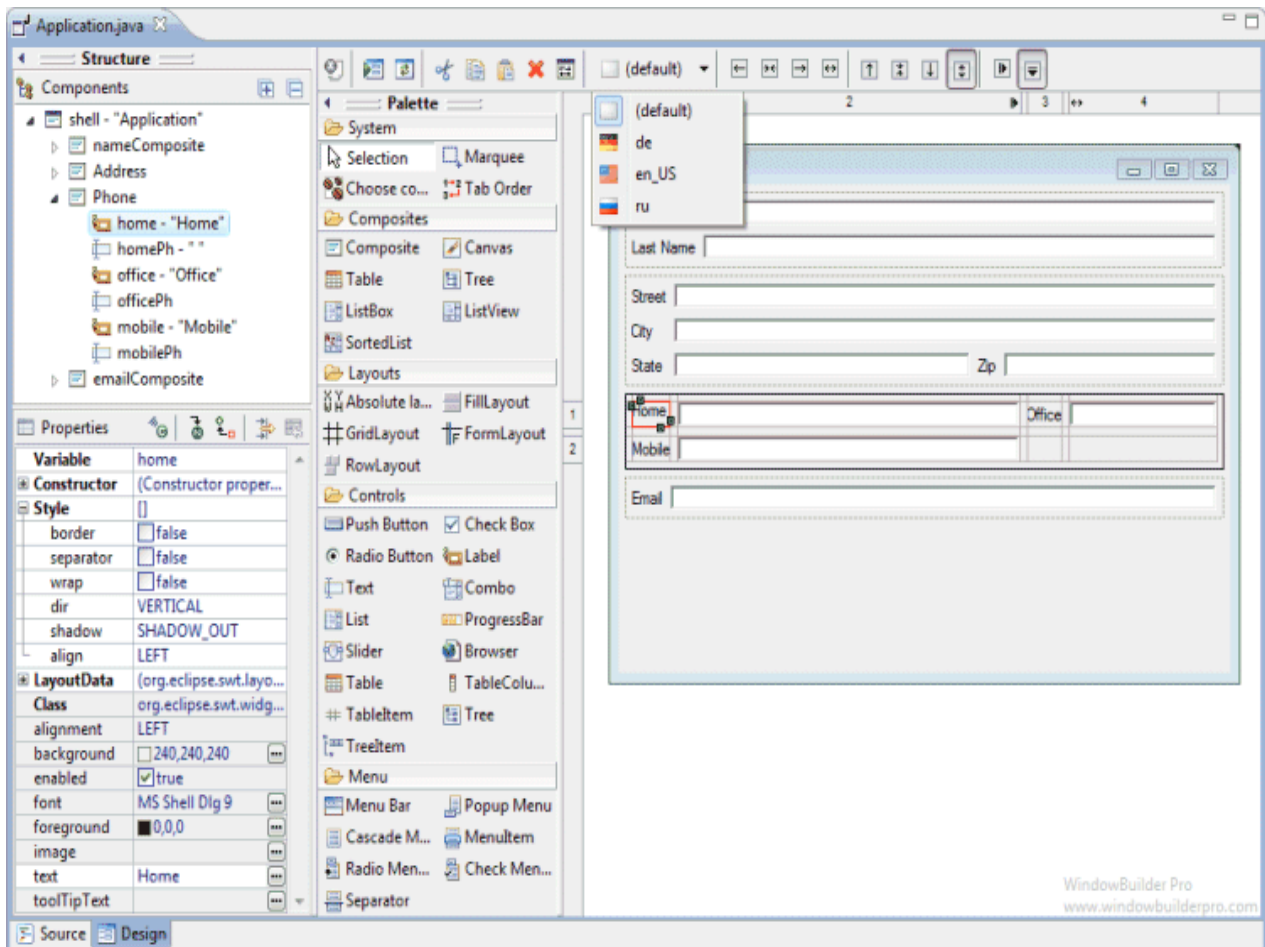


WindowBuilder Pro Tutorial

Informatica III – B Angelo Gargantini 2011

Materiale preso dal sito di google



Vedremo i seguenti passi:

1. Install the latest WindowBuilder Pro build into Eclipse 3.4, 3.5 or 3.6.
2. Create a new project using the appropriate project wizard.
3. Use a class wizard to create your first GUI window or edit an existing window
4. Choose an appropriate layout manager for your window.
5. Add various components from the Palette and edit their properties using the Property Pane.
6. Visually create and edit your window's menubar.
7. Add event handlers to various widgets to provide behavior.
8. If you create a new Composite or JPanel, you can embed it in another window using the Choose Component command.
9. You can test launch your window by right-clicking on it and selecting Run As > Java Application.
10. Internationalize your application and manage multiple locales.

1. *Install the latest WindowBuilder Pro build into Eclipse*

If you are already familiar with installing Eclipse plugins, you can just use the update site URL below for your version of Eclipse.

Eclipse 3.7 (Indigo)

<http://dl.google.com/eclipse/inst/d2wbpro/latest/3.>

2. *Create a new project using the appropriate project wizard*

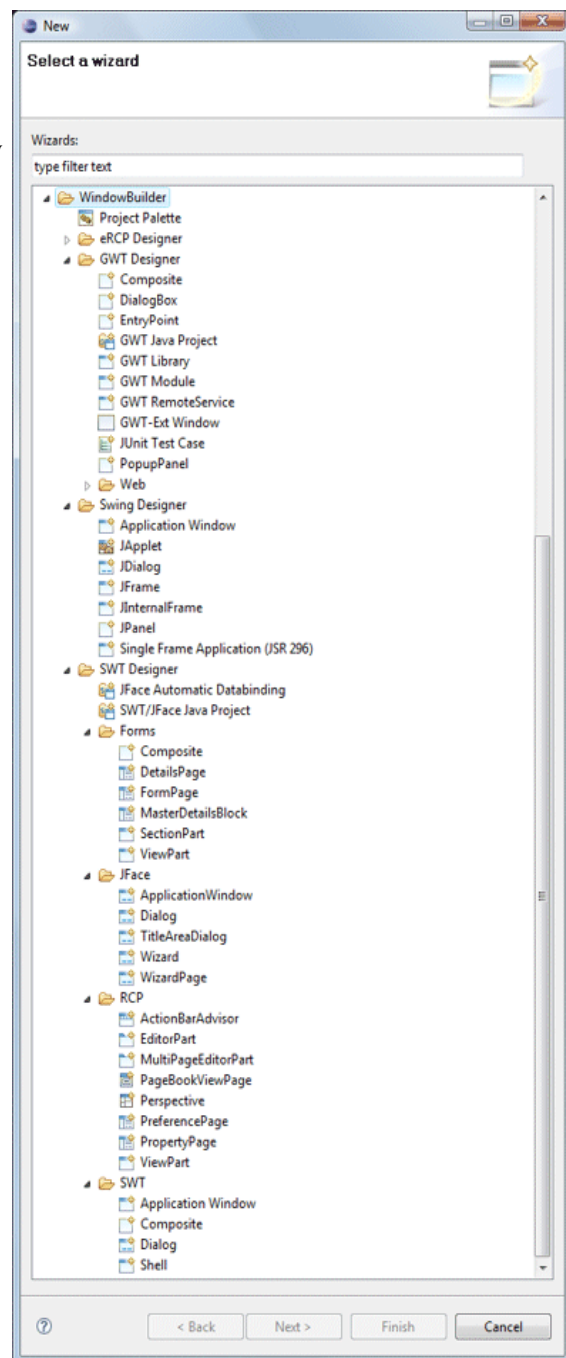
WindowBuilder Pro provides a large number of wizards. Each toolkit contributes one or more wizards.

The wizards may be accessed from the Eclipse **New** wizard (**File > New > Other**) or the drop down wizard menu in the Eclipse toolbar.

Ci sono tantissimi oggetti:

Quelli più usati sono

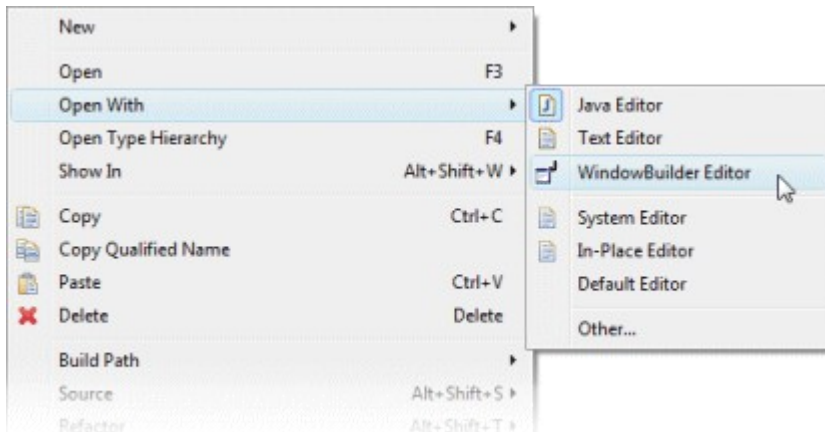
- JFrame
- Jdialog



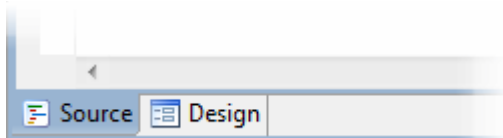
3. *Use a class wizard to create your first GUI window or edit an existing window*

Editing Existing Windows

To open a file that has not been created by the tool or that is currently edited by a different editor (such as the standard **Java Editor**), simply right-click on the file and select **Open With > WindowBuilder Editor**.



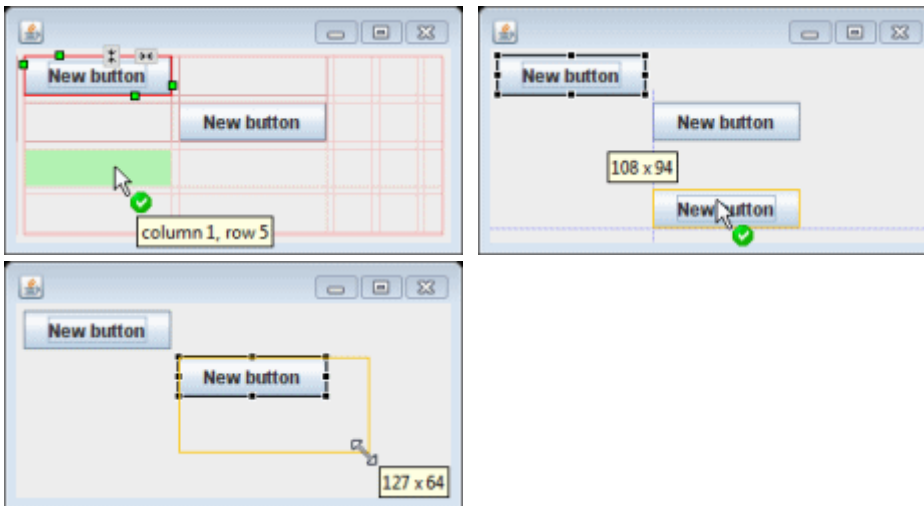
This will open the file in the **WindowBuilder** Editor in the [Source View](#) by default (the editor layout may be easily customized if you prefer the [Design View](#) to appear first or if you want the [Source View](#) and [Design View](#) to appear at the same time). There are two tabs at the bottom of the editor window labeled **Source** and **Design** as shown below. If you click on the **Design** tab or hit **F12**, the file will be displayed in the [Design View](#) and you will be able to visually design your GUI.



Note: If a file was last opened in the **WindowBuilder** Editor, it will continue to open in that editor until a different editor is selected to edit the file. If you do not see the **Source** and **Design** tabs, this means that you are not using the **WindowBuilder** Editor.

4. *Choose an appropriate layout manager for your window - Layout Managers*

A program written in Java may be deployed on multiple platforms. If you were to use standard UI design techniques, specifying absolute positions and sizes for your UI components, your UI won't be portable. What looks fine on your development system might be unusable on another platform. To solve this problem, Java provides a system of portable layout managers. You use these layout managers to specify rules and constraints for the layout of your UI in a way that will be portable.



Layout managers can provide the following advantages:

- Correctly positioned components that are independent of fonts, screen resolutions, and platform differences.
- Intelligent component placement for containers that are dynamically resized at runtime.
- Ease of translation. If a string increases in length after translation, the associated components stay properly aligned.

About layout managers

When writing Java applications, you may need to use layouts to give your windows a specific look. A layout controls the position and size of children in a container. Layout classes are subclasses of the abstract class `Layout`. Both SWT and Swing provide several standard layout classes, and you can write custom layout classes. Other UI toolkits embed implicit layout managers in each panel type.

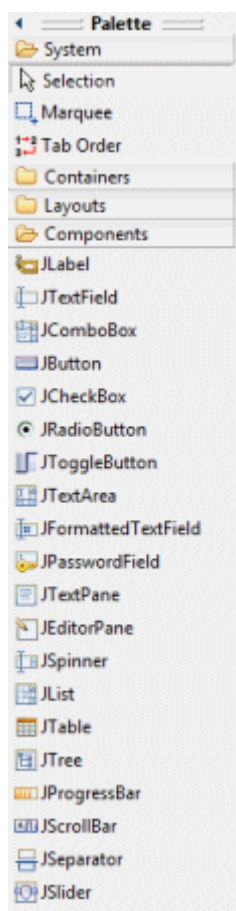
In Java, positioning and sizing does not happen automatically. Applications can decide to size and place a container's children initially, or in a resize listener. Another option is to specify a layout class to position and size the children. If children are not given a size, they will have zero size and they cannot be seen.

The layout manager sets the sizes and locations of the components based on various factors such as:

- The layout manager's layout rules.
- The layout manager's property settings, if any.
- The layout datas associated with each component.
- The size of the container.

Each layout manager has characteristic strengths and drawbacks. Grid-based layouts and constraint-based layouts are the most useful and powerful of the standard layouts, but they are also the most complicated. When using the design view, you can change the layout whenever you like. The tool adjusts the code as needed on the fly. Change the layout either by explicitly adding a layout manager to the source code for the container, or by selecting a layout for the composite on the design surface.

5. Add various components from the Palette and edit their properties using the Property Pane



The **Palette** provides quick access to toolkit-specific components as well as any custom components installed by the user. The **Palette** is organized into categories which may be expanded, collapsed or hidden.

To add a components to the [Design View](#), you can:

- Select it in the palette and drop it on the [Design View](#) or [Component Tree](#) by clicking in the desired location.
- Use the **Choose Component** command to select the widget type from **Choose Component** dialog.

Multiple widgets of the same type may be added to the [Design View](#) by holding down the **Ctrl** key when selecting the widget in the **Palette**.

The palette may be customized by right-clicking anywhere in the palette to access the [palette context menu](#) or by opening the [Palette Manager](#) dialog..

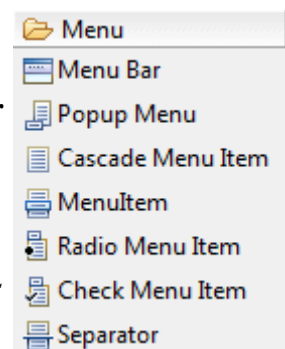
The following commands are common to every palette:

- Selection
- Marquee
- Choose component
- Tab Order

6. Visually create and edit your window's menubar

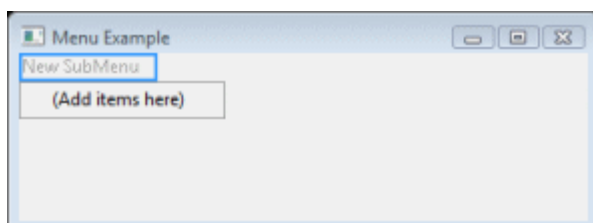
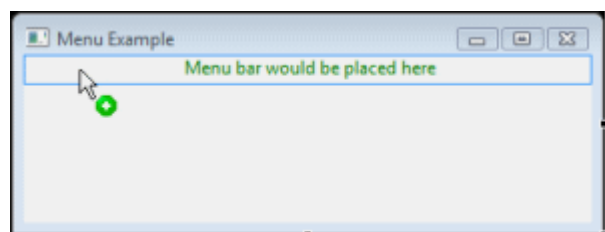
Menus are a must for just about all GUIs. Using the editor, it is quick and painless to create menu bars and popup menus.

To create a menu bar on your window, select **Menu Bar** (SWT) or **JMenuBar** (Swing) from the [Palette](#), and then place it on your window in the [Design View](#).

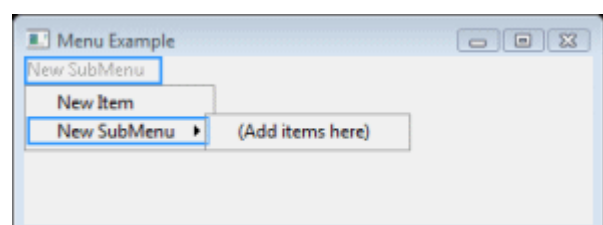


Note that these instructions only apply to window types that support menubars (e.g., SWT Shells, SWT Application Windows, and Swing JFrames). They do not apply to window types that can't have menubars (e.g., Dialogs & Wizards) or that manage their own menubars through a special framework (e.g., JFace ApplicationWindow).

Each window can have at most one **Menu Bar**, and the only place that you can put a **Menu Bar** is directly on a window. A particular menu will only be displayed on the window on which it was created. After the **Menu Bar** is placed, you should see a blank menu bar across the top of your window in the [Design View](#).



To




add individual menus to the menu bar, select **Cascade Menu Item** (SWT) or **JMenu** (Swing) from the [Palette](#), and then place it on the menu bar.

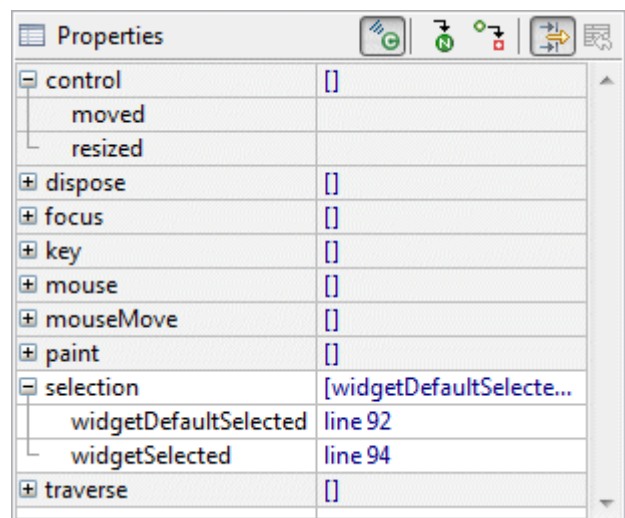
7. Add event handlers to various widgets to provide behavior

Events occur when the user interacts with the UI. The appropriate event-handling code is then executed. In order to know when events occur, event handlers must first be added to your components. The tool makes it very easy to add and remove event listeners to your components.

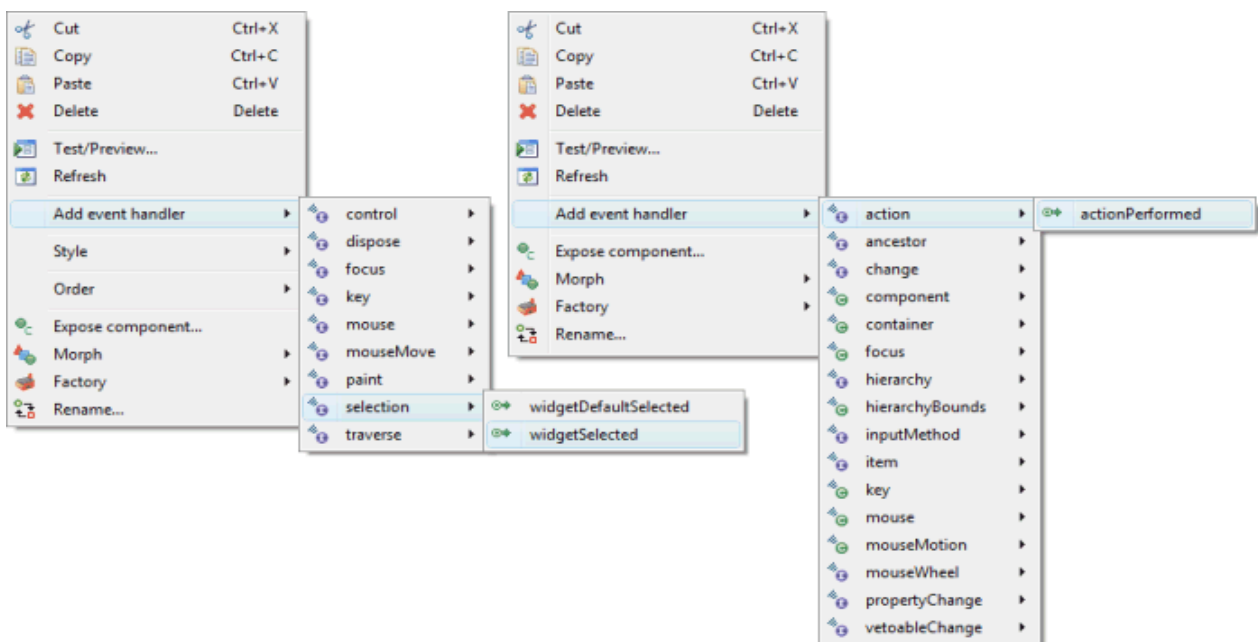
8. Adding an Event Handler

There are two ways to add an event handler with the the editor. The first way is through the [Property Pane](#). First select a component in either the [Component Tree](#) or the [Design View](#). Click the **Show Events**  button to reveal the events list in the [Property Pane](#). Expand an event and double-click or press **Enter** on the event handler you wish to implement.

The second way to add an event handler is to simply [right-click](#) on a component (either in the [Design View](#) or in the [Component Tree](#)), select **Add event handler** > [name of the event] > [name of the event handler to implement].



A quick way to add event listeners to buttons (including check and radio buttons) and menu items is to simply double-click on the button or menu item. When double-clicked, a selection event listener




will be created.

Any way you add an event, the tool will automatically create an event handler for the selected event method. The editor will then switch to the [Source View](#) and go directly to the new event handler

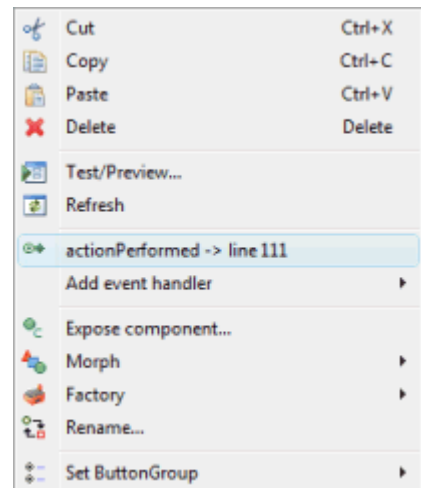
method. The code generated looks like the following:

```
browseButton = new Button(shell, SWT.NONE);
browseButton.addSelectionListener(new SelectionListener() {
    public void widgetDefaultSelected(SelectionEvent arg0) {
    }
    public void widgetSelected(SelectionEvent arg0) {
    }
});
browseButton.setText("Browse...");
```

There are two ways to quickly return to the event-handling code from the [Design View](#). In the [Property Pane](#), click the **Show Events**  button and expand the event. Then double-click or press **Enter** on the event method to return to the code. The events' list in the [Property Pane](#) is a list of all the events implemented, and each individual event's property value is the list of methods implemented. If a method has been implemented, its property value is the line number in the source code where its event-handling code begins.

The other way to quickly return to the event-handling code is to right-click on a component (either in the [Design View](#) or in the [Component Tree](#)), and then select the menu item for the correct method that was implemented. Shown to the right is

"**actionPerformed -> line 111**" being selected. In this example, "actionPerformed" is the event method, and "111" is the line number in the source code on which the method begins.



Note: the tool is also able to generate an event handler stub method when a new event is created by setting the appropriate code generation preferences.

9. Deleting an Event Handler

There is only one way to delete an existing event handler in the editor. Select a component in either the [Design View](#) or in the [Component Tree](#). In the [Property Pane](#) expand expand the event, click on the event method you wish to delete and then press **Delete**. If there are no other methods that have been implemented for that event, the tool will also delete the event listener for that event.

10. If you create a new Composite or JPanel, you can embed it in another window using the Choose Component command.

11. You can test launch your window by right-clicking on it and selecting Run As > Java Application.

12. Internationalize your application and manage multiple locales.