Tutoriais

JavaBeans

por: Gleydson Lima
gleydson@jspbrasil.com.br

JavaBeans são componentes de software que são projetados para serem unidades reutilizáveis, que uma vez criados podem ser reusados sem modificação de código, e em qualquer propósito de aplicação, seja um applet, um servlet ou qualquer outra.

Um modelo de componente é definido como um conjunto de classes e interfaces na forma de pacotes Java que deve ser usado em uma forma particular para isolar e encapsular um conjunto de funcionalidades.

Os componentes JavaBeans são também conhecidos como Beans. Passaremos a usar esta nomenclatura no restante deste tutorial.

Neste tutorial não falaremos sobre todos os conceitos de JavaBeans, daremos atenção àqueles conceitos que mais são utilizados nas páginas JSP.

Regras para Escrever Beans

Para que esses componentes possam ser reconhecidos de forma geral, sua implementação deve seguir um conjunto de regras que serão usadas pelas ferramentas para instrospecção da classe, ou seja, o processo de descobrir quais as funcionalidades do Bean e disponibilizá-la para o usuário. Cabe ao usuário fazer a interface entre o componente e o restante da aplicação, criando assim um novo tipo de programador, **o programador de componenetes.**

```
import java.io.Serializable;

01 public class MyBean implements Serializable {
02
03      private int property1;
04      private boolean property2;
05
06        public MyBean() {
07
08        }
09
10      public void setProperty1(int property1) {
11           this.property1 = property1;
12      }
13
14      public void setProperty2(boolean property2) {
15           this.property2 = property2;
16      }
17
18      public int getProperty1() {
19           return property1;
20      }
21
22      public boolean isProperty2() {
23           return property2;
24      }
25  }
```

Esta classe é um exemplo do que devemos seguir para tornar uma classe um bean, ou seja, o conjunto de regras que devemos obdecer para tornar o compoenente reconhecível por ferramentas e usável.

Um bean, como modelo de componente, usa a idéia de encapsulamento. Portanto, as suas variáveis devem ser acessadas somente através de métodos. As variáveis de um Bean são suas propriedades.

A primeira regra que devemos respeitar é o construtor. O construtor de um bean deve ser sem parametros.

Outra regra que devemos seguir diz respeito ao nome dos métodos que darão acesso as propriedades do bean. Os métodos podem ser divididos em duas categorias:

Métodos de Leitura:
Os métodos de leitura de propriedades devem seguir a seguinte convenção:
```
public TipoDaPropriedade getNomeDaPropriedade()
```

Como mostrado no código acima a propriedade chamada **property1** tem o método de acesso **getProperty1** . Note que a primeira letra da propriedade deve ser minuscula enquanto a primeira letra depois do get deve ser em maiscula. A palavra TipoDaPropriedade deve ser substituida por o tipo da propriedade. Para variáveis booleanas vale o mesmo conceito, mas, ao invés do **get** usamos a palavra **is**. No exemplo temos o acesso a propriedade **property2** que é booleana como:
```
public boolean isProperty2()
```

Observação: Você pode não definir um método de leitura para a variável. Se isto for feito você não permitirá o acesso a ela.

Métodos de escrita:
Os métodos de escrita permitem ao usuário do componente modificar o conteúdo da propriedade. Ele segue a seguinte terminologia:
```
public void setNomeDaPropriedade(TipoDaPropriedade varName)
```

Nos métodos de escrita não há caso especial para variáveis booleanas que também podem ser modificadas através de métodos set.

Você pode não definir métodos de escrita para uma determinada variável, fazendo com que o usuário não seja capaz de modificar o conteúdo da propriedade.

Outra regra que um Bean deve seguir é a implementação da interface **Serializable**. A implementação desta interface fornece ao bean a propriedade chamada de **persistência**.

O conceito de persistência é permitir que o usuário faça uso do componente em um determinado momento e possa salvar o seu estado para o uso posterior partindo do mesmo ponto. A tecnologia que possibilita essa propriedade é a **Serialização de Objetos**. Esta tecnologia permite salvarmos o objeto em um fluxo para posterior recuperação. Quando houver a recuperação, o objeto deve se comportar como se fosse exatamente o mesmo de quando foi salvo, se olharmos o objeto como uma máquina de estado, então podemos dizer que o objeto é recuperado no mesmo estado de quando foi salvo.

A tecnologia de serialização de objetos é muito importante em Java pois permite a linguagem atividades complexas como computação distribuída e além de muitas outras funcionalidades interessantes.

Para sermos capazes de serializar objetos basta fazer a classe implementar a interface **Serializable** e nada mais.

Propriedades de um Bean

Os beans possuem diversos tipos de propriedades, que são:

- **Simples:** Propriedades simples alteram a aparência ou comportamento do bean e são acessadas através dos métodos de escrita e leitura mostrados.
- **Ligadas:** As propriedades deste tipo quando alteradas provocam uma notificação para algum evento.
- **Reprimidas:** As propriedades podem ter sua mudança vetada pelo próprio bean ou por objetos externos.
- **Indexadas:** Propriedades indexadas representam coleções de valores acessados por indices, como arrays. A seguinte regra deve ser seguida por propriedades indexadas:

Para as propriedades indexadas temos:

- Métodos de leitura do array inteiro:

```
public TipoDaPropriedade[] getNomeDaPropriedade();
public void setNomeDaPropriedade(TipoDaPropriedade[] value);
```

- Métodos de leitura de elementos individuais:

```
public TipoDaPropriedade getNomeDaPropriedade(int index);
public void setNomeDaPropriedade(int index, TipoDaPropriedade value);
```

Como as ferramentas lêem os Beans ?

JavaBeans são desenvolvidos param serem usados por terceiros (ou não). As ferramentas de desenvolvimento em Java são capazes de ler beans e prover a funcionalidade do componente para o usuário. Mas como as ferramentas são capazes de descobrir a funcionalidade de um Bean já que ele não é uma classe conhecida, ou seja, pode ser qualquer componente?

Para a realização desta tarefa, que chamamos instrospecção, é usada uma API disponível no Ambiente Runtime da plataforma Java, a **Java Reflection API** . Usando a funcionalidade desta API a ferramenta procura os métodos públicos do bean e faz a ligação dos métodos com as propriedades. Após a ferramenta ter obtido todas as informações que ela necessita sobre a classe então as propriedades e os métodos são disponibilizados para uso.

A instrospecção é realizada pela classe **java.beans.Introspector** que usa a **Reflection API** para descobrir as funcionalidades do bean por casamento de padrão. No entanto, alternativamente você pode definir as informações sobre o Bean em separado, em uma classe que implementa a interface **BeanInfo**, porém não entraremos em detalhes sobre isso.

É importante saber que um bean também pode implementar outros métodos que não aqueles que lêem ou gravam as propriedades. Esses métodos não diferem em nada de outros métodos de qualquer outra classe em Java.

JavaBeans x Enterprise JavaBeans

Já conhecemos o que é um JavaBean, então agora podemos compará-los com o Enterprise JavaBeans (EJB). Quais as diferenças?

Os JavaBeans assim como o EJB são modelos de componente, porém apesar dos dois compartilharem o termo **JavaBeans** os modelos não são relacionados. Uma parte da literatura tem referenciado EJB como uma extensão dos JavaBeans, mas isto é uma interpretação errada. As duas APIs servem para propósitos bem diferentes.

O JavaBean tem a intenção de ser usado em propósitos do mesmo processo (intraprocesso) enquanto EJB é projetado para ser usado como componentes interprocessos. Em outras palavras, os JavaBeans não têm a intenção de ser usado como componente distribuído assim como o EJB.

O EJB é uma tecnologia que está em grande ascenção e será abordado em futuros materias do JSPBrasil.