**Applets**

The discussion that follows assumes that you are already familiar with the rudiments of creating a web page.

***Introduction***

An applet is a Java application that runs in a web page. Two components are needed to run an applet:

1. an HTML document that contains an applet markup tag
2. a byte code file for the applet, that is, a compiled Java applet in a **.class** file.

An applet markup tag has the following form:

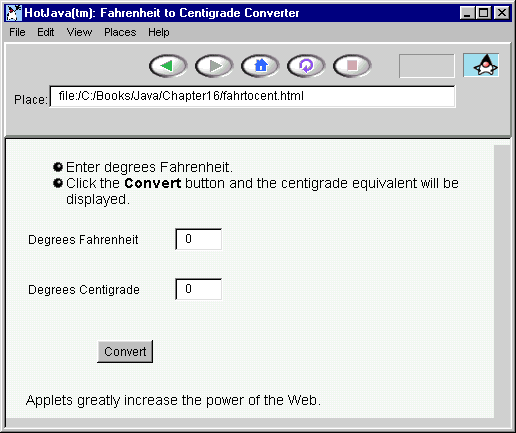
<APPLET CODE="*byte code file name"* WIDTH=*width* HEIGHT=*height*> </APPLET>

The width and height are the width and height, respectively, of the applet’s screen area within the browser window as measured in pixels.

For convenience, it is easiest to place applets and the HTML documents that use them in the same directory. This rule can be violated, but doing so involves some additional complexities that we are not going to discuss.

***Example***

Let us assume that the Fahrenheit to centigrade conversion program has already been rewritten as a Java applet. It might appear in a web page as shown in the next snapshot.



Here is the HTML code for the example:

<html>

<head>

<TITLE>Fahrenheit to Centigrade Converter</TITLE>

</head>

<body>

<UL>

<LI>Enter degrees Fahrenheit.

<LI>Click the <STRONG>Convert</STRONG> button and the

centigrade equivalent will be displayed.

</UL>

<APPLET CODE="FahrenheitToCentigrade.class" WIDTH=200 HEIGHT=150> </APPLET>

<P>

Applets greatly increase the power of the Web.

<body>

<html>

***Converting an Application to an Applet***

Previously, we have used the class GBFrame to provide the framework for GUI-based applications. We now show how to use a similar class, GBApplet, to write GUI-based applets. To convert Java applications to applets, we must do four things:

1. Replace the name GBFrame with the name GBApplet at the beginning of the class definition.
2. Delete the method main.
3. Eliminate any use of the setTitle method (discussed in a later section of the tutorial).
4. Replace the constructor, if any, by the method init:

public void init(){

...

}

The following listing shows these changes:

import java.awt.\*;

import BreezyGUI.\*;

public class FahrenheitToCentigrade extends GBApplet{

Label degreesFahrenheitLabel =

                  addLabel ("Degrees Fahrenheit",1,1,1,1);

IntegerField degreesFahrenheitField =

                  addIntegerField (0,1,2,1,1);

Label degreesCentigradeLabel =

                  addLabel ("Degrees Centigrade",2,1,1,1);

IntegerField degreesCentigradeField =

                  addIntegerField (0,2,2,1,1);

Button convertButton =

                  addButton ("Convert",3,1,2,1);

public void buttonClicked (Button buttonObj){

int fahrenheit;

int centigrade;

fahrenheit = degreesFahrenheitField.getNumber();

centigrade = (fahrenheit - 32) \* 5 / 9;

degreesCentigradeField.setNumber (centigrade);

}

}

***Constraints on Applets***

There are several constraints on the use of applets. Here they are:

1. **No menu bars --** Applets do not have menu bars, and consequently no pull-down menus; however, buttons often provide an acceptable substitute.

1. **No file access --** To ensure security on the user's machine, applets cannot access files. Imagine how dangerous it would be to download applets across the Internet if the applets could trash the files on your computer.

1. **Location of additional classes --** Java programs, whether they are stand-alone applications or applets, frequently utilize classes in addition to those in the standard Java libraries. These classes should be in the same directory as the applet. If the classes are part of a package, then do one of the following two things (depending on your browser one or both approaches may work):

a) Place a jar file contains the package in the same directory as the applet. For instance, include BreezyGUI.jar in the directory.

b) Create a subdirectory with the same name as the package and place all the package's classes in this subdirectory. For instance, you might create a subdirectory called BreezyGUI in which you would place the classes from the BreezyGUI package.

There are other alternatives that we do not discuss here (see [java.sun.com](http://java.sun.com)).

1. **Applets and browsers --** Not all browsers are capable of running applets; however, one applet ready browser is available from [java.sun.com](http://java.sun.com).

1. **Restrictions on dialogs --** The technique for defining dialogs to use in applications, as described later in this tutorial, applies to applets, with three qualifications:

* 1. The parameter of the dialog's constructor should not be the applet, but instead an anonymous frame. Thus, the value passed to the constructor could simply be new Frame().
  2. You will see a warning message at the bottom of the dialog.
  3. You can return to the web page while the dialog is still open. However, if you do so, you cannot interact with the applet, but you can browse to other pages, quit the browser, and perhaps hang the system.

***Passing Parameters to Applets***

It is possible to send information from an HTML page to an applet. The information is passed in HTML parameter tags and is retrieved in the applet's code. In the following example, a parameter tag binds the string "5" to the name “numberOfClasses.” The parameter tag must appear between the opening and closing applet tag:

<APPLET CODE="Courses.class" WIDTH=150 HEIGHT=100>

<PARAM NAME=numberOfCourses VALUE="5">

</APPLET>

At any point within the applet, the method getParameter can retrieve the parameter’s value, but always as a string:

String str = getParameter ("numberOfCourses");

int num = (new Integer(str)).intValue();

A common location for such code is in the init method.

If there are several parameters, each requires its own tag.