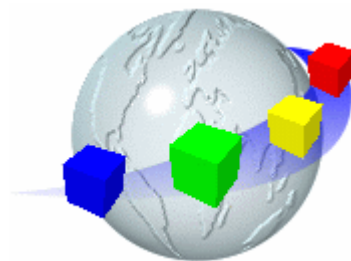


WebCab ColorPicker

WebCab
Components



Preface

The aim of this guide is to provide clear and concise guidance of all aspects of WebCab's ColorPicker JavaBeans component that are likely to be encountered on a day-to-day basis by developers. ColorPicker is a red-green-blue color picker of very good precision that integrates with any interface, allow colors to be changed on the fly.

This documentation contains a detailed description of the component packages content, functionality and applicability. An overview of the JavaBeans technology and its installation and use within IDEs is illustrated. Section three forms the Programmers guide which gives the developer a road map for taking advantage of every feature and functionality of the component. In the next chapter we provide a UML diagram of the ColorPicker Component. Finally, we introduce WebCab Components, its philosophy and approach to serving the Java and .NET development community with robust and powerful Components.

Good luck with your project and thank you for your interest in our component.

The WebCab Components team

Contents

Preface	i
1 Introduction	1
1.1 Product Description	1
1.1.1 Overview	1
1.1.2 Details	1
1.2 Prerequisites and Compatibility	1
1.2.1 System Requirements	1
1.2.2 Compatibility	2
2 JavaBeans Guide	3
2.1 Overview	3
2.2 Installing the JavaBeans	3
2.3 Using JavaBeans inside an IDE	4
2.3.1 Importing the <i>JAR</i> file into Sun's One IDE	4
2.3.2 Importing the <i>JAR</i> file into Borland's JBuilder IDE	4
2.4 Developer Support	5
2.4.1 Documentation	5
2.4.2 Compilation and Custom Modification of Source Code	5
2.4.3 Online Support	6
3 Programmer's Guide	7
3.1 Getting Started	7
3.2 The ColorPicker JavaBean	7
4 UML Diagrams	9
4.1 ColorPicker UML Diagram	10
5 Guide to WebCab Components	11
5.1 The Company	11
5.2 Presentation of Products	11
5.3 Supported Clients, IDEs, Containers and DBMSs	11
5.4 Transparent Functionality	12
5.5 Code Conventions	12
5.6 Company Culture and Activity	12
5.7 Product Life cycle	12
5.8 Support, Warranty and Upgrades	12

Chapter 1

Introduction

1.1 Product Description

1.1.1 Overview

Selects the color using the 256 scale for each of the base RGB colors. Both Java 1 (JDK 1.1) and Java 2 (JDK 1.2 and JDK 1.3) are supported.

1.1.2 Details

ColorPicker is a red-green-blue color picker of very good precision that integrates with any interface, allowing it to change on the fly.

This JavaBeans is designed to work within a graphical component to allow the user a more accessible and flexible interface. It is fully compatible with applets designed with the old JDK 1.1 and compatible with any Java Container.

The currently available functionality are:

- Flexibility - This JavaBeans is designed to work within a graphical component and can be applied on regions or text
- Precision - Each of red, green and blue are selected according to the full 256 color scale.

1.2 Prerequisites and Compatibility

1.2.1 System Requirements

- An Operating System running Java
- Pentium II 500 Mhz
- 64MB RAM

1.2.2 Compatibility

Operating System for Deployment

- Windows XP, 2000, 2003, NT, 9x
- Sun Solaris
- Linux
- IBM AIX
- HP-UX

Component Type

- JavaBeans

Built Using

- Java2 SDK Standard Edition 1.3.1

Compatible IDE's

- Borland JBuilder
- Sun's ONE IDE (formerly Forte for Java)
- Eclipse
- BEA Studio
- Oracle JDeveloper

Chapter 2

JavaBeans Guide

2.1 Overview

The JavaBeans technology offers a new perspective on how to divide the labor between software developers and graphic designers. It provides mechanisms that both minimize the design and implementation effort for modest system upgrades while fully supporting the design, implementation, and assembly of enterprise wide software solutions.

The distinguishing feature of JavaBeans components as reusable components is that they operate across all platforms supported by Java. This means that any JavaBeans component will work inside any operating system running a Java Virtual Machine, or even inside a browser supporting Java applets.

2.2 Installing the JavaBeans

This JavaBeans suite contains the following files:

- Documentation
 - Product Details
 - * Product Description
 - * System Requirements
 - * License Agreement
 - JavaDocs Files
 - JavaBeans Guide
 - Guide to WebCab Components
- JavaBeans Java Archive
- Examples and Demos

The documentation provided includes the description for the product, the *java generated documentation* with precise explanations for each *class*, *method* and *field*. The beans come wrapped into an archive with a *JAR* extension. This file contains the whole directory structure of the beans, additional classes and serialized components and a *manifest file*. The manifest file is required by IDEs to tell the Java beans from the other classes in

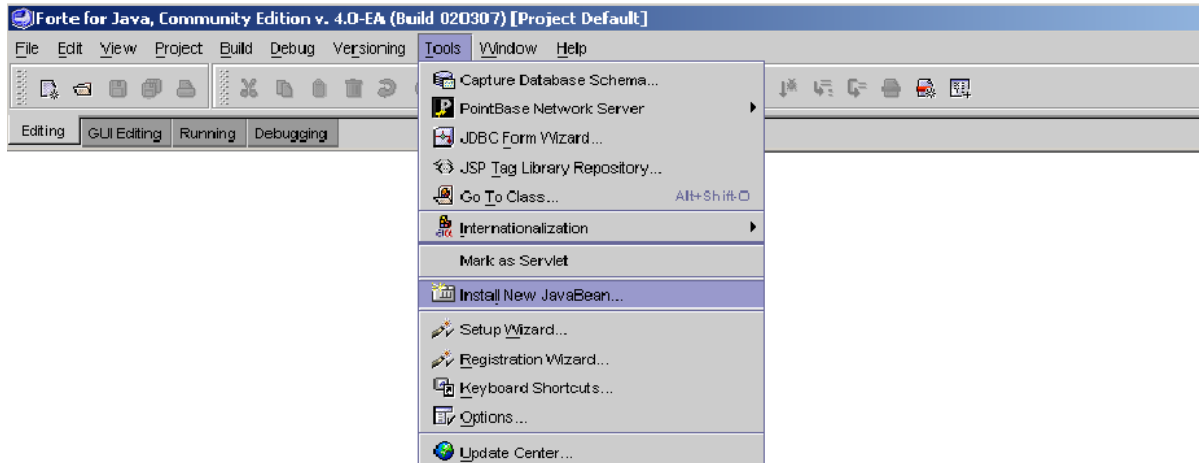
the archive. It is located in *META-INF/MANIFEST.MF*. After identifying each of these components, you may take the *JAR* file and import it into any compatible IDE.

2.3 Using JavaBeans inside an IDE

By importing the *JAR* file into a JavaBeans independent development environment (IDE), you can start adding the beans into your working area. As soon as you do this, you will be given the opportunity to edit their settings in a properties window. This is where attributes such as *font styles*, *colors*, *on/off switches*, *titles* can be fully customized, allowing the designer to control the design aspect of the bean without worrying about its implementation.

2.3.1 Importing the *JAR* file into Sun's One IDE

To make the *JAR* file available to the Sun's One IDE (formerly Forte For Java), first start up the IDE and select 'Tools > Instant New JavaBean'. Now select the *JAR* file which contains the JavaBeans by browsing the local file directory within the pop-up window and click OK. Another pop-up window will display the names of the JavaBeans contained within the *JAR* file. Select the beans you wish to install and click OK. In the next window please select the 'Palette Category' you wish to add these JavaBeans to and click OK. This completes the installation of the JavaBean component to the Palette from which the beans can be used within future application development.

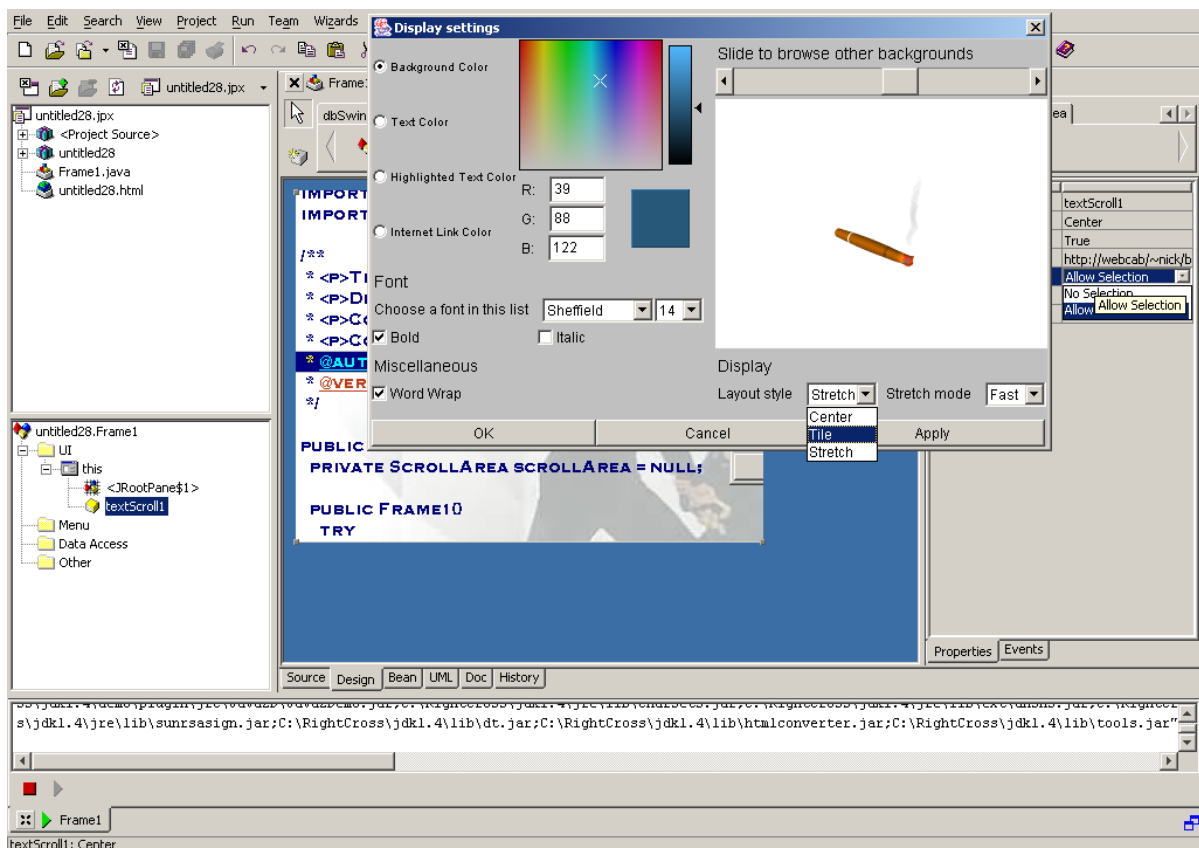


2.3.2 Importing the *JAR* file into Borland's JBuilder IDE

To import the *JAR* file containing the JavaBean components into a JBuilder project select from the menu 'Tools > Configure Palette'. Now a pop-up window will appear which has two tags, 'Pages' and 'Add Component'. You can either choose to create a page for the imported JavaBeans (i.e. *JAR*) or they may be added to an existing page. To create a new page click on 'Add' in 'Pages', chosen a name and click OK. To add the *JAR* file first select the 'Add Components' tab and then click on 'Select Library'. Another window will pop-up entitled 'Select a different library', if the *JAR* file is not displayed in the 'user home' folder then click the 'New' button. Now within the pop-up entitled 'New Library Wizard' enter the name you wish to give the imported library of JavaBeans contained

within the *JAR* and click ‘Add’. Select the *JAR* file by browsing your local file directory in the pop-up window and then click OK. You should have now been returned to the ‘New Library Wizard’ with the *JAR* file location inserted within the ‘Library Paths’, if this is the case click OK, otherwise click ‘Add’ and repeat the previous step. Select the *JAR*’s library name within the ‘Select a different library’ window and click OK. To finish click the ‘Add from Selected Library’ button within the ‘Palette Properties’ window. The results of the import will be displayed within a ‘Results’ window which completes the imported onto the JBuilder Palette.

Below is a screen shot of working with JGraph inside the JBuilder software.



2.4 Developer Support

2.4.1 Documentation

In order to use the JavaBeans component in any application, you should first read the information provided in the *Product Description* and *System Requirements*. As soon as you are aware of any incompatibilities that might turn up, you may start browsing the *Java Generated Documentation* (JavaDocs) to understand the structure of each bean.

2.4.2 Compilation and Custom Modification of Source Code

All WebCab Components are compiled using Sun’s JDK1.3, and should you prefer compilation of the source code to be done using another compiler then please contact us. Some functions (for example standard mathematical operations) are embedded within the source

code, thus, if you license classes which you believe offer a similar level of functionality from a third party and wish to include these classes within our components then please contact us. Both of the above services are offered free of charge at WebCab's discretion.

2.4.3 Online Support

Should you have any questions or queries, please feel free to email us at:

support@WebCabComponents.com

For updates and new releases, visit:

www.WebCabComponents.com

Chapter 3

Programmer's Guide

This section will explain every detail and aspect a programmer needs to know in order to fully exploit the features and capabilities of the WebCab ColorPicker JavaBeanTM component.

3.1 Getting Started

WebCab ColorPicker is a JavaBean component and because of this, it is meant to be used as a part of a Java-based application. There are at least two ways of embedding it inside an application; one is by deploying the JavaBean inside a Development Environment, such as Borland's JBuilder or Eclipse, the other way is by direct implementation. Deployment inside a Development Environment is easy-to-use, comprehensive and does not require intricate knowledge about Java as a programming language. Yet, in order to entirely benefit the advantages a JavaBean can offer, the developer will need to acquire a deeper understanding of the ways it works.

This information can be obtained from this PDF documentation and this products JavaDocs, located with the JavaDoc subdirectory. After getting acquainted with the basics of WebCab ColorPicker, you will be - almost exclusively - using the JavaDocs, where details concerning all classes, their fields and methods is described.

3.2 The ColorPicker JavaBean

The ColorPicker JavaBean is represented by several non-public classes, such as *Gradient*, *ColorPlate* and *PlateCreationThread*, and a central public JavaBeanTM component class, named *ColorPicker*. This bean represents an important part of the communication bridge between administrator and the end-user, because it allows the user to personalize the background color and text color, with a simple drag of mouse. Since it is only used inside the "Configuration Dialog", the color picker has little development importance, so that methods provided require minimum configuration effort and time.

In order to start using the WebCab Color Picker inside your Java-based application, you will need to create an instantiation of the bean, with the following code:

```
import java.beans.*;
import com.webcab.chat.gui.*;

...

try {
    ColorPicker cPicker = (ColorPicker) Beans.instantiate (
        getClass ().getClassLoader (), "com.webcab.chat.gui.ColorPicker");
}
catch (IOException e) {
}
catch (ClassCastException e) {
}
```

After instantiating the *ColorPicker* bean, you may use the following two methods to access the picker's color:

1. **Color getColor** - This method tells the developer what color is currently displaying inside the color picker.
2. **void setColor (Color newColor)** - In order to initialize the color picker with a new color, invoke this method with a corresponding **newColor** argument.

Chapter 4

UML Diagrams

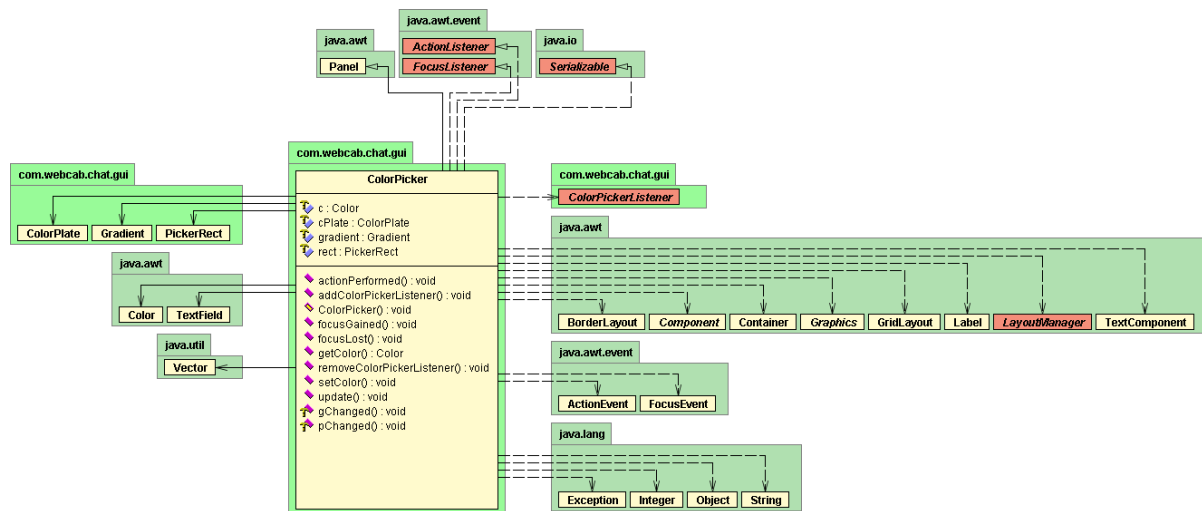
Within this chapter we provide UML diagram(s) for the interfaces of the ScrollArea component. The diagrams use standard UML notation detailing the inheritance, dependency, association, interface, methods, fields and properties of this component and associated packages. We present the UML diagrams with a large amount of white space so that the developer has space in which to add additional remarks.

UML Diagram Key

- **Extended Classes (interfaces)** - A solid line with a large triangle that points from the subclass (resp subinterface) to the superclass (resp inherited interface)
- **Classes** - Displayed in a rectangular box with a default yellow background with the name at the top and field, methods, and properties listed below it
- **Abstract Class and Methods** - Displayed in italic font
- **Implementing Classes** - A dashed line with a large triangle which points from the implementing class to the inherited interface
- **Interfaces** - A rectangle with orange background and the interface name in italic font
- **Implemented Interfaces** - A dashed line with a large triangle which points from the implementing class to the implemented interface
- **Dependencies/Reverse Dependencies** - A dashed line with arrowhead
- **Associations/Reverse Associations** - A solid line with an arrowhead
- **Packages** - A green rectangle with a tab and the package name in the tab or below it
- **Methods** - Listed below members and fields, including the return type
- **Members/fields** - Listed below the class name, including the return type
- **Properties** - Properties are displayed separately in the bottom of the class diagram
- **Static** - Static members, fields, variables, and methods are underlined in the diagram

4.1 ColorPicker UML Diagram

Zoom to at least 360% in order to be able to read the names of the fields, methods, component names and packages.



Chapter 5

Guide to WebCab Components

5.1 The Company

WebCab is a privately owned British company that has built business solutions since its inception in 1999. We continue to refine and develop our Mathematical and Financial Framework which we have implemented on the J2SE, J2EE and .NET platforms.

5.2 Presentation of Products

We aim to provide good quality, useful information to help you decide which J2EE application best suits your development needs. WebCab is committed to honesty and realism when presenting our products. In order to achieve this a detailed, clear and factual style for presentation is adopted within our documentation and marketing material.

5.3 Supported Clients, IDEs, Containers and DBMSs

We support all major development, server and client side technologies. Each product contains detailed examples and advice concerning the integration of our enterprise applications within existing infrastructure. Our documentation provides the information that the developer needs to get their applications up and running as quickly and as easily as possible.

We detail exactly how the developer can use our J2EE Applications with the following technologies:

- Client containers - Internet Explorer, Netscape Navigator, Opera
- IDEs - JBuilder, Eclipse, BEA Studio, Sun ONE (formerly Forte For Java), IBM VisualAge, Oracle JDeveloper
- Client Side Technologies - Application Clients, Servlet, JSP, Applets
- EJB containers - IBM WebSphere, BEA WebLogic, Oracle 9iAS, Sun ONE AppServer, Borland AppServer, Sybase EAServer, Ironflare Orion, JBoss
- DBMS - Oracle, IBM DB2, SQL Server, Sybase, MySQL
- Platforms - Windows XP/2000/NT, Linux, Solaris, IBM AIX

For these technologies we include all the deployment descriptors, installation scripts and template examples which will help the developer quickly and easily assemble their multi-tier enterprise application.

5.4 Transparent Functionality

All technical and business intelligence incorporated within our products is described within the associated documentation. This allows the developer to see the nature of the methodology implemented within our proprietary algorithms.

5.5 Code Conventions

WebCab adheres to the ‘Code Conventions for the Java Programming Language’ as specified on April 20, 1999 by Sun Microsystems Inc. These conventions cover filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions and programming practices. Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

5.6 Company Culture and Activity

WebCab Components is focused solely on the production of high quality software modules within our Financial and Mathematical Framework. The WebCab team contains a wide range of expertise and experience from the academic, investment banking and software development worlds.

Within the company there exists significant internal competitive pressures with constant peer review and evaluation, resulting in higher quality products, which adhere to high professional standards and offer extended functionality.

5.7 Product Life cycle

We continuously add value to our products by evolving them according to new J2EE specifications, customer feedback and market demands. We give particular emphasis to incorporating our clients suggested modifications and additions into our product development cycle.

5.8 Support, Warranty and Upgrades

WebCab warrants that each component will perform substantially in accordance with the accompanying written material. We provide with all our products without charge sixty (60) days product support services including fixing bugs, compatibility issues and other technical support issues.

All maintenance updates (including service packs) will be distributed free of additional license cost. New version releases of this product will be offered to present licensee holders

at a greatly reduced rate.

Dr Ben Fairfax

Founder and CEO

WebCab Components - From Developer, To Developer

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. .NET is a trademark of Microsoft Corporation in the U.S. and other countries