removePropertyChangeListener( String propertyName,
                                  PropertyChangeListener l );
void addVetoableChangeListener( String propertyName,
                               VetoableChangeListener l );
void removeVetoableChangeListener( String propertyName,
                                  VetoableChangeListener l );
```

Monitoring Specific Properties

- Alternative property specific listener registration methods are provided

```
void add<PropertyName>Listener( PropertyChangeListener l );
void remove<PropertyName>Listener( PropertyChangeListener l );
void add<PropertyName>Listener( VetoableChangeListener l );
void remove<PropertyName>Listener( VetoableChangeListener l );
```

Supporting Classes

- Two classes are provide to assist in managing event listeners and event notification
  - Methods for registering listeners
  - Methods for firing events

  java.beans.PropertyChangeSupport

  java.beans.VetoableChangeSupport
- Instantiate instances of these classe and delegate to them

Introspection

- The process of determining the properties, events and methods supported by a bean
- By default reflection is used, based on design patterns
- The java.beans.BeanInfo interface may be used to precisely specify a beans features
  - The BeanInfo class will have the same name as the bean suffixed with "BeanInfo"

BeanInfo

- Provides methods for obtaining information about a bean
  - Display icon
  - Feature descriptors, all deriving from FeatureDescriptor
    - Programatic name
    - Display name
    - Short description
    - Others…

Descriptors

- BeanDescriptor
  - Bean class
  - Customizer class
- PropertyDescriptor
  - Name
  - Type
  - Is bound/constrained
  - Editor class
  - Read/write methods

Descriptors

- EventSetDescriptor
  - Listener registration methods
  - Listener event notification methods
  - Listener class
  - Is unicast
- MethodDescriptor
  - The method itself
  - Parameter descriptors for the method parameters