ЛАБОРАТОРНАЯ РАБОТА № 12
«Использование апплетов в документах HTML»

Цель: *Обучить использованию апплетов в документах HTML.*

Ход работы

1. Разобрать приведенные примеры все приведенные примеры.
2. Законспектируйте новый материал
4. Вывод
5. Ответы на контрольные вопросы

Апплет (applet) ~ небольшая Java-программа, временно загружаемая браузером на компьютер пользователя и выполняемая при открытии данной Web-страницы.

Обратимся к небольшому примеру. Пусть пользователь загружает с помощью обозревателя Internet следующий файл HTML.

```
<HTML>

<HEAD>

</HEAD>

<BODY>

<H1>Апплет</H1><BR>

<APPLET CODE=MyApplet. class WIDTH=350 HEIGHT=30>

</APPLET>

</BODY>

</HTML>
```

В данном случае обозреватель Internet отобразит строку текста Апплет, а затем выделит область экрана высотой 30 пикселей и шириной 350 пикселей для отображения результата запуска встроенного апплета MyApplet.class. После этого обозреватель загрузит файл апплета MyApplet.class, который, предположительно, находится на сервере в том же каталоге, что и данная страница HTML, если только не будет указан другой каталог (просмотрите информацию, касающуюся атрибута CODEBASE дескриптора APPLET). После загрузки файла класса начинается непосредственное выполнение апплета. Вся информация, появляющаяся в результате выполнения аплета,

будет отображаться в той области экрана, которая была специально выделена для этого обозревателем еще до загрузки самого апплета.

Исходный код каждого апплета должен начинаться с импорта двух пакетов, необходимых для его разработки: java.awt и Java.applet.

Любой апплет представляет собой расширение предопределенного класса Applet, который поставляется вместе с JDK. Он может реализовывать несколько методов — init( ), start ( ), stop ( ), paint ( ) и update ( ). Все эти методы могут быть переопределены, так что стандартная предложенная их реализация может быть заменена собственной.

При первом запуске апплета сразу же вызывается метод init( ), стандартная реализация которого, как было упомянуто выше, может быть заменена собственной. Данный метод используется для установки начальных параметров, необходимых для работы апплета. Одной из важных особенностей метода init ( ) является то, что этот метод запускается только лишь один раз за все время работы апплета.

Каждый раз, когда апплет становится видимым в окне обозревателя, автоматически выполняется метод start ( ), и каждый раз, когда апплет становится невидимым, выполняется метод stop ( ). Для того чтобы понять, что же такое видимость/невидимость апплета, лучше всего рассмотреть очень простой пример, который четко характеризует эти понятия. Метод stop( ) выполняется каждый раз, когда пользователь минимизирует окно обозревателя Internet. Когда же он снова возвращает окну его первоначальный размер, выполняется метод start ( ) (но не метод init ( )). Аналогичным примером может быть ситуация, когда пользователь щелкает на кнопках Forward (Вперед) и Back (Назад) обозревателя Internet,

Метод paint ( ) — это один из методов, который вызывается для того, чтобы отобразить в области обозревателя Internet, выделенной для выполнения апплета, какой-нибудь текст или изображение. Как правило, данный метод практически всегда переопределяется, так как апплеты используются в основном для вывода на Web-страницу какого-либо текста или графики.

Рассмотрим практический пример использования апплета. Все, что происходит в результате выполнения апплета FirstApplet — это отображение в области, выделенной обозревателем для данного апплета, некоторой строки текста. Сама область при этом окрашивается в красный цвет, а текст ото-бражается белым. Для того чтобы посмотреть на результат встраивания апплета в Web-страницу, необходимо разместить на Web-сервере файлы firstApplet.html и FirstApplet.class. Затем, открыв в обозревателе Internet файл firstApplet.html, можно увидеть результат выполнения апплета FirstApplet.

Файл FirstApplet.Java имеет следующий вид.

```
import java. awt.*;

import java. applet.*;
```

```
public class FirstApplet extends Applet
{
String message = "";
public void init ( )
{
message = getPararneter ("message") ;
}
public void paint (Graphics g)
{
g. setColor(new Color (255, 0, 0));
g. fillRect(0,0,200,30);
g. setColor(new Color(255,. 255, 255));
g. drawString(message, 10, 10); } .
}
```

Файл firstApplet. html имеет такой вид.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<APPLET CODE=FirstApplet. class WIDTH=350 HEIGHT=30>
<PARAM NAME="message" VALUE="Сообщение из файла HTML">
</APPLET> </BODY> </HTML>
```

Дескриптор <APPLET> позволяет не только указывать размеры области окна обозревателя, выделяемой для выполнения апплета, и имени файла класса самого апплета, но и обеспечивает возможность передачи в апплет параметров. В приведенном ниже примере апплету передается параметр message:

```
<PARAM NAME="message" VALUE="Сообщение из файла HTML">
```

В методе init ( ) апплета FirstApplet выполняется чтение переданного ему параметра message.

message = getParameter("message");

get Parameter ( ) — это метод класса Applet, который выполняет чтение параметра, переданного апплету из документа HTML.

Помимо переопределения метода init( ), апплет FirstApplet переопределяет также и метод paint ( ). Все, что с помощью этого метода будет выведено на экран компьютера, отобразится в области обозревателя Internet, выделенной для данного апплета.

public void paint(Graphics g)

Этот метод принимает в качестве параметра идентификатор объекта Graphics. Данный объект содержит такие методы, как drawstring ( ) и drawlmage ( ), которые позволяют соответственно выводить на экран компьютера текст и изображения. Кроме этого, объект Graphics имеет методы, предназначенные для установки фона, на котором будет отображаться текст.

Также объект Graphics может быть использован для отрисовки простейших геометрических примитивов, таких как линии и прямоугольники. В апплете FirstApplet осуществляется прорисовка прямоугольника, заполненного красным цветом. Сначала вызывается метод, задающий цвет, а затем — метод fillRect ( ), рисующий *сам прямоугольник.*

g. setColor(new Color(255, 255, 255));

g. drawstring(message, 10, 10);

Цвет задается с помощью стандартной модели RGB, где каждый компонент цвета представляется числом от 0 до 255. Метод fillRect ( ) принимает четыре аргумента. Первые два из них представляют собой соответственно координаты x и y левого верхнего угла прямоугольника, заданные относительно области окна, в которой выполняется апплет. Следующие два параметра определяют длину и высоту прямоугольника.

После этого снова вызывается метод, устанавливающий цвет. На этот раз выполняется выбор цвета текста, который будет отображен на фоне прямоугольника. Довольно естественно на фоне красного прямоугольника смотрится текст белого цвета. После выбора цвета выводится строка текста, которая отстоит от левого верхнего угла области апплета на 10 пикселей по оси *x и на 15 пикселей по оси y.*

g. setColor(new Color(255, 255, 255));

g. drawstring(message, 10, 10);

Потоки в апплетах

Апплет, в первом примере, выглядит несколько статично. Добавим в него анимацию. Например, попробуем создать эффект бегущей строки в окне апплета вместо того, чтобы просто отображать в нем неподвижный текст. Рассмотрим способ создания бегущей строки, базирующийся на апплете, который создает поток. Этот поток можно запускать тогда, когда апплет становится видимым, и останавливать, когда он становится невидимым. Поток создает эффект бегущей строки путем повторения вызовов метода paint ( ), при этом строка текста каждый раз отображается в новой позиции по оси x.

Также в данном примере будет затронута еще одна концепция языка Java — события и их обработка. Рассматриваемый апплет реагирует на щелчки мышью. Если щелкнуть мышью на области, в которой выполняется апплет, то строка, в том случае, если она находилась в движении, остановится. Для того чтобы возобновить движение строки, необходимо просто щелкнуть на области апплета еще раз.

Для того чтобы создать поток, апплет должен реализовать интерфейс Runnable. В результате реализации данного интерфейса апплет должен определить метод run ( ). Этот метод автоматически вызывается при запуске потока.

Для того чтобы отслеживать щелчки мышью, апплет может определить такие методы, как mouseDown(), mouseUp(), mouseEnter() и mouseExit().

```
Import java. awt.*;

import Java. applet.*;

public class AnimationApplet extends Applet implements Runnable

{

int xPos = 0;

int delay = 500;

boolean stopped = false;

String message = "";

Thread ap = null;

public void init() {

message = getParameter("message"); }

public void start () {

ap = new Thread(this);

ap. start(); }
```

```
public void stop() {
ap = null; }
public void run() {
Thread. currentThread( ).setPriority(Thread. MIN_PRIORITY);
while(ap!= null)
{
repaint () ;
try{
Thread. sleep(delay);
}catch(Exception ex){}
if(stopped = = false) {
xPos += 10;
if(xPos > 200)
xPos = 0; } }
}
public void paint(Graphics g)
{
g. setColor(new Color (0, 0, 0));
g. drawstring(message, xPos, 10);
}
public boolean mouseDown(Event e, int x, int y)
if(stopped = = false)
stopped = true;
else
stopped = false;
return false; }
}
}
```

Как видно из приведенного кода, апплет AnimationApplet реализует интерфейс Runnable. В методе start ( ) создается и запускается новый поток (объект Thread):

```
ap = new Thread(this); ap.start();
```

В результате запуска нового потока вызывается метод run (). Внутри этого метода организован бесконечный цикл, в теле которого, в частности, производится вызов метода repaint (). В результате вызова метода repaint () происходит, в свою очередь, вызов метода paint ().

```
while(ap != null)

{

repaint ( ) ;
```

После этого потоку дается команда "уснуть" (т. е. прекратить выполнение) на некоторое время.

```
try{

Thread. sleep(delay);

}catch(Exception ex){}
```

Одним из ключевых моментов, необходимых для создания анимации, является изменение значения переменной xPos, определяющей позицию в окне апплета, с которой будет начинаться вывод строки. Таким образом, при каждом вызове метода g. drawstring () в методе paint () строка текста будет отображена в новой позиции.

```
if(stopped = = false)

{

xPos +=10;

if(xPos > 200)

xPos = 0;

}
```

Переменная stopped является булевой переменной, которая определяет условия, при которых значение переменной xPos должно быть изменено. На самом деле, условие одно — если пользователь щелкнул мышью на области, в которой выполняется апплет, желая остановить бегущую строку, то, естественно, строка будет остановлена.

Значение переменной stopped изменяется в методе mouseDown(). Этот метод вызывается автоматически, когда пользователь щелкает мышью в области запуска апплета. Значение переменной stopped при этом изменится на противоположное (false на true, true на false), так что щелкнув на апплете еще раз, можно, в зависимости от состояния бегущей строки, вновь запустить или остановить ее.

```
public boolean mouseDown(Event e, int x, int y) {

if(stopped == false)

stopped = true;

else

stopped = false;

return false; }
```

На этом рассмотрение апплетов, к сожалению, приходится завершить. И хотя в данном разделе было разобрано всего лишь несколько практических примеров, на их основе можно создать много достаточно интересных типов анимации. Для того чтобы отобразить в области, отведенной для выполнения апплета, какое-нибудь изображение, следует обратиться к документации пакета JDK и просмотреть информацию по методу drawlmage () .

Контрольные вопросы

1. Объясните создание потока
2. Назначение апплета
3. Объясните метод start ( ), метод stop ( ), метод init( ) и update ( ).