Contents

- Introduction
- Advantages
- JavaBeans vs. Class Libraries
- JavaBeans Concepts
- JavaBean Characteristics

Contents

- Key Concepts
- Security Issues
- JavaBeans and Threads
- Beans Development Kit (BDK)
- Properties

Introduction

- A Java Bean is a software component that has been designed to be reusable in a variety of different environments
- There is no restriction on the capability of a Bean. It may perform a simple function, such as obtaining an inventory value, or a complex function, such as forecasting the performance of a stock portfolio

Introduction

- A Bean may be visible to an end user. One example of this is a button on a graphical user interface
- A Bean may also be invisible to a user. Software to decode a stream of multimedia information in real time is an example of this

Advantages

- A Bean obtains all the benefits of Java's "write-once, run-anywhere" paradigm
- The properties, events, and methods of a Bean that are exposed to another application can be controlled
- Auxiliary software can be provided to help configure a Bean

Advantages

- The configuration settings of a Bean can be saved in persistent storage and restored at a later time

- A Bean may register to receive events from other objects and can generate events that are sent to other objects

JavaBeans vs. Class Libraries

- Beans are appropriate for software components that can be visually manipulated

- Class libraries are good for providing functionality that is useful to programmers, and doesn't benefit from visual manipulation

JavaBeans Concepts

- A component is a self-contained reusable software unit
- Components expose their features (public methods and events) to builder tools
- A builder tool maintains Beans in a palette or toolbox
- You can select a bean from the toolbox, drop it in a form, and modify its appearance and behavior
- Also, you can define its interaction with other beans

JavaBean Characteristics

- a public class with 0-argument constructor
- it has properties with accessory methods
- it has events
- it can be customized
- its state can be saved
- it can be analyzed by a builder tool

Key Concepts

- A builder tool discover a bean's features by a process known as *introspection*.
  - Adhering to specific rules (design pattern) when naming Bean features
  - Providing property, method, and event information with a related Bean Information class
- Properties (bean's appearance and behaviour characteristics) can be changed at design-time

Key Concepts

- Properties can be customized at design-time. Customization can be done:
  - using property editor
  - using bean customizers
- Events are used when beans want to intercommunicate
- Persistence: for saving and restoring the state
- Bean's methods are regular Java methods

Security Issues

- JavaBeans are subject to the standard Java security model
- The security model has neither extended nor relaxed
- If a bean runs as an untrusted applet then it will be subject to applet security
- If a bean runs as a stand-alone application then it will be treated as a normal Java application

JavaBeans and Threads

- Assume your beans will be running in a multi-threaded environment
- It is your responsibility (the developer) to make sure that their beans behave properly under multi-threaded access
- For simple beans, this can be handled by simply making all methods

Beans Development Kit (BDK)

- To start the BeanBox:
  - run.bat (Windows)
  - run.sh (Unix)
- ToolBox contains the beans available
- BeanBox window is the form where you visually wire beans together
- Properties sheet: displays the properties for the Bean currently selected within the BeanBox window

Beans Development Kit (BDK)

```
import java.awt.*;
import java.io.Serializable;
public class FirstBean extends Canvas implements Serializable {
    public FirstBean() {
      setSize(50,30);
      setBackground(Color.blue);
   }
}
```

Properties

- Bean’s appearance and behavior -- changeable at design time.
- They are private values
- Can be accessed through getter and setter methods
- getter and setter methods must follow some rules -- design patterns (documenting experience)

THANK YOU