CMSC433, Spring 2001
JavaBeans, with examples

Alan Sussman
May 10, 2001

---

Administrivia

• Project 6
  – for testing, register some known URLs, and try running 2 copies of your program, registering different URLs in each one
    • on different machines, or different ports on same machine (not ones for other students …)
• Practice questions for final posted soon
  – questions on those, or anything else, on Tuesday

Last time

• JavaBeans, with examples
  – Properties – get and set
  – Events
    • Beans fire events, typically within property set methods, and allow other Beans to register as listeners
    • listener Beans register and receive events
  – Bound and constrained properties
    • notify listeners and/or allow them to veto property changes
  – Serialization
    • to allow Beans to persist
    • default provided, but can be customized

Externalizable interface

• To get complete control over Bean’s serialization, implement Externalizable instead of Serializable
  – e.g., for writing/reading a specific file format
  – implement readExternal() and writeExternal() methods
  – these classes also require a no-argument constructor

BeanInfo Interface

• An alternative for a builder tool to using java.beans.Introspector and the core reflection API to discover properties, events, methods, etc. about a Bean
• A class that implements the BeanInfo interface explicitly exposes a Bean’s features

BeanInfo features

• BeanInfo capabilities
  – can expose only features Bean wants to expose
  – can expose some features and allow using low-level reflection to expose others
  – associate an icon with a Bean
  – segregate features into normal and expert types
  – provide more descriptive name, or additional info, about a bean feature
BeanInfo methods

- PropertyDescriptor[] getPropertyDescriptors();
- MethodDescriptor[] getMethodDescriptors();
- EventSetDescriptor[] getEventSetDescriptors();
- Other descriptors for the Bean’s class type and name (BeanDescriptor), and for method parameters (ParameterDescriptor)

Creating a BeanInfo class

- Name the BeanInfo class
  - append BeanInfo to Bean class name
- Subclass SimpleBeanInfo
  - adaptor class with all methods returning null, or a no-op value, so only override methods needed
- Override appropriate methods to return properties, methods, events want to expose
  - if leave a feature out, won’t be exposed
  - if feature’s getter method (e.g., getMethodDescriptor) returns null, then low-level reflection used for that feature
- Optionally associate icon with the Bean
- Specify the Bean class

BeanInfo class for ExplicitButton

```java
public class ExplicitButtonBeanInfo extends SimpleBeanInfo {
    public PropertyDescriptor[] getPropertyDescriptors() {
        try {
            PropertyDescriptor background = new PropertyDescriptor("background", beanClass);
            PropertyDescriptor foreground = new PropertyDescriptor("foreground", beanClass);
            PropertyDescriptor font = new PropertyDescriptor("font", beanClass);
            PropertyDescriptor rv[] = {background, foreground, font};
            return rv;
        } catch (IntrospectionException e) {
            throw new Error(e.toString());
        }
    }
}
```

Example, cont.

```java
public java.awt.Image getIcon(int iconKind) {
    if (iconKind == BeanInfo.ICON_MONO_16x16 ||
        iconKind == BeanInfo.ICON_COLOR_16x16) {
        java.awt.Image img = loadImage("EButtonIcon16.gif");
        return img;
    }
    if (iconKind == BeanInfo.ICON_MONO_32x32 ||
        iconKind == BeanInfo.ICON_COLOR_32x32) {
        java.awt.Image img = loadImage("EButtonIcon32.gif");
        return img;
    }
    return null;
}
```

Example, finished

```java
public BeanDescriptor getBeanDescriptor() {
    return new BeanDescriptor(beanClass);
}
```

Controlling exposed features

- Base class features not exposed
  - use BeanInfo.getAdditionalBeanInfo
- Properties, events, methods without descriptors not exposed
- Low-level reflection used for features with getter methods returning null