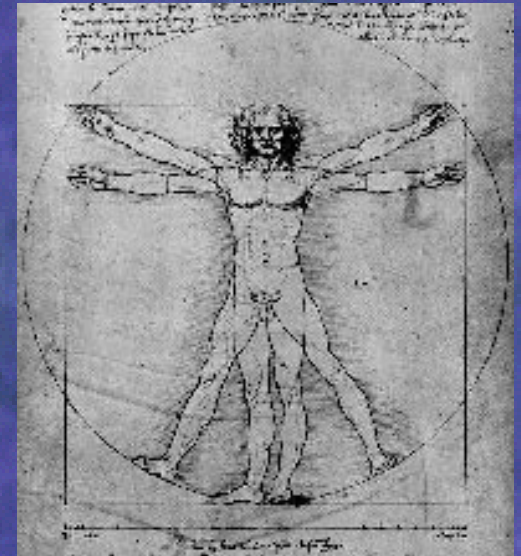


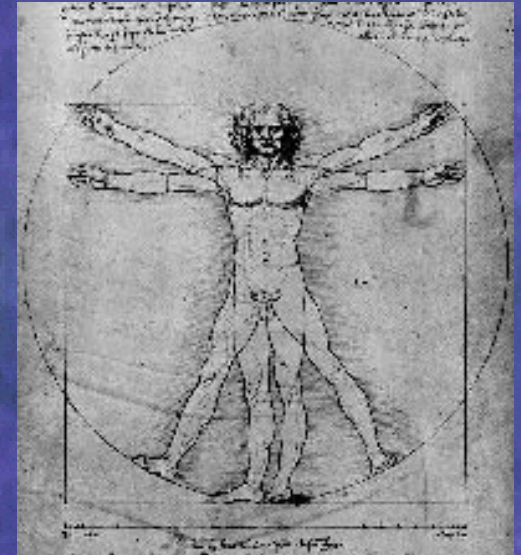
# Anatomy of a JavaBean

- Properties
- Events
- Methods
- Introspection/BeanInfo
- Customization
- Archiving/Persistence



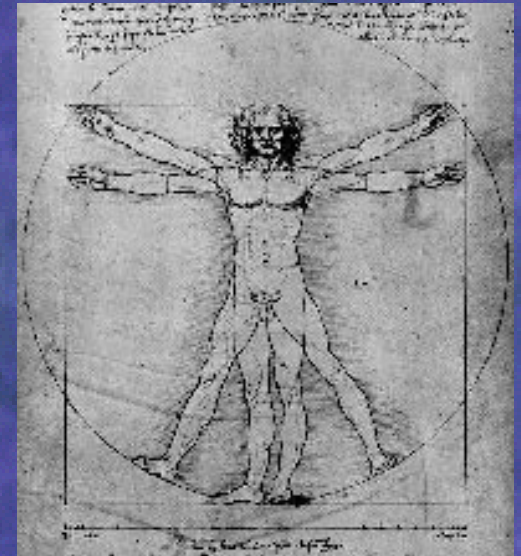
# Anatomy of a JavaBean- Properties

- Properties are discrete, named attributes of a Java Bean that can affect its appearance or its behavior.
  - Indexed properties- supports a range of values. (An array)
  - Bound properties- binds special behavior to property changes
  - Constrained properties- validates a change and reject it if it is inappropriate



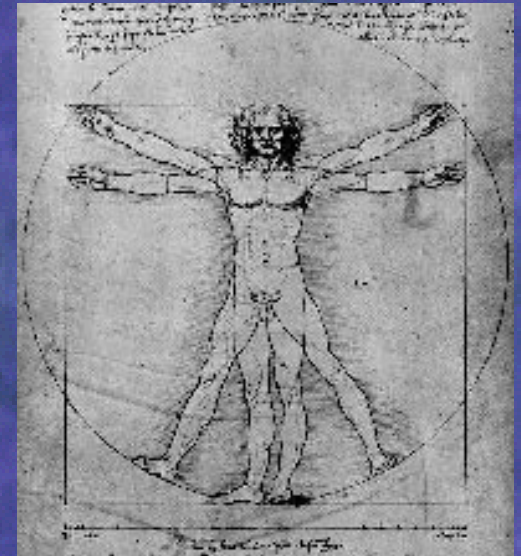
# Anatomy of a JavaBean- Events

- Events are a mechanism for propagating state change notifications between a source object and one or more target listener objects



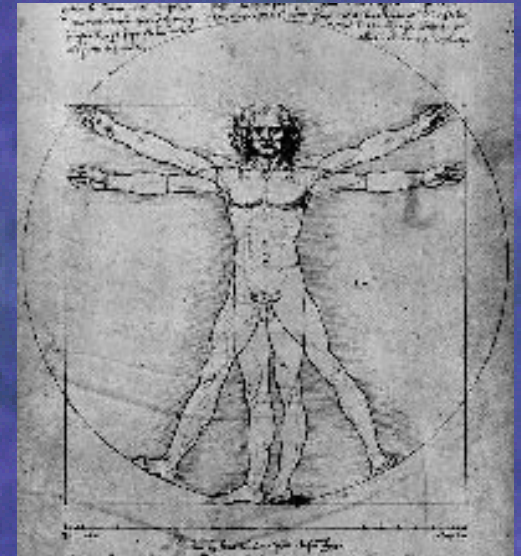
# Anatomy of a JavaBean- Methods

- Methods are used to implement Properties and Events. And... perform public operations on the bean.



# Anatomy of a JavaBean- Introspection

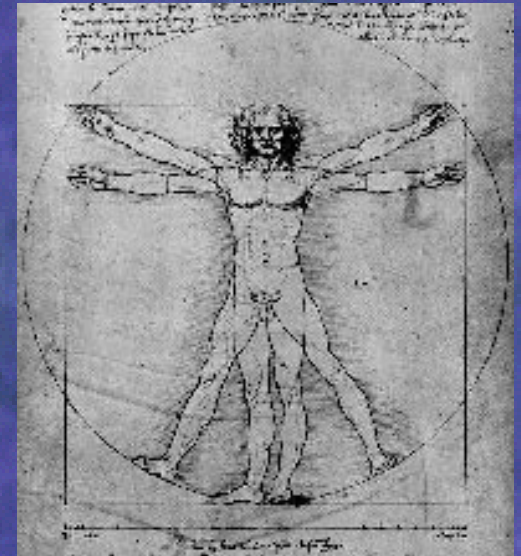
- Introspection - At runtime and in the builder environment we need to be able to figure out which properties, events, and methods a Java Bean supports.





# Anatomy of a JavaBean-Customization

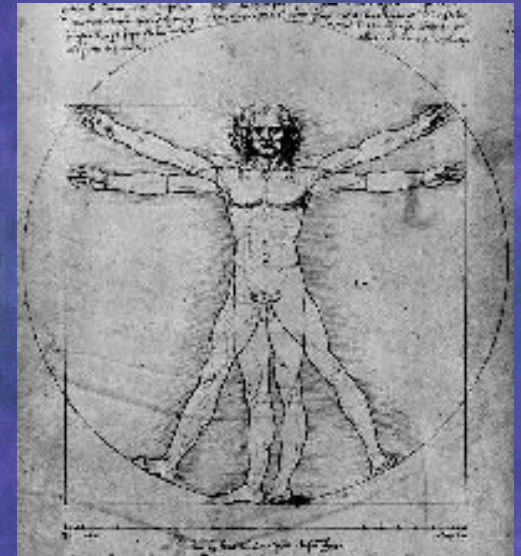
- Customization - When a user is composing an application in an application builder we want to allow them to customize the appearance and behavior of the various beans they are using.



# Anatomy of a JavaBean- Customization

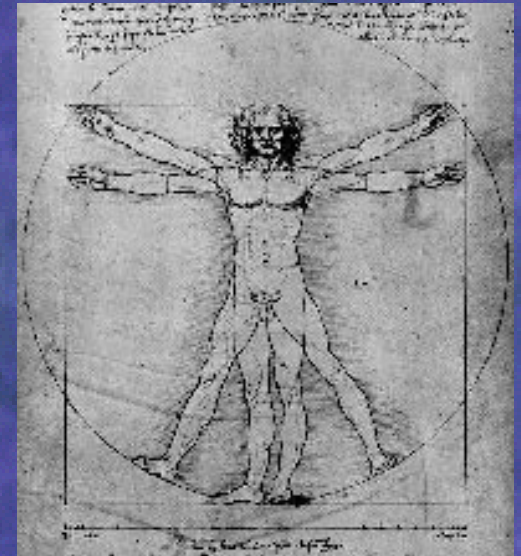
## ■ BeanInfo

- ▶ Works with Introspection, Overrides defaults
- ▶ Design time class
- ▶ Provides...
  - Bean icon
  - Prop. Sheet/Prop. Editor
  - Descriptors for P.E.M
  - Friendly names
  - Expert descriptor



# Anatomy of a JavaBean

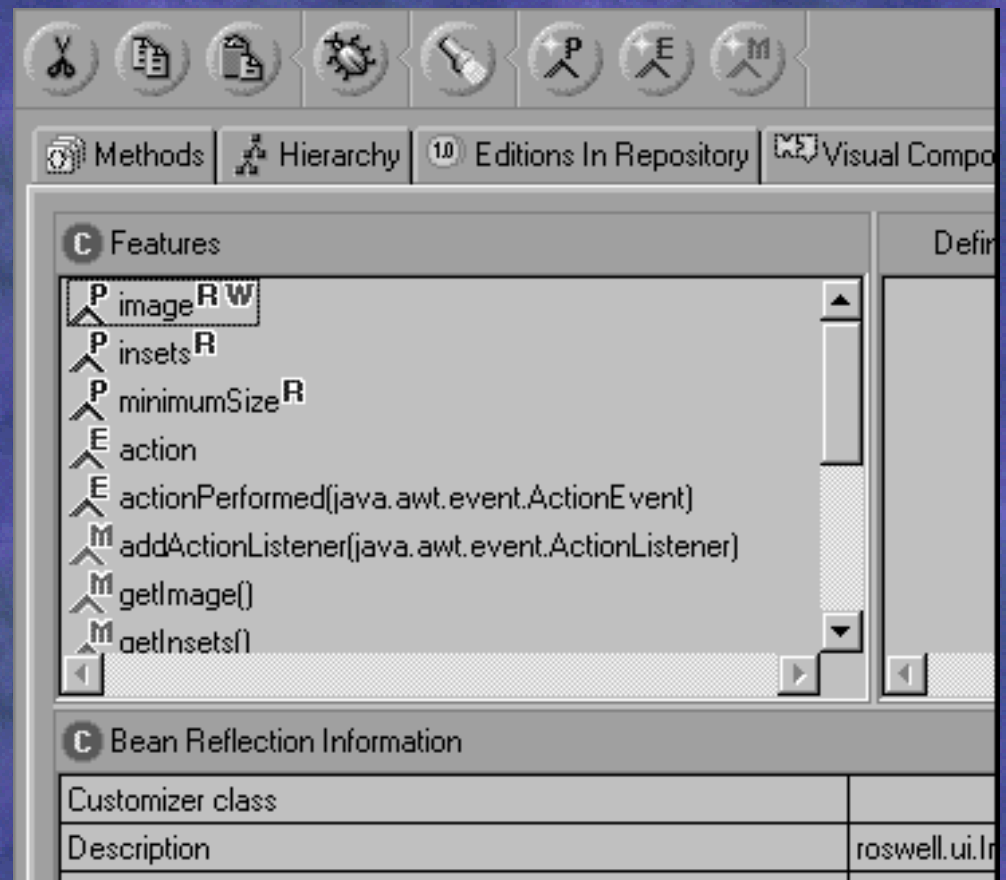
- Archiving
  - ▶ JAR (Java Archive)
  - ▶ Manifest (tags beans)
- Persistence- a bean can be customized in an application builder and then have its customized state saved away and reloaded later.





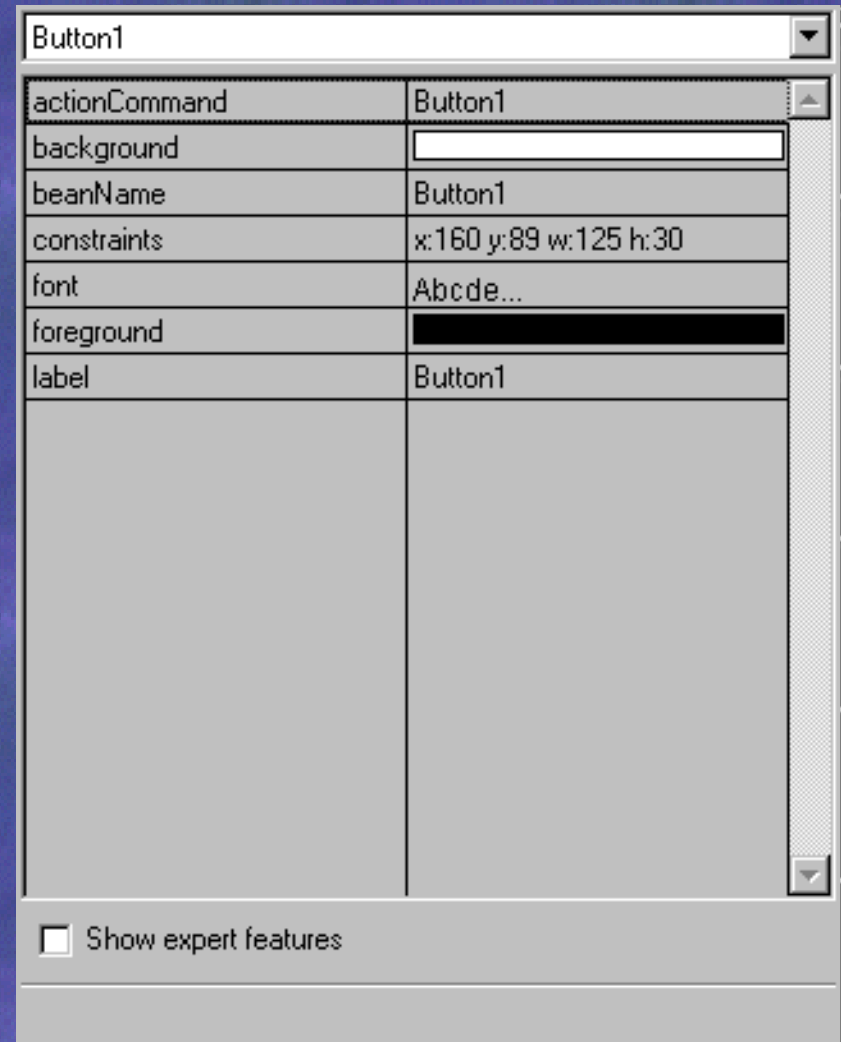
# How can a builder tool help?

- Create the Bean Anatomy
- Visually Manipulate the Bean
- Generate Code



# Demo- Visual Manipulation in VA-J

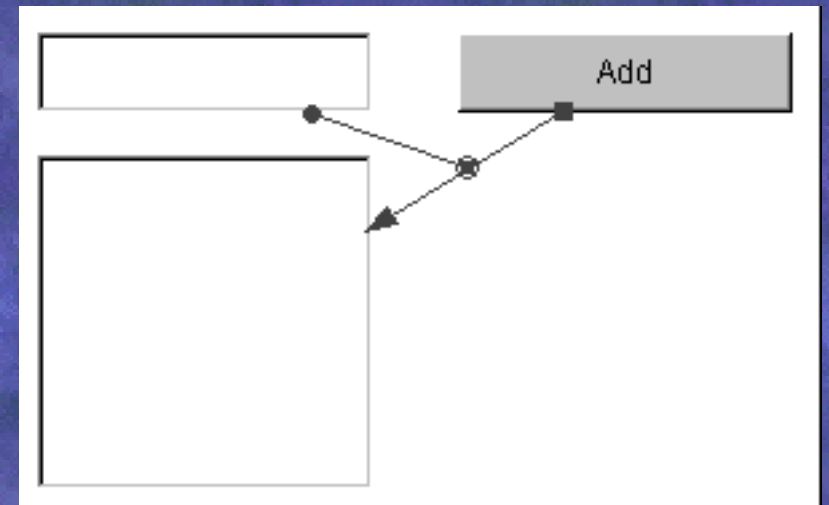
- Show how the Bean wizard can be used to create Bean P.E.M's.
- Show how a Beans can be Wired.



# Demo- Bean Evolution

*Example of Generic beans  
using VA-J*

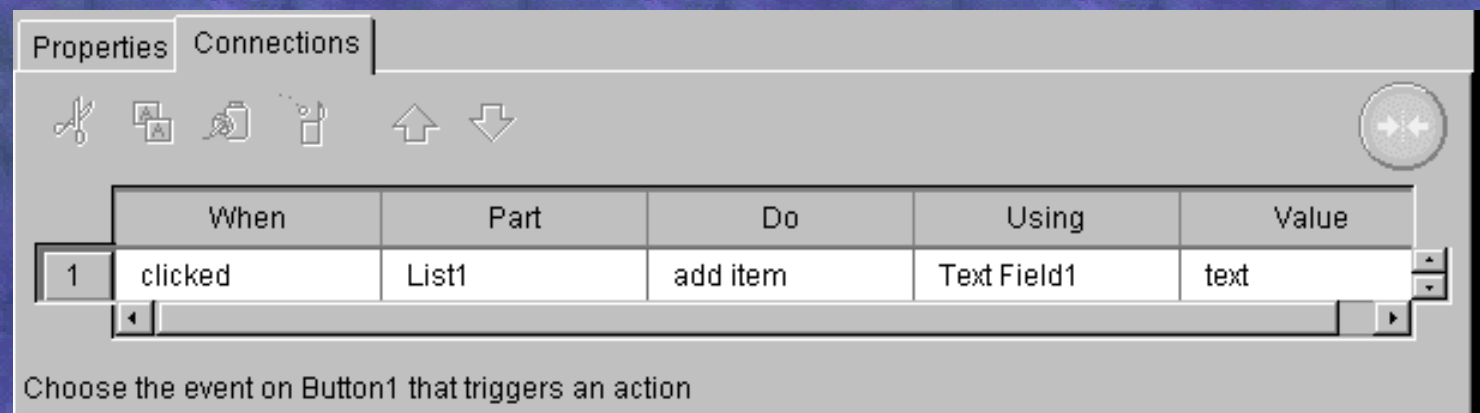
- ▶ Show Simple  
Button, EntryField,  
ListBox
- ▶ Show Wiring



# Demo- Bean Evolution

## *Example of Specific Bean using Lotus BeanMachine*

- ▶ BeanMachine has a verbose wiring model (attractive to non-programmers)
- ▶ The beans no longer deal with generic events. They now expose application events
- ▶ These beans interoperate well because they were designed to do so...



# Demo- The Auction

## The Catalog Page

- ▶ Wiring
- ▶ DB Access

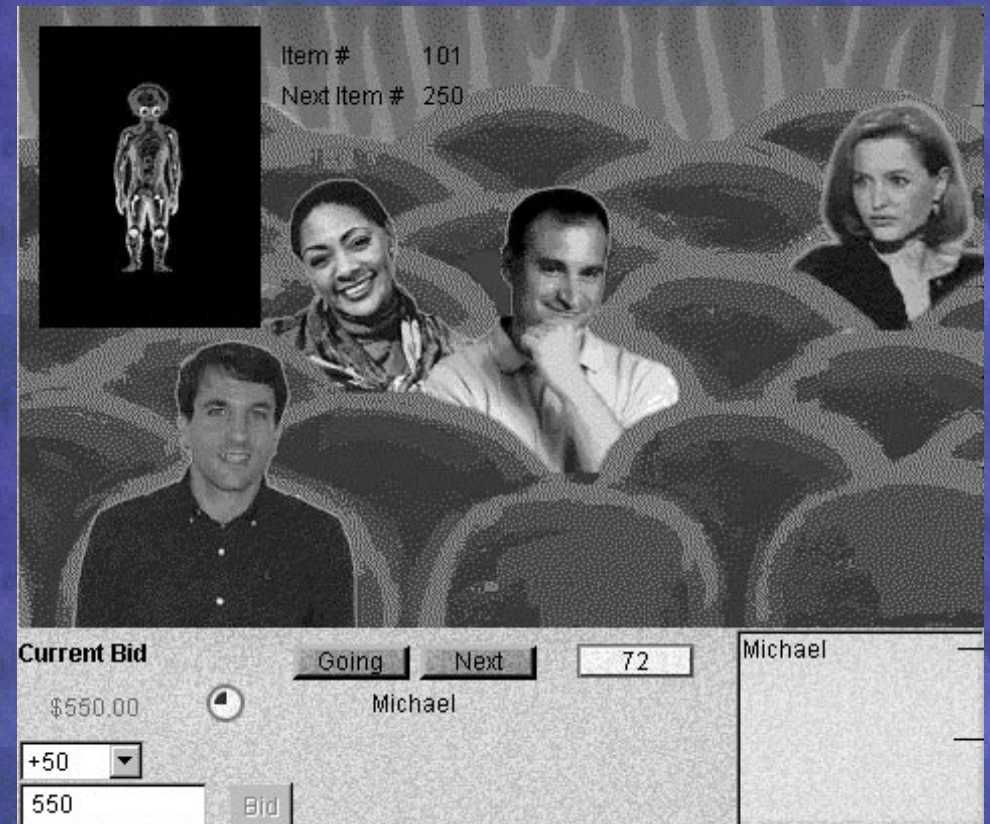




# Demo- The Auction

## The Auction/Auctioneer beans

- ▶ Extending Auction to create Auctioneer
- ▶ Add new beans to Auctioneer
- ▶ Wire new beans to create new behavior of Auctioneer



# Summary

- Beans are for builders, builders are for developers
- Beans are most "useful" when they are designed to work together (at a P.E.M. Level)
- The less "useful" the more code you write. (Need to write code to connect P.E.M's)
- IBM VA-Java and Lotus Bean Machine allow for the creation and manipulation of JavaBeans
- Beans run in Bean containers (applets, activeX)