Компоненты JavaBeans

Компоненты JavaBeans

JavaBeans API это основа для определения многократно используемых, модульных и встраиваемых компонентов ПО.

Наиболее широко компоненты применяются в элементах графического пользовательского интерфейса, например компоненты пакетов `java.awt` и `javax.swing`.

Хотя всеми компонентами JavaBeans можно манипулировать визуально, не у каждого компонента Java есть свое собственное визуальное представление. Например, класс `javax.sql.RowSet` является компонентом JavaBeans который представляет данные, полученные в результате запроса к базе данных

Компоненты JavaBeans

Компонентная модель JavaBeans состоит из пакетов `java.beans` и `java.beans.beancontext`, а также ряда важных соглашений по именованию объектов и прикладным интерфейсам. Этим соглашениям должны соответствовать компоненты Java и инструментарий для работы с ними.

Основы компонентов Java

Компонентом Java может быть любой объект, который соответствует некоторым основным правилам; не существует класса `Bean`, от которого должны порождаться все компоненты Java.

Многие компоненты Java являются AWT-компонентами, но также полезно создавать «невидимые» компоненты Java, которые не видны на экране. Если у компонента Java нет представления на экране в законченном приложении, то это вовсе не значит, что с ним нельзя будет работать визуально в среде разработки.

Основы компонентов Java

Компонент Java характеризуется свойствами, событиями и методами, которые он экспортирует. Именно этими свойствами, событиями и методами оперирует разработчик приложений, применяя контейнер компонентов.

Свойство – это часть внутреннего состояния компонента Java. Его можно установить программно и/или получить его значение – обычно при помощи стандартной пары методов доступа `get` и `set`.

Основы компонентов Java

С помощью генерации *событий* компонент Java общается с приложением, в которое он встроен, и с другими компонентами Java.

JavaBeans API применяет ту же самую модель событий, которую используют AWT и Swing-компоненты. Эта модель основана на классе `java.util.EventObject` и интерфейсе `java.util.EventListener`.

Основы компонентов Java

Схема работы модели событий:

Компонент Java определяет событие, если он предоставляет методы add и remove для регистрации (добавления) и удаления объектов-слушателей данного события.

Приложение, которое хочет получать уведомление о том, что произошло событие определенного типа, использует эти методы для регистрации объекта-слушателя событий соответствующего типа.

Когда происходит событие, компонент Java уведомляет зарегистрированные приемники путем передачи событийного объекта, который описывает событие, методу, определенному интерфейсом слушателя событий.

Основы компонентов Java

Однонаправленное событие – это редкий вариант события, для которого может быть зарегистрирован единственный объект-приемник. Метод регистрации `add` для однонаправленного события вызывает исключение `TooManyListenersException` в том случае, если предпринимается попытка зарегистрировать более одного приемника.

Методы, экспортируемые компонентом Java, – это все методы с модификатором `public`, определенные компонентом Java, за исключением методов доступа к свойствам и методов, которые регистрируют и удаляют слушателей событий.

Основы компонентов Java

В дополнение к обычным свойствам JavaBeans API поддерживает несколько специализированных подтипов.

Индексированное свойство – это свойство, значением которого является массив, а также методы доступа, которые позволяют обращаться как к отдельным элементам массива, так и ко всему массиву в целом.

Связанное свойство – это свойство, которое посылает событие `PropertyChangeEvent` любым заинтересованным объектам `PropertyChangeListener`, когда значение свойства изменяется.

Основы компонентов Java

Ограниченное свойство – это свойство, любые изменения в котором могут быть заблокированы любым заинтересованным слушателем. Когда меняется значение ограниченного свойства компонента Java, он должен выслать `PropertyChangeEvent` списку заинтересованных объектов `VetoableChangeListener`. Если любой из этих объектов вызывает исключение `PropertyVetoException`, то значение свойства не меняется, а исключение `PropertyVetoException` возвращается методу, который устанавливает свойство.

Основы компонентов Java

Контейнер компонентов использует процесс под названием интроспекция (`introspection`) для того, чтобы определить свойства, события и методы, экспортируемые компонентом Java. Механизм интроспекции реализуется классом `java.beans.Introspector`; он опирается на механизм отражения `java.lang.reflect` и на ряд соглашений по именованию JavaBeans. Например, `Introspector` может определить список свойств, поддерживаемых компонентом Java, путем сканирования класса на предмет наличия методов, которые несут имена и сигнатуры, представляющие их как методы доступа к свойствам `get` и `set`.

Основы компонентов Java

Механизм интроспекции опирается не только на возможности отражения, присутствующие в Java. Любой компонент Java может определять вспомогательный класс `BeanInfo`, который предоставляет дополнительную информацию о компоненте Java и его свойствах, событиях, методах. `Introspector` автоматически пытается найти и загрузить класс `BeanInfo`, принадлежащий компоненту Java.

Основы компонентов Java

Класс `BeanInfo` предоставляет дополнительную информацию о компоненте Java в форме объектов `FeatureDescriptor`, каждый из которых описывает одну функциональную особенность этого компонента. Каждый `FeatureDescriptor` предоставляет имя и краткое описание функциональной особенности, которую он документирует.

Основы компонентов Java

Конкретные функциональные особенности компонентов Java – свойства, события и методы – описываются специфическими подклассами `FeatureDescriptor`, такими как `PropertyDescriptor`, `EventSetDescriptor` и `MethodDescriptor`.

.

Основы компонентов Java

Одна из главных задач контейнеров компонентов – предоставить пользователю возможность настройки компонента Java путем установки значений его свойств. Контейнер определяет редакторы свойств для часто используемых типов свойств, таких как числа, строки, шрифты и цвета. Если компонент Java имеет свойство более сложного типа, то он может определять класс `PropertyEditor`, который позволяет контейнеру компонентов предоставить редактор этого свойства.

Основы компонентов Java

Механизм отдельной настройки каждого свойства, который предоставляется большинством контейнеров компонентов, может оказаться недостаточным для сложного компонента Java. Такой компонент может определить класс `Customizer` для создания графического интерфейса, который позволит более удобно конфигурировать компонент Java. Особенно сложный компонент Java может определять модули настройки (`customizers`), которые служат в качестве «мастеров», предоставляющих пользователю возможность пошаговой настройки.

Основы компонентов Java

Контекст компонента Java – это логический (а часто и визуальный) контейнер для JavaBeans и, возможно, для других вложенных контекстов компонентов Java. На практике большая часть JavaBeans – это AWT- или Swing-компоненты или контейнеры. Контейнеры компонентов предусматривают это и позволяют составным компонентам Java размещаться внутри компонентов-контейнеров Java.

Основы компонентов Java

Контекст компонента Java может обеспечивать ряд сервисов (например, печать, отладку, связь с базой данных) для тех компонентов Java, которые он содержит. Можно написать компоненты Java, которые будут «осознавать» свой контекст и посылать запросы к контексту, чтобы воспользоваться доступными сервисами. Контексты компонентов Java реализуются с помощью `java.beans.beancontext` API.

Основы компонентов Java

JavaBeans Persistence API, который позволяет компоненту Java или дереву таких компонентов сохранять свое состояние в XML-файле, откуда впоследствии можно будет восстановить этот компонент или компоненты. Это выполняется с помощью классов `XMLEncoder` и `XMLDecoder` из пакета `java.beans`. Постоянство JavaBeans напоминает механизм сериализации пакета `java.io`, но он всецело основан на открытом API компонентов Java

Соглашения JavaBeans

Иногда соглашения называют проектными шаблонами; они определяют такие понятия, как имена и сигнатуры методов доступа к свойствам, которые определяются компонентом Java. Суть проектных шаблонов – возможность взаимодействия компонентов Java и контейнеров компонентов, которые с ними работают. Контейнеры компонентов могут использовать интроспекцию для определения списка свойств, событий и методов, поддерживаемых компонентом Java. Для того чтобы все это функционировало, разработчики компонентов Java должны использовать имена методов, которые понятны контейнерам компонентов.

Соглашения JavaBeans

Модель JavaBeans облегчает процесс, устанавливая соглашения по именованию. Одно такое соглашение состоит, например, в том, чтобы методы доступа для получения и установки свойства начинались с `get` и `set`. Не все эти шаблоны обязательны. Если компонент Java имеет методы доступа к свойству, которые не подчиняются соглашениями по именованию, то можно использовать объект `PropertyDescriptor` (представлен в классе `BeanInfo`) для обозначения таких методов.

Соглашения JavaBeans

Хотя класс `BeanInfo` является альтернативой соглашению по именованию, которое основано на методе доступа к свойствам, такой метод должен придерживаться соглашений, задающих количество, тип его параметров и возвращаемое значение.

.

Компоненты Java

Сам компонент Java должен придерживаться следующих соглашений:

Имя класса

На имя класса компонента Java не налагается ограничений.

Родительский класс

Компонент Java может расширять любой другой класс. Компоненты Java часто являются AWT- или Swing-компонентами, но ограничений не существует.

Компоненты Java

Создание экземпляра

Компонент Java должен предоставлять конструктор без параметров или файл, содержащий сериализованный экземпляр. Контейнер компонентов может десериализовать этот экземпляр для использования в качестве прототипа при создании экземпляра компонента Java. Файл, который содержит компонент Java, должен иметь то же название, что и компонент Java, а также расширение `.ser`.

Компоненты Java

Имя компонента Java

Имя компонента Java – это имя класса, который его реализует, или имя файла, который содержит сериализированный экземпляр компонента Java без расширения `.ser` и символов разделения каталогов (/), которые переведены в точки (.).

Свойства

Компонент Java определяет свойство *p* типа *T* в том случае, если у него есть методы доступа, которые соответствуют этим шаблонам (если *T* – boolean, то разрешена особая форма метода get):

Метод, возвращающий значение (getter)

public *T* get*P*()

Свойства

Метод, возвращающий значение boolean (boolean getter)

public boolean is*P*()

Метод, устанавливающий значение (setter)

public void set*P*(*T*)

Исключения

Методы доступа к свойству могут вызывать проверяемые и непроверяемые исключения любого типа.

Индексированные свойства

Индексированное свойство – это свойство типа «массив», которое предоставляет методы доступа, получающие и устанавливающие весь массив, равно как и методы, получающие и устанавливающие отдельные элементы массива. Компонент Java определяет индексированное свойство *p* типа *T*[], если он определяет следующие методы доступа:

Индексированные свойства

Метод, возвращающий массив

public *T*[] get*P*()

Метод, возвращающий элемент

public *T* get*P*(**int**)

Метод, устанавливающий массив

public void set*P*(*T*[])

Индексированные свойства

Метод, устанавливающий элемент

public void set*P*(**int**,*T*)

Исключения

Методы доступа к свойству могут вызывать проверяемые и непроверяемые исключения любого типа. В частности, они должны генерировать исключение `ArrayIndexOutOfBoundsException`, если запрашиваемый индекс выходит за пределы допустимого диапазона.

Связанные свойства

Связанное свойство – это свойство, которое генерирует событие `PropertyChangeEvent`, когда его значение меняется. Ниже приведены соглашения для связанного свойства:

Методы доступа

Методы доступа к связанному свойству отвечают тем же соглашениям, что и методы доступа к обычному свойству.

Связанные свойства

Интроспекция

При помощи одной только интроспекции контейнер компонентов не сможет отличить связанное свойство от несвязанного. Поэтому следует реализовать класс `BeanInfo`, который возвращает объект `PropertyDescriptor` для этого свойства. Метод `isBound()` объекта `PropertyDescriptor` должен возвращать true.

Связанные свойства

Регистрация слушателя

Компонент Java, определяющий одно или несколько связанных свойств, должен определить пару методов для регистрации слушателей, которые получают уведомление при изменении значения какого-либо связанного свойства. Такие методы должны иметь следующие сигнатуры:

```
public void addPropertyChangeListener(PropertyChangeListener)
public void removePropertyChangeListener
(PropertyChangeListener)
```

Связанные свойства

Регистрация слушателя для именованного свойства

Компонент Java может, но не обязан предоставлять дополнительные методы, позволяющие регистрировать слушателей событий, при которых изменяются значения отдельного связанного свойства. Этим методам передается имя свойства. Они имеют следующие сигнатуры:

```
public void addPropertyChangeListener(String,
   PropertyChangeListener)
public void removePropertyChangeListener(String,
   PropertyChangeListener)
```

Связанные свойства

Регистрация слушателя для отдельного свойства

Компонент Java может, но не обязан предоставлять дополнительные методы регистрации приемников событий, которые специфичны для конкретного свойства.

Для свойства *p* эти методы имеют следующие сигнатуры:

```
public void addPListener(PropertyChangeListener)
public void removePListener(PropertyChangeListener)
```

Методы такого типа позволяют контейнерам компонентов отличать связанное свойство от несвязанного.

Связанные свойства

Уведомление

Когда изменяется значение связанного свойства, компонент Java должен обновить свое внутреннее состояние, чтобы отразить это изменение, а затем передать `PropertyChangeEvent` методу `propertyChange()` каждого объекта `PropertyChangeListener`, зарегистрированного для такого компонента Java или же для конкретного связанного свойства.

Поддержка

Класс `java.beans.PropertyChangeSupport` полезен при реализации связанных свойств.

Ограниченные свойства

Ограниченное свойство – это свойство, любые изменения в котором могут быть заблокированы зарегистрированными слушателями. Большая часть ограниченных свойств также является связанными свойствами. Ниже приведены соглашения для ограниченного свойства:

Метод, возвращающий значение

Возвращающий метод для ограниченного свойства – это то же самое, что и возвра щающий метод для обычного свойства.

Ограниченные свойства

Метод, устанавливающий значение

Устанавливающий метод ограниченного свойства генерирует `PropertyVetoException`, если блокируется изменение свойства. Для свойства *p* типа *T* есть следующая сигнатура:

```
public void setP(T) throws PropertyVetoException
```

Ограниченные свойства

Регистрация слушателя

Компонент Java, который определяет одно или несколько ограниченных свойств, должен определить два метода для регистрации слушателей, которые получают уведомление, когда меняется значение свойства. Эти методы должны иметь следующие сигнатуры:

```
public void addVetoableChangeListener
(VetoableChangeListener)
public void removeVetoableChangeListener
(VetoableChangeListener)
```

Ограниченные свойства

Регистрация слушателя для именованного свойства

Компонент Java может, но не обязан предоставлять дополнительные методы, позволяющие регистрировать слушателей событий, при которых изменяется значение ограниченного свойства. Этим методам передается имя свойства. Они имеют следующие сигнатуры:

```
public void addVetoableChangeListener (String,
VetoableChangeListener)
public void removeVetoableChangeListener (String,
VetoableChangeListener)
```

Ограниченные свойства

Регистрация слушателя для отдельного свойства

Компонент Java может, но не обязан предоставлять дополнительные методы регистрации слушателей; такие методы специфичны для конкретного ограниченного свойства. Для свойства *p* эти методы имеют следующие сигнатуры:

```
public void addPListener(VetoableChangeListener)
public void removePListener(VetoableChangeListener)
```

Ограниченные свойства

Уведомление

Когда вызывается устанавливающий метод ограниченного свойства, компонент должен сгенерировать событие PropertyChangeEvent, которое описывает запрошенное изменение, и передать это событие методу vetoableChange() каждого объекта VetoableChangeListener, зарегистрированного для компонента или конкретного ограниченного свойства. Если какой-либо слушатель заблокирует это изменение, вызвав исключение PropertyVetoException, то компонент должен будет сгенерировать еще одно событие PropertyChangeEvent для того, чтобы вернуть свойство к его первоначальному значению. Затем компонент должен сам вызвать исключение PropertyVetoException. Если изменение свойства не будет заблокировано, то компонент должен обновить свое внутреннее состояние, чтобы оно отражало проведенное изменение. Если ограниченное свойство также является связанным свойством, то компонент должен в этот момент уведомить объекты PropertyChangeListener.

Ограниченные свойства

Поддержка

Класс java.beans.VetoableChangeSupport полезен при реализации ограниченных свойств.

.

События

В дополнение к событиям `PropertyChangeEvent`, которые генерируются в момент изменения связанных и ограниченных свойств, компонент Java может генерировать и другие типы событий. Событие, названое *E*, должно соответствовать следующим соглашениям:

Класс события

Класс события должен напрямую или косвенно расширять `java.util.EventObject`. Кроме того, он должен называться *E*`Event`.

События

Интерфейс слушателя

Событие должно быть ассоциировано с интерфейсом слушателя событий, который расширяет `java.util.EventListener` и называется *E*`Listener`.

Методы слушателя

Интерфейс слушателя событий может определять любое количество методов, которые принимают единственный аргумент типа *E*`Event` и возвращают `void`.

События

Регистрация слушателя

Компонент Java должен определять пару методов для регистрации слушателей событий, которые хотят получать уведомления в случае, если происходит событие *E*. Такие методы должны иметь следующие сигнатуры:

```
public void addEListener(EListener)
public void removeEListener(EListener)
```

События

Однонаправленные события

Однонаправленное событие позволяет регистрировать только одного слушателя в некий момент времени. Если *E* является однонаправленным событием, то у метода регистрации слушателя должна быть следующая сигнатура:

```
public void addEListener(EListener) throws
   TooManyListenersException
```

Методы

Контейнер компонентов может предоставлять методы компонента Java разработчикам приложений. Единственное формальное соглашение состоит в том, что эти методы должны быть объявлены public. Тем не менее полезно ознакомиться со следующими положениями:

.

Методы

Имя метода

Метод может иметь любое имя, которое не противоречит соглашениям по именованию свойств и событий. Имя должно быть как можно более значимым.

Параметры

Метод может иметь любое количество параметров любого типа. Тем не менее контейнеры компонентов лучше всего работают с методами, не имеющими параметров, или с методами, которые имеют простые параметры примитивных типов.

Методы

Исключение методов

С помощью реализации BeanInfo компонент Java может явно указать список методов, которые он экспортирует.

Документация

Компонент Java может предоставлять дружественные и простые в восприятии локализованные имена и описания методов посредством объектов MethodDescriptor, которые возвращаются реализацией BeanInfo.