JavaBeans

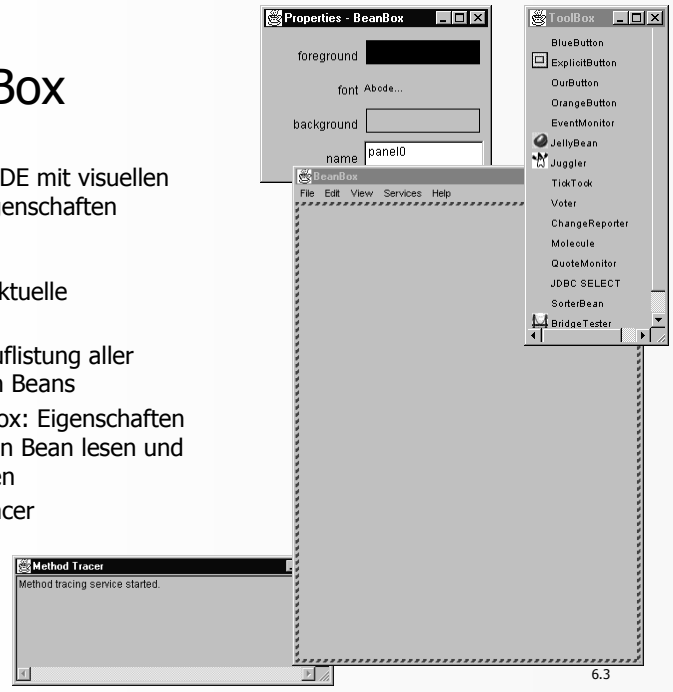
Java inproc Komponenten

Übersicht

- Komponentenbasierte Programmierung in Java
- JavaBeans werden in der JVM der Anwendung ausgeführt
- JavaBeans sind Komponenten, die
  - über Zugriffsklassen verfügen
  - bestimmte Namenskonventionen bei den Methoden einhalten
  - über Dialogboxen konfigurierbar sind
  - neuen Programmiermethoden (Visual Programming) zugänglich sind
- Vergleichbar mit
  - Controls in OLE, COM, ActiveX (inproc)
  - Konfiguiermöglichkeiten in Visual Studio

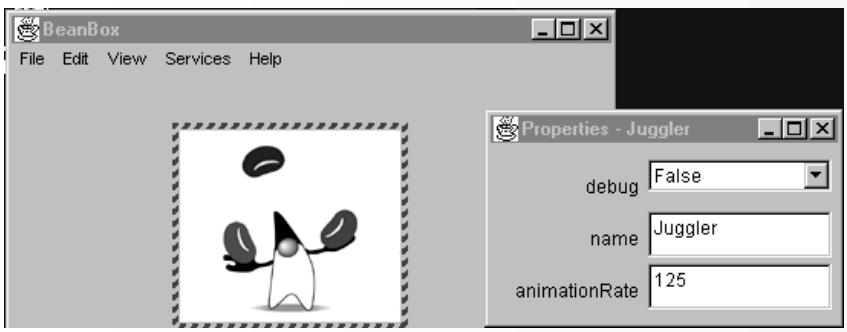
Die BeanBox

- Prototyp einer IDE mit visuellen Programmiereigenschaften
- 4 Fenster
  - BeanBox: Aktuelle Anwendung
  - ToolBox: Auflistung aller verfügbaren Beans
  - PropertiesBox: Eigenschaften der aktuellen Bean lesen und konfigurieren
  - MessageTracer



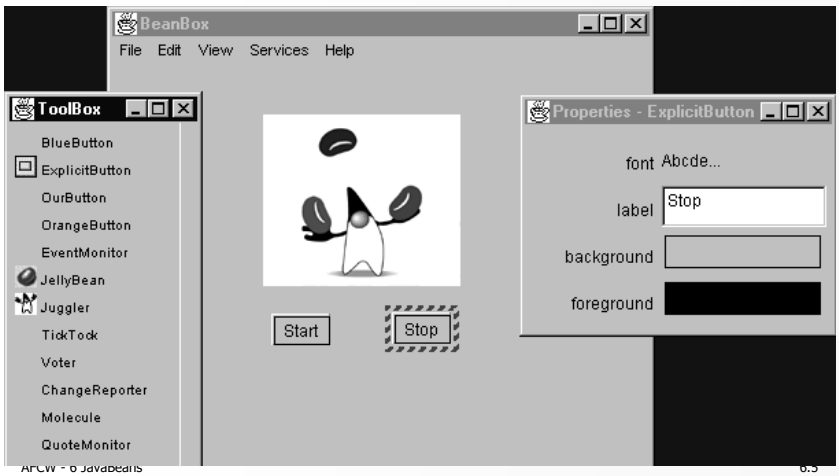
Beispiel: JugglerBean (1)

- JugglerBean aus Toolbox auswählen und plazieren



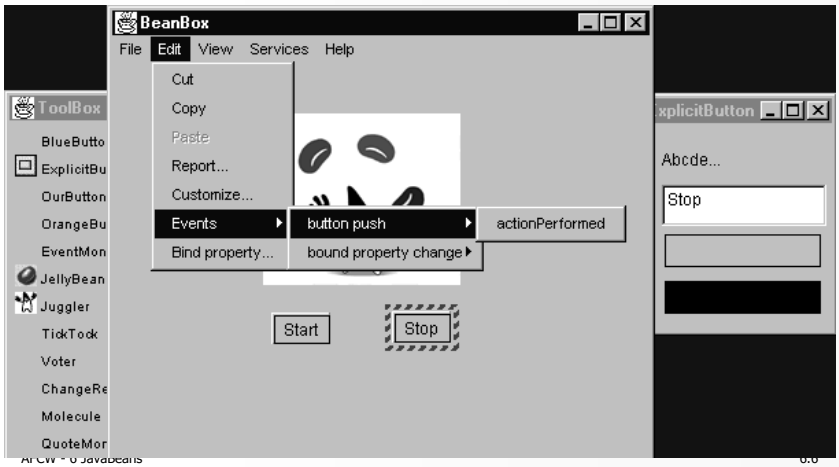
JugglerBean (2)

- 2 ExplicitButton mit „Start" und „Stop" hinzufügen



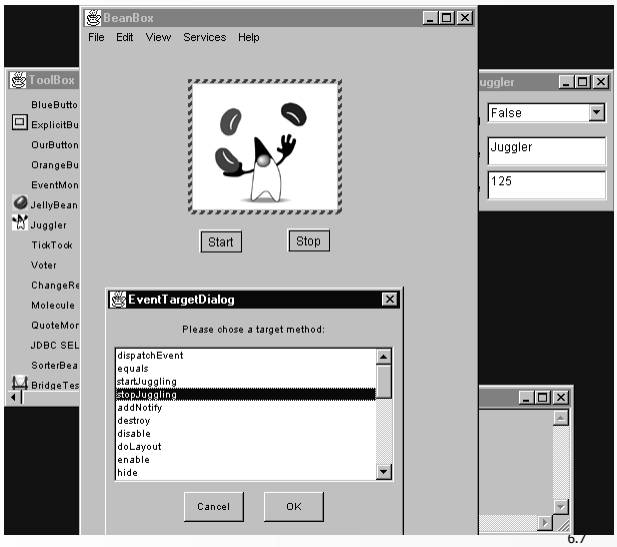
JugglerBean (3)

- „Stop": Edit->Events->button push->actionPerformed



JugglerBean (4)

- ... mit JugglerBean verknüpfen



JugglerBean (5)

- Automatisch generierter Code

```
package tmp.sunw.beanbox;
import sunw.demo.juggler.Juggler;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class ___Hookup_1653fe1423 implements
java.awt.event.ActionListener, java.io.Serializable {
    public void setTarget(sunw.demo.juggler.Juggler t) {
        target = t;
    }

    public void actionPerformed(java.awt.event.ActionEvent arg0) {
        target.stopJuggling(arg0);
    }

    private sunw.demo.juggler.Juggler target;

}
```

Eine eigene Bean

- Beispiel aus „Core Java 2, Volume II, Advanced Features"
- ImageViewerBean
  - Darstellung einer Bilddatei

ImageViewerBean

```
import java.awt.*;
import java.io.*;
import javax.swing.*;

public class ImageViewerBean
    extends JPanel
    implements Serializable {

    ...

    private static final int MINSIZE = 50;
    private Image image = null;
    private String fileName = "";

}
```

`setFileName()` und `getFileName()`

```
public void setFileName(String f) {
    fileName = f;
    image = Toolkit.getDefaultToolkit().getImage(fileName);
    MediaTracker tracker = new MediaTracker(this);
    tracker.addImage(image, 0);
    try { tracker.waitForID(0); }
    catch (InterruptedException e) {}
    repaint();
}

public String getFileName() {
    return fileName;
}
```


... und der Rest

```
public void paint(Graphics g) {
    if (image == null) {
           g.drawRect(0, 0,getWidth()-1,getHeight()-1);
    }
    else
           g.drawImage(image, 0, 0, this);
}

public Dimension getPreferredSize() {
    if (image == null)
           return new Dimension(MINSIZE, MINSIZE);
    return new Dimension(
           image.getWidth(null),image.getHeight(null));
}
```

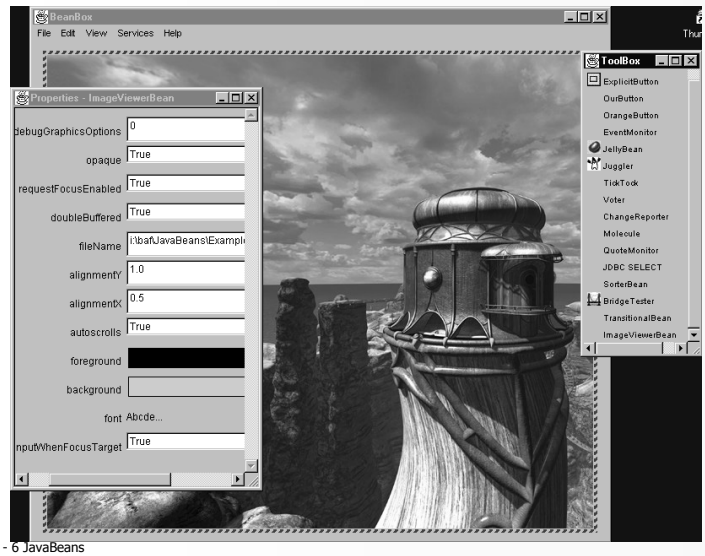
Vollständige Bean

- *.class Dateien für alle benötigten Klassen
- Manifsetdatei:
  - Welche Klassen gehören zur Bean
  - Beispiel für ImageViewerBean
    - ImageViewerBean.mf:

```
Name: ImageViewerBean.class
Java-Bean: True
```

- jar-File erzeugen:
  - `jar cfm ImageViewerBean.jar ImageViewerBean.mf *.class`

Bean testen



BeanProperties

- BeanBox erkennt Properties über Namen der Zugriffsfunktionen
- BeanProperty X
  - Verwendung sogenannter Decapitalization:
    - aus **`setMeineProp`** bzw. **`getMeineProp`** wird **`meineProp`**
- Angabe der Zugriffsfunktionen:
  - **`public X getX ()`**
    - Ausnahme: **`public boolean isX ()`**
  - **`public void setX ( X x )`**
- Nurleseeigenschaften: Keine setX()-Methode

Property-Arten

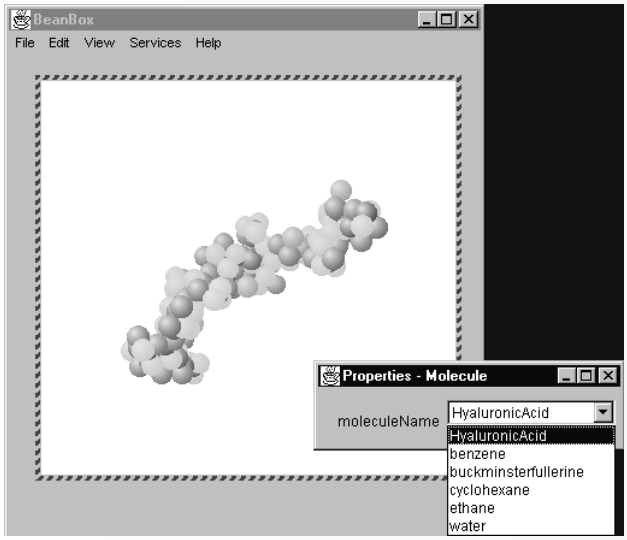
- Simple Property
  - Variable speichert einen einfachen Wert
- Indexed Property
  - Speicherung eines Feldes
  - Zugriffsfunktionen
    - **`X[] getX ()`**
    - **`void setX ( X[] x)`**
    - **`X getX ( int i )`**
      `void setX ( int i, X x )`

Property-Arten (contd.)

- Bound Property
  - Listener werden über Änderungen informiert
  - Zusätzlicher Implementierungsaufwand
    - Bean muß bei Änderung PropertyChange-Event senden
    - Verwaltung aller Listener:
      - void addPropertyChangeListener (...)
      - void removePropertyChangeListener (...)
    - Convenience-Klasse PropertyChangeSupport vorhanden
- Constraint Property
  - Bound Property mit Vetorecht der Listener

Property-Editoren

- BeanBox stellt für Grundtypen Editoren zur Verfügung
- Editoren für anwendungs-spezifische Methoden integrierbar



Alternativen

- BeanBox ist veraltete aber schöne Demonstration der wesentlichen Bean-Eigenschaften
  - Kleinere Inkompatibilitäten mit Java 1.4
- Neuere Fassung: Bean Builder
- Primärer Einsatz: Integrierte Entwicklungsumgebungen
  - JBuilder
  - NetBeans
  - …