Компоненты JavaBeans

(продолжение)

Вспомогательные классы

Компонент Java может предоставлять следующие вспомогательные классы:

`BeanInfo`

В целях предоставления дополнительной информации о компоненте *B* нужно реализовать интерфейс `BeanInfo` в классе с названием *`B`*`BeanInfo`.

Вспомогательные классы

Редактор свойств для конкретного типа

Чтобы предоставить контейнерам компонентов возможность работать со свойствами типа *T*, реализуйте интерфейс `PropertyEditor` в классе с названием *`T`*`Editor`.

Класс должен иметь конструктор, не содержащий параметров.

Редактор свойств для конкретного свойства

Для настройки способа, с помощью которого контейнер компонентов позволяет вводить значения отдельно взятых свойств, определите класс, который реализует интерфейс `PropertyEditor` и имеет конструктор без параметров; зарегистрируйте этот класс, передав ему объект `PropertyDescriptor`, который возвращается классом `BeanInfo` для компонента Java.

Вспомогательные классы

Настройщики

Чтобы определить мастеры настройки для конфигурирования компонента *B*, задайте AWT или Swing-компонент с конструктором без параметров, который выполняет настройку. Этот класс обычно называют *B*`Customizer`, но это не обязательно. Зарегистрируйте класс с помощью объекта `BeanDescriptor`, который возвращается классом `BeanInfo` для этого компонента Java.

.

Вспомогательные классы

Документация

Напишите документацию по умолчанию для компонента *B* в формате HTML 2.0 и сохраните ее в файле с именем `B.html`. Подготовьте локализованные переводы документации в файлах с тем же именем, но в каталогах, соответствующих конкретным регионам (`locale`).

.

Объединение компонентов в пакеты и их распространение

Компоненты Java распространяются в архивных файлах JAR, которые подчиняются следующим соглашениям:

Содержание

Класс или классы, которые реализируют компонент Java, должны быть включены в JAR-файл вместе со вспомогательными классами, такими как реализации `BeanInfo` и `PropertyEditor`. Если для компонента Java создается экземпляр из сериализованного ранее экземпляра, то такой экземпляр должен быть включен в архив JAR. Имя этого файла должно заканчиваться на *`.ser`*. JAR-файл может содержать HTML-документацию для компонента Java и любые файлы ресурсов, такие как файлы изображений, необходимые компоненту Java и его вспомогательным классам. Один JAR-файл может содержать несколько компонентов Java.

Объединение компонентов в пакеты и их распространение

Атрибут JavaBean

В манифесте (`manifest`) JAR-файла все файлы *`.class`* и *`.ser`*, которые определяют компонент Java, должны быть помечены следующим атрибутом:

```
JavaBean: true
```

Ю

Объединение компонентов в пакеты и их распространение

Атрибут `DependsOn`

В манифесте JAR-файла можно применять атрибут `DependsOn` для того, чтобы указать все остальные файлы в JAR-архиве, необходимые компоненту Java. Контейнер компонентов может использовать эту информацию во время генерации приложений или при повторном объединении компонентов Java. Каждый компонент Java может иметь несколько атрибутов `DependsOn`, каждый из которых может включать в себя несколько имен файлов, разделенных пробелами. В JAR-файле символ «/» всегда используется как разделитель каталогов.

Объединение компонентов в пакеты и их распространение

Атрибут DesignTimeOnly

В манифесте JAR-файла можно использовать атрибут DesignTimeOnly для указания вспомогательных файлов, таких как реализации BeanInfo. Эти файлы задействуются инструментом контейнера при редактировании, но не используются приложениями, применяющими компонент Java. Контейнер при редактировании может использовать эту информацию при повторном объединении компонентов Java в пакеты.

Пакет java.beans

Пакет `java.beans` содержит классы и интерфейсы для поддержки компонентов `JavaBeans`. Большинство из них используются программами управления компонентами (`bean`), а не самими компонентами. Эти классы и интерфейсы также используются или реализуются вспомогательными классами, с помощью которых разработчики компонентов предоставляют дополнительную информацию для инструментов управления компонентами.

Пакет java.beans

Класс `Beans` определяет несколько широко используемых статических методов. Особенно важен метод `instantiate()`. Класс `Introspector` используется для получения информации о компоненте и его экспортируемых свойствах, событиях и методах. Для этого он использует класс `FeatureDescriptor` вместе с его классами-потомками. Пакет `java.beans` также определяет класс `PropertyChangeEvent` и интерфейс `PropertyChangeListener`, которые часто используются в библиотеках `AWT` и `Swing` для рассылки сообщений при изменении связанного свойства компонента GUI.

Пакет java.beans.beancontext

Пакет `java.beans.beancontext` расширяет модель компонентов `JavaBeans`, добавляя в нее понятие иерархии вложений (`containment`). Он также поддерживает контейнеры компонентов, которые предоставляют содержащимся в них компонентам контекст выполнения и могут предоставлять набор сервисов. Как правило, этот пакет используется опытными разработчиками компонентов и программ для управления компонентами. Разработчикам приложений, которые лишь используют компоненты, этот пакет не требуется.

Пакет java.beans.beancontext

`BeanContext` – главный интерфейс пакета. Он является контейнером для компонентов и определяет несколько методов, которые возвращают информацию о контексте для компонентов. `BeanContextServices` расширяет интерфейс `BeanContext`: в нем определены методы, позволяющие компонентам из контейнера запрашивать доступные сервисы. В компоненте, который нужно уведомлять о его принадлежности контейнеру, должна быть реализация интерфейса `BeanContextChild`. Сам контекстный объект (`BeanContext`) является дочерним контекстом (`BeanContextChild`), то есть контексты могут быть вложены в другие контексты.