

Java Beans

- JavaBeans são componentes que apresentam a vantagem de serem reutilizáveis e redistribuídos;
- Componentes JavaBeans podem ser utilizados em paletas de ferramentas gráficas para facilitar sua configuração e utilização.
- JavaBeans tem API: [java.beans.*](#);

Exemplos de Beans

- Componentes GUI;
- Geradores de gráficos/relatórios;
- Planilhas/Editores;
- Ferramentas de manipulação de dados;
- Qualquer código de software que possa funcionar independente e permita ser acoplado a outros aplicativos é um excelente candidato a JavaBean;

Principais Conceitos usados no ModeloJava Beans

- **Propriedades e Métodos:** JavaBeans divulgam um conjunto bem definido de propriedades e métodos, permitindo que as propriedades sejam alteradas e métodos sejam invocados;
- **Eventos:** JavaBeans divulgam um conjunto bem definido de eventos que produzem, e permitem que outros objetos registrem interesse na ocorrência destes eventos;
- **Introspecção e Reflexividade:** Um JavaBean usa um padrão de codificação que permite que uma ferramenta de edição visual interaja com o componente e deduza/altere suas características (eventos, propriedades e métodos) em build-time ou run-time;
- **Persistência e Empacotamento:** Capacidade de armazenar, recuperar ou transmitir um componente através de uma mídia digital (disco, conexão de rede, etc);

Propriedades e Métodos

- Propriedades são atributos privados que dispõem de pares de métodos no padrão
 - `XXX getXXXX();`
 - `void setXXXX(XXXX obj);`
- Estes métodos garantem o acesso (leitura) e o envio de mensagens (modificação) das propriedades.

Eventos

- JavaBeans usam o modelo de Eventos padrão do Java 1.1 ([Pattern Observer](#)). Resumindo:
- Quem dispara um evento é um Event Source;
- Um objeto interessado neste evento se cadastra junto ao Source e passa a ser um Listener;
- Todos os listeners são avisados do evento (através de um callback feito pelo Source);
- O evento pode encapsular ou não informação especial para os listeners;
- Uma referência à fonte do evento (o objeto Source) sempre está disponível dentro do evento.

Eventos - Exemplo

```
import java.beans.*;

class MyButton extends Button {
    private PropertyChangeSupport changes = new PropertyChangeSupport(this);

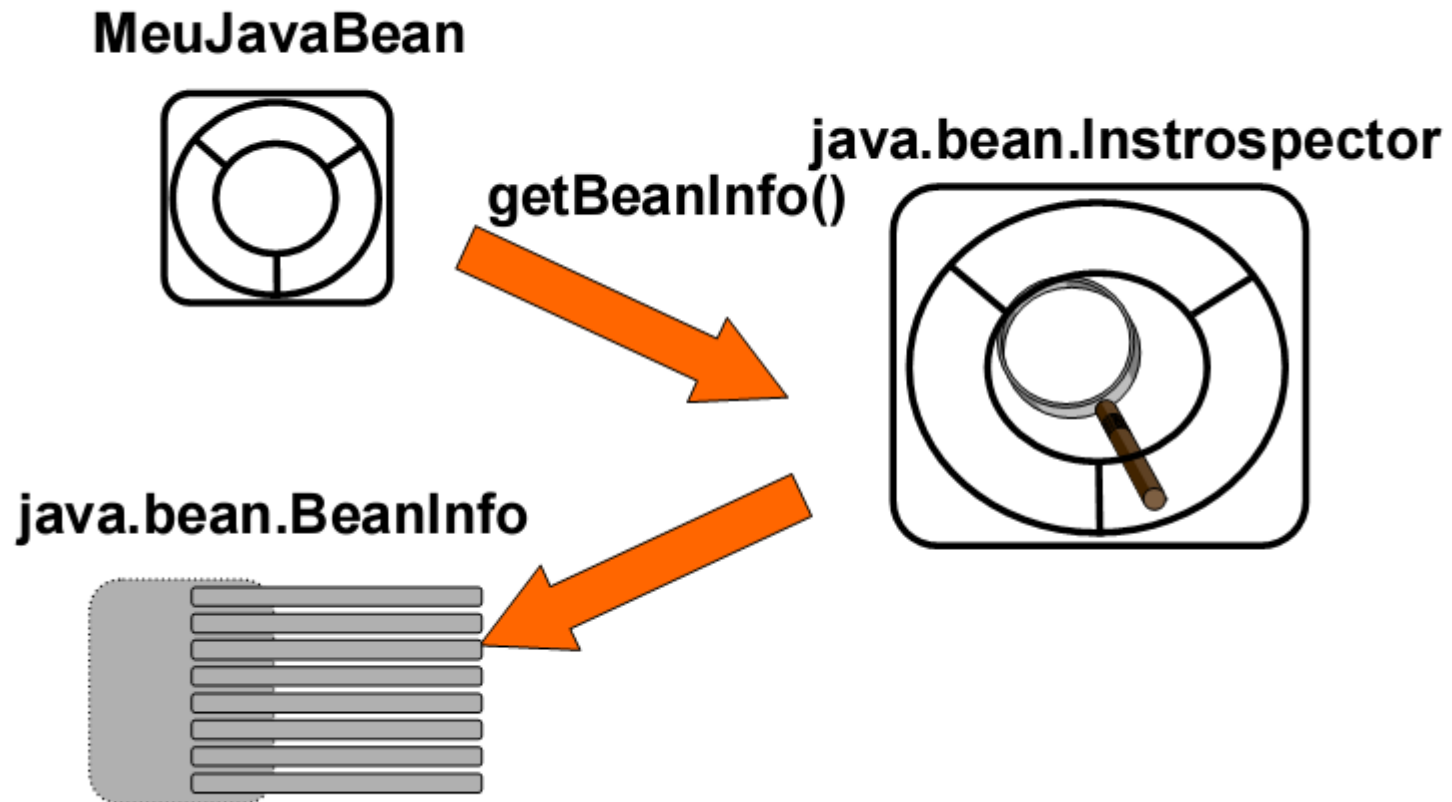
    public void addPropertyChangeListener(PropertyChangeListener l) {
        changes.addPropertyChangeListener(l);
    }
    public void removePropertyChangeListener(PropertyChangeListener l) {
        changes.removePropertyChangeListener(l);
    }
    public void setLabel(String newLabel) {
        String oldLabel = label;
        label = newLabel;
        changes.firePropertyChange("label", oldLabel, newLabel);
    }
}
```

Classes especiais para ajudar: [PropertyChangeSupport](#) e [VetoableChangeSupport](#)
Ajudam a mandar os eventos para todos os listeners e outras tarefas semelhantes

Um Bean Mínimo

- Tem um construtor default sem argumentos:
 - Para poder instanciar um Bean na ferramenta visual;
- Implementa a interface [java.io.Serializable](#):
 - Para poder salvar o Bean customizado em tempo de design:
 - Quando usamos componentização, objetos podem ser instanciados em tempo de design (usando componentes como fábricas) e seus atributos podem ser alterados em tempo de design;
 - É necessário ter serialização para poder obter os objetos novamente em tempo de execução.

Introspecção



Reflexividade

- Permite manipular classes, interfaces e objetos contidos na máquina virtual;
- Usada na construção de depuradores, ferramentas de construção de GUI, browsers de classes e bibliotecas modernas;
- Executar em run-time, várias operações que normalmente são programadas;
- `java.lang.reflect.*`;

Uso de Reflexão

- Determinar a classe de um objeto;
- Obter informação sobre modificadores da classe, seus métodos, campos, construtores e superclasses;
- Observar quais constantes e métodos fazem parte de uma interface;
- Carregar uma classe na máquina virtual, cujo nome só é conhecido em tempo de execução;
- Ler e modificar o valor do campo de um objeto, mesmo que o nome do campo só seja conhecido em tempo de execução;
- Invocar um método de um objeto, mesmo que o método só tenha sido conhecido em tempo de execução.

Plain Old Java Objects – POJO's

- Plain Old Java Objects, ou POJOs, são objetos Java que seguem um **desenho simplificado**. Um **JavaBean é um POJO que segue definições rígidas de estrutura** (construtor default sem argumentos e métodos que seguem o padrão de getters e setters para seus atributos).
- O termo ganhou aceitação por causa da necessidade de um termo comum e facilmente inteligível que contrasta com a complexidade dos frameworks de objetos. É mais atrativo do que o termo bean do Java devido à confusão gerada pela semelhança dos termos JavaBeans e os EJB (Enterprise JavaBeans).

Referências

- <http://java.sun.com/products/javabeans>
- <http://pt.wikipedia.org/wiki/JavaBean>
- http://pt.wikipedia.org/wiki/Plain_Old_Java_Objects;
- Google, as always ;)