Компоненты JavaBeans в страницах JSP

Stephanie Bodoff

Компоненты JavaBeans являются Java классами, которые могут быть легко повторно использованы и скомпонованы друг с другом в приложение. Любой Java класс, который следует определенным соглашениям, может быть компонентом JavaBeans.

Технология JavaServerPagesнепосредственно поддерживает применение компонентов JavaBeans в элементах языка JSP. Вы можете легко создавать и инициализировать бины, а также получать и устанавливать значения их свойств (полей). Эта глава содержит основную информацию о компонентах JavaBeans и элементах языка JSP для получения компонентов JavaBeans из ваших страниц JSP. Для дополнительной информации о модели компонента JavaBeans смотрите http://java.sun.com/products/javabeans.

1. Соглашения проектирования компонентов JavaBeans

Соглашения проектирования компонентов JavaBeans определяют свойства класса и те публичные методы, которые дают доступ к этим свойствам.

Свойство компонента JavaBeans может быть:

- Доступным на чтение/запись, только чтение или только запись (Read/write, read-only, orwrite-only).
- Простым (simple), если оно содержит единственное значение, или индексированным (indexed), если оно представляет массив значений.
  е существует требования, чтобы свойство было реализовано экземпляром какой-либо переменной; это свойство просто должно быть доступно к использованию публичными методами, которые соответствуют определенным соглашениям:

- Для каждого доступного для чтения свойства бин должен иметь метод следующего вида: `PropertyClass getProperty() { ... }`
- Для каждого доступного для записи свойства бин должен иметь метод вида:
```
setProperty(PropertyClasspc) {
... }
```

В добавок к этим методам компонент JavaBeans должен определить конструктор без параметров.

Страницы JSP приложения `Duke'sBookstore: enter.jsp`, `bookdetails.jsp`, `catalog.jsp`, `showcart.jsp` используют компоненты JavaBeans: database.BookDB и database.BookDetails. `BookDB` предусматривает внешний интерфейс компонента JavaBeans для enterprise bean `BookDBEJB`. Оба бина широко пользуются заказными тегами (смотрите Tags That Define Scripting Variables). Страницы JSP `showcart.jsp` и `cashier.jsp` используют cart.ShoppingCart , чтобы представить пользовательскую карту покупок (корзинку).

Страницы JSP `catalog.jsp`, `showcart.jsp` и `cashier.jsp` используют компонент JavaBeans util.Currency, чтобы задавать формат денег в локально - чувствительном стиле. Этот бин имеет два writable свойства, `locale` и `amount`, и одно readable свойство, `format`. Свойство `format` не является экземпляром какой-либо переменной, но возвращает функцию от свойств `locale` и `amount`.

```
public class Currency {
   private Locale locale;
   private double amount;
   public Currency() {
      locale = null;
      amount = 0.0;
   }
   public void setLocale(Locale l) {
      locale = l;
   }
   public void setAmount(double a) {
      amount = a;
   }
   public String getFormat() {
      NumberFormat nf =
```

```
         NumberFormat.getCurrencyInstance(locale);
      return nf.format(amount);
   }
}
```

2. Почему используются JavaBeans компоненты?

Страницы JSP могут создавать и использовать любые типы Java объектов в декларациях или скриплетах. Следующий скриплет создает карту покупок магазина bookstore и сохраняет ее как атрибут сессии:

```
<%
   ShoppingCart cart = (ShoppingCart)session.
      getAttribute("cart");
// Если пользователь еще не имеет карты покупок, создадим новую
   if (cart == null) {
      cart = new ShoppingCart();
      session.setAttribute("cart", cart);
   }
%>
```

Если объект cart соответствует соглашениям JavaBeans, страница JSP может использовать элементы JSP, чтобы создать и получить доступ к объекту. Например, страницы `bookdetails.jsp`, `catalog.jsp`, и `showcart.jsp` заменяют этот скриплет более кратким элементом JSP `useBean`:

```
<jsp:useBean id="cart" class="cart.ShoppingCart"
   scope="session"/>
```

3. Создание и использование JavaBeans компонентов

Вы декларируете, что ваша страница JSP будет использовать компонент JavaBeans, с помощью любого из следующих форматов:

```
<jsp:useBean id="beanName"
   class="fully_qualified_classname" scope="scope"/>
```

или

```
<jsp:useBean id="beanName"
   class="fully_qualified_classname" scope="scope">
   <jsp:setProperty .../>
</jsp:useBean>
```

Второй формат используется, если вы хотите включить выражения `jsp:setProperty`, описанные в следующем разделе, для инициализации свойств бина.

Элемент `jsp:useBean` декларирует, что эта страница будет использовать бин, который запоминается внутри и доступен из определенного контекста, который может быть приложением, сессией, запросом или страницей. Если такой бин не существует, выражение создает этот бин и сохраняет как атрибут scope объекта (смотрите ScopeObjects). Значение атрибута `id` определяет *имя* бина в области видимости и *идентификатор,* используемый, чтобы ссылаться на бин из других элементов JSP и скриплетов.

Следующий элемент создает экземпляр `Currency`, если он не существует, сохраняет его как атрибут объекта `application`, и делает бин доступным повсюду в приложении с помощью идентификатора `currency`:

```
<jsp:useBean id="currency" class="util.Currency"
   scope="application"/>
```

3.1. Установка свойств JavaBeans компонента

Имеется 2 способа установить свойства JavaBeans компонента в странице JSP:

- С помощью элемента `jsp:setProperty`
- С помощью скриплета: `<%= beanName.setpropName(value) %>`

Синтаксис элемента `jsp:setProperty` зависит от источника значения этого свойства. Таблица 1 перечисляет различные пути установки свойства компонента JavaBeans с помощью элемента `jsp:setProperty`:

Источник значения	Синтаксис элемента
Строковая константа	`<jsp:setProperty name="beanName" property="propName" value="string constant"/>`

Параметр запроса	`<jsp:setProperty name="beanName" property="propName" param="paramName"/>`
Имя параметра запроса подбирает свойство бина	`<jsp:setProperty name="beanName" property="propName"/>` `<jsp:setProperty name="beanName" property="*"/>`
Выражение	`<jsp:setProperty name="beanName" property="propName" value="<%= expression %>"/>`

Таблица 1. Установка свойств компонента JavaBeans.

1. Имя бина beanName должно быть тем же самым, которое определено для атрибута `id` в элементе `useBean`.
2. В компоненте JavaBeans должен быть реализован метод *setPropName*.
3. Имя параметра *paramName* должно быть именем параметра запроса.

Свойство, устанавливаемое константой или параметром запроса, должно иметь тип из списка таблицы 2. Поскольку и константа, и параметр запроса являются строками, web контейнер автоматически конвертирует это значение к типу свойства; конверсия применяется как показано в таблице. Значение, присваиваемое индексированному свойству, должно быть массивом, и эти правила применяются к его элементам.

Тип свойства	Конверсия в строковое значение
`boolean` или `Boolean`	Как указано в `java.lang.Boolean.valueOf(String)`
`byte` или `Byte`	Как указано в `java.lang.Byte.valueOf(String)`
`char`или`Character`	Как указано в `java.lang.Character.valueOf(String)`
`double` или `Double`	Как указано в `java.lang.Double.valueOf(String)`
`int` или `Integer`	Как указано в `java.lang.Integer.valueOf(String)`
`float` или `Float`	Как указано в

	`java.lang.Float.valueOf(String)`
`long` или `Long`	Как указано в `java.lang.Long.valueOf(String)`

Таблица 2. Адекватное присваивание значения.

Вы можете использовать выражение времени выполнения (a runtime expression), чтобы установить значение свойства, имеющего сложный тип Java. Повторный вызов из выражений JSP используется, чтобы вставить значение выражения языка скриптов, конвертированное в строку, в поток, возвращаемый клиенту. Если элемент использован в `setProperty`, выражение просто возвращает его значение; автоматическая конверсия не выполняется. Как следствие, тип, возвращаемый выражением, должен быть равным или приводимым к типу свойства.

Приложение `Duke'sBookstore` демонстрирует, как использовать элемент `setProperty` и скриплет, чтобы установить текущую книгу для database helper bean. Например, bookstore3/bookdetails.jsp использует форму:

```
<jsp:setProperty name="bookDB" property="bookId"/>
```

в то время как bookstore2/bookdetails.jsp использует форму:

```
<% bookDB.setBookId(bookId) %>
```

Следующие фрагменты страницы bookstore3/showcart.jsp иллюстрируют, как инициализировать в бине `currency` свойства `locale` и `amount`, определив значения выражений во время выполнения запроса. Поскольку первая инициализация вкладывается (вставляется) в элемент `useBean`, она выполняется только тогда, когда создается бин.

```
<jsp:useBean id="currency" class="util.Currency"
   scope="application">
   <jsp:setProperty name="currency" property="locale"
      value="<%= request.getLocale() %>"/>
</jsp:useBean>
<jsp:setProperty name="currency" property="amount"
   value="<%=cart.getTotal()%>"/>
```

3.2. Извлечение свойств компонента JavaBeans

Имеется несколько путей извлечения свойств компонента JavaBeans. Два из этих методов конвертируют значение свойства в строку и вставляют это значение явно в текущий объект `out`: элемент `jsp:getProperty` и выражение:

- `<jsp:getProperty name="beanName" property="propName"/>`
- `<%= beanName.getpropName() %>`

Для обоих методов, beanName должно быть тем же самым, как это определено в атрибуте `id` в элементе `useBean`, причем должен быть определен метод `getpropName` в компоненте JavaBeans.

Если вам нужно извлечь значение свойства без его конвертирования и вставки в объект out, вам следует использовать скриплет:

```
<% Object o = beanName.getpropName(); %>
```

Отметим разницу между выражением и скриплетом; выражение имеет знак '=' после открывающего знака '%' и не заканчивается точкой с запятой, как скриплет.

Приложение `Duke'sBookstore` демонстрирует, как использовать обе формы, чтобы извлечь форматированный currency из бина currency и вставить его в страницу. Например, bookstore3/showcart.jsp использует:

```
<jsp:getProperty name="currency" property="format"/>
```

в то время как bookstore2/showcart.jsp использует:

```
<%= currency.getFormat() %>
```

Страница `bookstore2/showcart.jsp` приложения `Duke'sBookstore` использует следующий скриплет, чтобы извлечь количество книг из бина shoppingcart и открыть условную вставку текста в выходной поток:

```
<%
   // печатаем резюме карты покупок
   int num = cart.getNumberOfItems();
   if (num > 0) {
```

```
%>
```

Хотя скриплеты очень полезны для динамической обработки, использование заказных тегов (смотрите Custom Tags in JSPPages) для доступа к свойствам объектов и выполнение контроля потока считается лучшим способом. Например, `bookstore3/showcart.jsp` заменяет скриплет следующим заказным тегом:

```
<bean:define id="num" name="cart" property="numberOfItems" />
<logic:greaterThan name="num" value="0" >
```

Рисунок 1 подводит итог тому, где хранятся различные типы объектов и как получить доступ к этим объектам из JSP. Объекты, созданные тегом `jsp:useBean`, сохраняются как трибуты scope объектов и могут быть доступны с помощью тегов `jsp:[get|set] Property` и в скриплетах и в выражениях. Объекты, созданные в декларациях и скриплетах, сохраняются как переменные класса сервлета JSP страницы и могут быть доступны в скриплетах и выражениях.