Конспект лекций по Java. Занятие 19

В.Фесюнов

1. Класс JList библиотеки Swing (продолжение)

На предыдущем занятии мы не рассмотрели механизм программирования действий по выбору элемента списка. Сейчас мы рассмотрим этот механизм с точки зрения практического использования.

При выборе элемента списка генерируется событие **javax.swing.event.ListSelectionEvent** . Соответственно, для программирования реакции на это событие нужно создать и зарегистрировать слушателя этого события.

Рассмотрим те методы и классы, которые необходимы для реализации этого процесса.

В классе `JList` имеется метод

```
public void addListSelectionListener(ListSelectionListener listener)
```

Он добавляет слушателя типа *ListSelectionListener* .

Интерфейс `ListSelectionListener` определен в пакета `javax.swing.event` . Посмотрим документацию. Этот интерфейс имеет единственный метод

```
public void valueChanged(ListSelectionEvent e)
```

Таким образом, каждый раз при выборе очередного элемента списка `JList` вызывает метод `valueChanged` для всех зарегистрированных слушателей.

В качестве параметра методу `valueChanged` передается объект класса `ListSelectionEvent` . Внутри метода `valueChanged` можно использовать методы этого класса для определения того, какой элемент списка был выбран. Но гораздо удобнее воспользоваться методами самого класса `JList` , которые позволят определить номер или номера выбранных элементов. Рассмотрим эти методы.

Метод

```
public int getSelectedIndex()
```

позволяет определить номер выбранного элемента списка. Если выбрано несколько элементов, он возвращает номер первого из них. Если не выбрано ни одного элемента, он возвращает -1.

Метод

```
public Object getSelectedValue()
```

позволяет получить первый из выбранных элементов списка. Если не выбрано ни одного элемента, он возвращает null.

Как мы видим, класс `JList` позволяет выбрать не один, а несколько элементов списка. В диалоге это можно сделать так. Удерживая клавишу *Shift* , кликнуть на первом элементе, который нужно выбрать, потом на последнем. В результате будет выбран непрерывный блок элементов списка. Можно выбрать и несвязанную группу элементов, используя вместо *Shift* клавишу *Ctrl* . Соответствующие методы для работы с группой выбранных элементов — `getSelectedIndices` и `getSelectedValues` (см. документацию).

Возможность выбора групп элементов может быть включена/выключена методом `setSelectionMode` (см. документацию).

1.1. Практическая работа

Применим рассмотренные возможности для того, чтобы завершить разработку примера с записной книжкой. Нам нужно, чтобы при выборе некоторого элемента списка в полях "Имя" и "Телефон" в правой панели появились данные из выбранного объекта *Person* .

Теперь, зная все необходимые для этого средства, сделать это очень легко. Нужно вставить в конструктор основного класса фрагмент, который создает и регистрирует слушателя. А в слушателе — запрограммировать заполнение полей "Имя" и "Телефон" из текущего выбранного объекта *Person* .

2. Аплеты (applets)

Познакомимся еще с одним важным понятием — с **аплетами** . Аплеты — это маленькие программы, которые работают внутри *браузера* (*browser*).

Здесь прилагательное "маленькие" отражает типичную практику использования аплетов, а не какое-то формальное требование. Теоретически аплеты могут быть сколь угодно большими и сложными. Но объем аплета влияет на время его запуска, поскольку обычно код аплета передается по сети Internet/Intranet. Соответственно, большой аплет потребует много времени на загрузку и его использование не будет оправданным.

Наиболее популярными браузерами являются *Netscape Navigator* и *Microsoft Internet Explorer* (IE). Текущие версии обоих браузеров поддерживают работу аплетов. Для этого в каждый из них встроена VM (виртуальная машина Java), позволяющая выполнять аплет.

Поскольку существует своя специфика выполнения программ на Java под браузером, то существуют свои правила построения этих программ, т.е. правила построения аплетов. Мы рассмотрим эти правила чуть позже, а пока рассмотрим некоторые общие проблемы аплетов.

2.1. Проблема безопасности

Суть проблемы в том, что аплет — это, как правило, программа, полученная из внешнего источника. Соответственно, она опасна. По злому умыслу или из-за ошибки она может нанести вред. Более того, аплет как бы составляет одно целое с web-страничкой и для пользователя может быть даже не совсем очевидно, что безобидная web-страница скрывает за собой потенциально опасный программный код.

В целях безопасности аплеты, по умолчанию, сильно ограничены в своих правах. Эти ограничения определяются настройками конкретного браузера. Они могут быть ослаблены самим пользователем как в отношении всех аплетов, так и для конкретных аплетов.

Ограничений достаточно много, но мы рассмотрим только основные из них.

Аплеты не могут ни читать, ни писать на локальный диск. Они могут передавать информацию только в тот адрес, откуда загружен аплет. Окно аплета специально выделяется или делается какая-то другая пометка для того, чтобы пользователь представлял себе, что он имеет дело с аплетом, а не с обычной web-страницей.

2.2. Проблема совместимости версий

Кроме того, с аплетами связаны и другие проблемы.

Каждая новая версия Java вводит какие-то усовершенствования в аплеты, иногда очень существенные. После публикации этих усовершенствований фирмы, разработчики браузеров, должны ввести соответствующие изменения в свои браузеры. На это требуется время.

Потом новые версии браузеров должны быть установлены пользователями. На что тоже требуется определенное время.

Поэтому, при разработке аплетов приходится учитывать факт наличия устаревших браузеров, не поддерживающих те или иные нововведения.

Так, в частности, относительно недавно (с точки зрения времени, необходимого на установку современных версий браузеров) появилась возможность упаковки Java-программ, в том числе и аплетов, в jar-файлы, что резко снижает время загрузки аплета и сильно влияет на технологию разработки практически эффективных аплетов.

Ранее аплет старались выполнить в виде одного или небольшого числа классов. Это связано с тем, что каждый класс транслируется в отдельный файл, а загрузка каждого файла по Internet требует одного соединения (connection). В последних версиях это несущественно — на скорость загрузки влияет не количество классов, а размер jar-файла.

Кроме того, постоянно изменяется, совершенствуется структура jar-файлов. В их состав включаются опциональные компоненты с определенной функциональностью. Так, в последних версиях в состав jar-файла может входить цифровая подпись, гарантирующая, что аплет разработан конкретным разработчиком и не был подменен каким-то злоумышленником.

2.3. Создание аплетов

Техника написания аплетов базируется на классе `JApplet` пакета `javax.swing` .

Этот класс имеет много своих методов и ряд методов, унаследованных от класса `Applet` . Однако, их изучение по документации мало что даст для овладения техникой построения аплетов. Просто нужно знать, как построить аплет при помощи этих методов.

Для построения аплета нужно создать класс — наследник класса `JApplet` и переопределить в нем ряд методов класса `Applet` . В классе `JApplet` эти методы реализованы как пустые заглушки, которые ничего не делают. При работе аплета внутри браузера он вызывает эти методы в определенных ситуациях. Если мы определим свои методы, то браузер вызовет их, а не методы класса `Applet` .

Рассмотрим эти методы.

```
public void init()
```

Вызывается браузером сразу после загрузки аплета перед первым вызовом метода start(). Этот метод нужно переопределять практически всегда, если в аплете требуется хоть какая-то инициализация.

```
public void start()
```

Вызывается браузером при каждом "посещении" данной страницы. Имеется в виду, что можно загрузить данную страницу, потом загрузить другую, не убирая данную, а потом вернуться к данной. Используется обычно в комбинации с методом `stop` для экономии ресурсов в том случае, например, если аплет выполняет некоторую анимацию. Тогда `stop` может ее остановить, а `start` запустить снова.

```
public void stop()
```

Вызывается браузером при деактивизации данной страницы как в случае загрузки новой страницы без выгрузки данной, так и в случае выгрузки данной. В последнем случае `stop` вызывается перед `destroy` .

```
  public void destroy()
```

Вызывается браузером перед выгрузкой данной страницы.

Обычно при создании аплета переопределяют метод init() и реализуют в нем формирование экрана. При этом вся функциональность аплета обеспечивается слушателями (listeners) полей, кнопок и других активных визуальных компонент. Рассмотрим пример.

2.3.1. Пример аплета

В данном чисто демонстрационном примере присутствует метка, с надписью "Первый аплет", кнопка и текстовое поле, в которое при нажатии на кнопку выводится текст "Привет".

Файл *AppDemo.java* :

```
// AppDemo.java

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AppDemo extends JApplet {

    JTextField txt = new JTextField(15);

    public AppDemo() {}

    public void init() {
      Container c = getContentPane();
      c.setLayout(new FlowLayout());
      JLabel lbl = new JLabel("Первый аплет");
      c.add(lbl);
      JButton btn = new JButton("Нажать один раз");
      btn.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            txt.setText("Привет");
          }
      });
```

```
      c.add(btn);
      c.add(txt);
    }

}
```

Для запуска этого аплета кликните на следующей ссылке: Пример аплета .

Отметим одну деталь. Как и для JFrame для JApplet следует добавлять компоненты в панель getContentPane().

Т.е. аплет выглядит очень похоже на обычную диалоговую программу на Java, только в отличие от рассмотренных нами примеров диалоговых программ, где формирование экрана выполнялось в конструкторе, здесь экран формируется в методе init().

2.4. Запуск аплетов

Аплет запускается браузером при просмотре web-страницы, в которой имеется ссылка на этот аплет. Ранее для запуска аплета было достаточно написать совсем простой код в составе HTML-странички. Например, для запуска вышеприведенного *AppDemo* можно было создать такой файл (*AppDemo.html*):

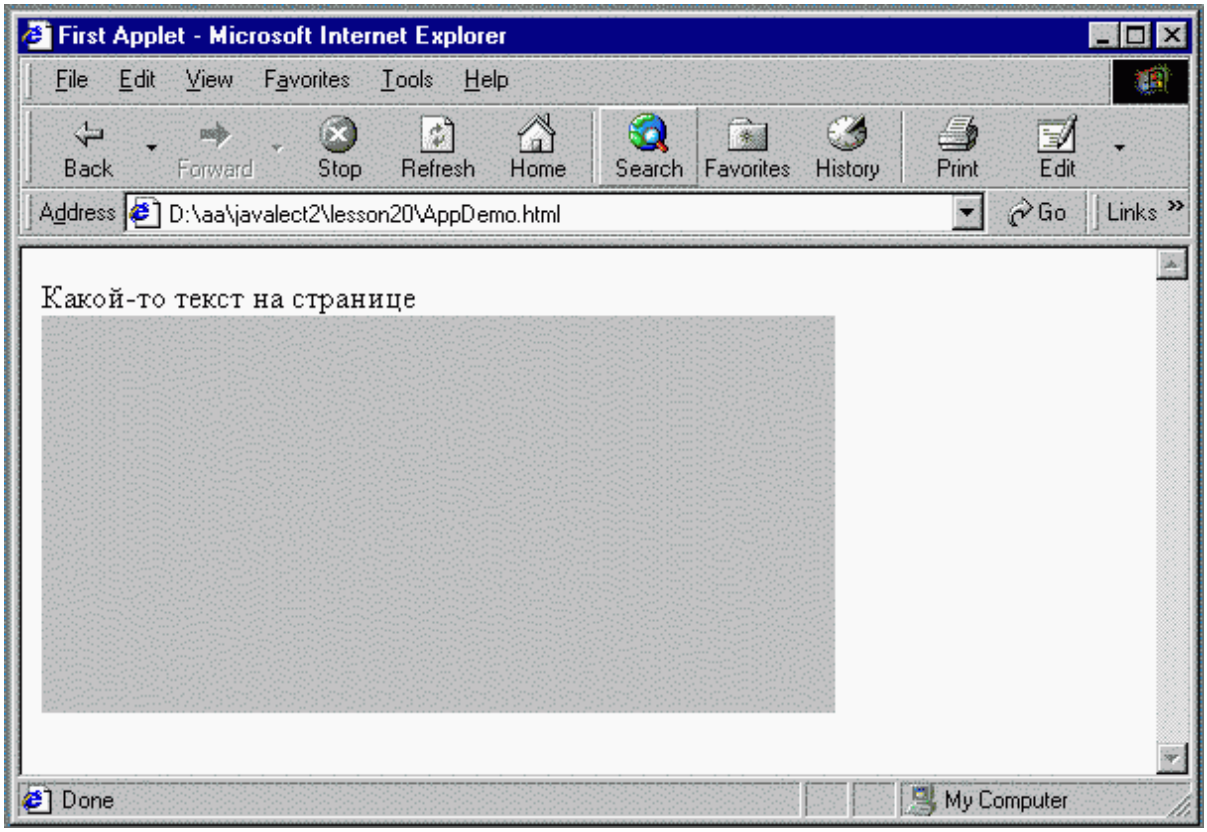
```
<html>
<head><title>First Applet</title></head>
<body>
Какой-то текст на странице
<applet code=AppDemo width=400 height=200>
</applet>
</body>
</html>
```

Для тех, кто знаком с основами HTML, понять этот код не составит труда. Здесь для подключения аплета использован таг `<applet>` . Этот таг имеет параметры `code`, `width` и `height` . Ссылка на аплет реализована при помощи параметра `code` . Если аплет, как рекомендовано, заархивирован в jar-файл, для ссылки на архив нужно использовать параметр `archive` тага `applet` . Например, для файла MyApplet.jar:

```
<applet code="." width=400 height=200 archive=MyApplet.jar>
```

```
</applet>
```

Однако, если попробовать воспользоваться вышеприведенным кодом *AppDemo.html* , то результат будет примерно таким, как показано на рис.



Дело в том, что сейчас в силу ряда причин коммерческого характера таг `applet` не поддерживается браузерами и все стало несколько сложнее. Фирме Sun для независимой поддержки возможностей работы аплетов пришлось создать подключаемые (plug-in) элементы для браузеров Internet Explorer и Netscape Navigator. Соответственно, в html-коде должны присутствовать ссылки специального вида на эти элементы.

Например, эквивалент приведенного выше текста сейчас должен выглядеть так.

```
<html>
<head><title>First Applet</title></head>
```

```
<body>
Какой-то текст на странице

<!--"CONVERTED_APPLET"-->
<!-- CONVERTER VERSION 1.3 -->
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = 400 HEIGHT = 200
codebase="http://java.sun.com/products/plugin/1.3/jinstall-13-win32.cab#Version=1,3,0,0
<PARAM NAME = CODE VALUE = AppDemo >

<PARAM NAME="type"
VALUE="application/x-java-applet;version=1.3">
<PARAM NAME="scriptable" VALUE="false">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.3"
CODE = AppDemo WIDTH = 400 HEIGHT = 200 scriptable=false
pluginspage="http://java.sun.com/products/plugin/1.3/plugin-install.html"><NOEMBED></CO

</NOEMBED></EMBED>
</OBJECT>

<!--
<APPLET CODE = AppDemo WIDTH = 400 HEIGHT = 200>


</APPLET>
-->
<!--"END_CONVERTED_APPLET"-->

</body>
</html>
```

Разобраться в этом коде уже несколько сложнее. К счастью, этого можно не делать. Фирмой Sun разработан html-конвертер для автоматического преобразования html-страниц из старой версии в новую. На момент написания данного текста последнюю версию этого конвертера можно было загрузить по адресу http://java.sun.com/products/plugin/1.3/converter.html . В частности, приведенный выше текст получен именно путем конвертации при помощи html-конвертера.

- Аплет можно просмотреть при помощи вспомогательной программы

appletviewer.exe , входящей в состав JDK. Так, приведенный выше аплет можно запустить командой

```
appletviewer.exe AppDemo.html
```

Если аплет создан и нормально работает на Вашей машине, то это еще ничего не значит. Как и для всех сетевых приложений, для аплетов нужно проводить дополнительное тестирование в сети Internet. Для этого нужно иметь свой сайт, поместить аплет на сайт и вызвать его через Internet с нескольких различных машин, желательно, использующих различные браузеры.

Несколько простых рекомендаций

- В составе web-страницы ссылка на аплет может быть сделана в виде URL, если аплет размещен не там же, где и сама web-страница. Но лучше не мудрствуя лукаво размещать аплет там же, где и страницу.
- По той же причине не стоит помещать аплет в какой-либо пакет.
- Аплет рекомендуется паковать в jar-файл. При этом на web-странице в таг <applet...> нужно вставить ссылку на этот jar-файл через параметр `archive` , как указывалось выше.

Замечание:

Иногда при написании аплетов вместо `init` переопределяют метод `paint` . Это метод класса `Container` , являющегося базовым для класса `Applet` . Такая реализация возможна, но не является рекомендованной.

2.5. Аплеты и приложения

Практически любое диалоговое приложение на Java может быть реализовано так, что оно сможет работать и как программа и как аплет (не нужно только забывать об ограничениях, налагаемых на аплеты по доступу к ресурсам машины).

Изменим приведенный выше пример так, чтобы он мог работать и как обычное приложение.

```
// AppDemo.java

import java.awt.*;
import java.awt.event.*;
```

```
import javax.swing.*;

public class AppDemo extends JApplet {

    JTextField txt = new JTextField(15);

    public AppDemo() {}

    public void init() {
      Container c = getContentPane();
      c.setLayout(new FlowLayout());
      JLabel lbl = new JLabel("Первый аплет");
      c.add(lbl);
      JButton btn = new JButton("Нажать один раз");
      btn.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            txt.setText("Привет");
          }
      });
      c.add(btn);
      c.add(txt);
    }

  public static void main(String[] args) {
    JApplet applet = new AppDemo();
    JFrame frame = new JFrame("Applet Example");
    frame.addWindowListener(
      new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
          System.exit(0);
        }
    });
    frame.getContentPane().add(applet);
    frame.setSize(400,100);
    applet.init();
    applet.start();
    frame.setVisible(true);
  }

}
```

Для этого понадобилось реализовать метод `main` . В нем мы создаем объект нашего класса `AppDemo` , объект класса `JFrame` , помещаем объект класса `AppDemo` на фрейм, устанавливаем размер фрейма. После этого мы вызываем методы `init` и `start` , выполняя те действия, которые сделал бы браузер, если бы мы запускали аплет с его помощью. И, наконец, активизируем фрейм методом `setVisible` .

3. Практическая работа

Реализовать пример с записной книжкой так, чтобы он мог работать и как приложение, и как аплет.