

L'oggetto Controller  
 deve poter accedere all'oggetto di Vista  
 e agli oggetti MODEL

L'oggetto di Vista  
 deve poter accedere all'oggetto Controller

L'oggetto di vista dispone di metodi  
 EVENT HANDLER

Cogni metodo handler manda in esecuzione  
 il metod. di Control che dà luogo  
 alla azione richiesta

CiaoMondoGUI - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Files : Serv... Start Page x CiaoMondoGUIApp.java x CiaoMondoGUIView.java x

CiaoMondoGUI

- Source Packages
  - META-INF.se
  - ciaomondogu
    - CiaoMon
    - CiaoMon
    - CiaoMon
  - ciaomondogu
  - ciaomondogu
  - model
  - Contator
- Libraries

incrementaEMostra ...

Members View

CiaoMondoGUIApp ::

- configureWindow
- getApplication()
- incrementaEMost
- main(String[] arg
- startup()
- conta : Contator
- vista : CiaoMond

**New Project**

**Steps**

1. Choose Project
2. ...

**Choose Project**

Categories:

- Java
- Maven
- PHP
- NetBeans Modules
- Samples

Projects:

- Java Application
- Java Desktop Application
- Java Class Library
- Java Project with Existing Sources
- Java Free-Form Project

Description:

Creates a skeleton of a desktop application based on the Swing Application Framework (JSR 296). This template provides basic application infrastructure such as a menu bar, persisting of window state, and status bar. With this template, you can also generate code to create a GUI interface for a database table.

< Back Next > Finish Cancel Help

2 | 54 | 5 | INS

CiaoMondoGUI - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Start Page x CiaoMondoGUIApp.java x CiaoMondoGUIView.java x

CiaoMondoGUI

- Source Packages
  - META-INF.se
  - ciaomondogu
    - CiaoMon
    - CiaoMon
    - CiaoMon
  - ciaomondogu
  - ciaomondogu
  - model
  - Contator
- Libraries

incrementaEMostra ...

Members View

CiaoMondoGUIApp ::

- configureWindow
- getApplication()
- incrementaEMost
- main(String[] arg
- startup()
- conta : Contator
- vista : CiaoMond

### New Desktop Application

**Steps**

1. Choose Project
2. Disclaimer
3. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

Application Class:

Choose Application Shell

- Basic Application
- Database Application

A basic application skeleton containing everything needed to start with a general desktop application, such as the following:

- main frame with a menu bar and status bar
- About box
- main Application class

Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

Set as Main Project

< Back    Next >    Finish    Cancel    Help

2 | 54 | 5 | INS

NetBeans IDE 7.0.1 window titled "BottoneSingolo". The interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a search bar (Search (Ctrl+I)), and a toolbar with icons for file operations and development actions.

The main workspace is divided into several panels:

- Files:** Shows the project structure for "BottoneSingolo". The tree includes "Source Packages" (META-INF.services, bottonesingolo) and "Libraries" (CiaoMondoGUI). The "bottonesingolo" package contains "BottoneSingoloAboutBox.java", "BottoneSingoloApp.java", and "BottoneSingoloView.java".
- Inspector:** Displays the component hierarchy for the selected "Form BottoneSingoloView". The tree shows "[FrameView]" containing "mainPanel [JPanel]", "menuBar [JMenuBar]", and "statusPanel [JPanel]".
- Palette:** Lists "Swing Containers" such as Panel, Tabbed Pane, Split Pane, Scroll Pane, Tool Bar, Desktop Pane, Internal Frame, and Layered Pane.
- Properties:** Shows the properties for the selected "[FrameView]". The "Properties" tab is active, displaying a table of component names and their corresponding classes:

Component Name	Class Name
component	mainP...
menuBar	menuBar
statusBar	status...
toolBar	<none>

The central editor area shows the "Design" view of the "BottoneSingoloView.java" form. A tooltip message states: "The Inspector window displays a tree hierarchy of components in the opened form." The form itself is a simple window with a title bar containing "File" and "Help" menus, and a single text input field at the bottom.

The **Output - CiaoMondoGUI (run)** panel shows the following log:

```
run:
BUILD SUCCESSFUL (total time: 15 seconds)
```

The status bar at the bottom indicates 2 files, 54 lines, and 5 instances (INS).

NetBeans IDE 7.0.1 window showing the development of a Java GUI application named "BottoneSingolo".

**Project Structure (Left Panel):**

- BottoneSingolo
  - Source Packages
    - META-INF.services
    - bottesingolo
      - BottoneSingoloAboutBox.java
      - BottoneSingoloApp.java
      - BottoneSingoloView.java
    - bottesingolo.resources
    - bottesingolo.resources.busyicons
  - Libraries
  - CiaoMondoGUI
    - Source Packages
      - META-INF.services
      - ciaomondogui
        - CiaoMondoGUIAboutBox.java

**control - Navigator**

Members View

- BottoneSingoloView :: FrameView
  - BottoneSingoloView(SingleFrameApplication a
  - initComponents()
  - showAboutBox()
  - aboutBox : JDialog
  - busyIconIndex : int
  - busyIconTimer : Timer
  - busyIcons : Icon[]
  - control : BottoneSingoloApp
  - idleIcon : Icon
  - mainPanel : JPanel
  - menuBar : JMenuBar

```

10  import org.jdesktop.application.FrameView;
11  import org.jdesktop.application.TaskMonitor;
12  import java.awt.event.ActionEvent;
13  import java.awt.event.ActionListener;
14  import javax.swing.Timer;
15  import javax.swing.Icon;
16  import javax.swing.JDialog;
17  import javax.swing.JFrame;
18
19  /**
20   * The application's main frame.
21   */
22  public class BottoneSingoloView extends FrameView {
23
24      private BottoneSingoloApp control;
25
26      public BottoneSingoloView(SingleFrameApplication app) {
27          super(app);
28
29          initComponents();
30
31          // status bar initialization - message timeout, idle icon and busy animatio
32          ResourceMap resourceMap = getResourceMap();
33          int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");

```

**Output - CiaoMondoGUI (run)**

```

run:
BUILD SUCCESSFUL (total time: 15 seconds)

```

NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Proj... Files Services

**BottoneSingolo**

- Source Packages
  - META-INF.services
  - bottesingolo
    - BottoneSingoloAboutBox.java
    - BottoneSingoloApp.java
    - BottoneSingoloView.java
  - bottesingolo.resources
  - bottesingolo.resources.busyicons
- Libraries
- CiaoMondoGUI
  - Source Packages
    - META-INF.services
    - ciaomondogui
      - CiaoMondoGUIAboutBox.java

**Members View**

BottoneSingoloView :: FrameView

- BottoneSingoloView(SingleFrameApplication app)
- initComponents()
- showAboutBox()
- aboutBox : JDialog
- busyIconIndex : int
- busyIconTimer : Timer
- busyIcons : Icon[]
- control : BottoneSingoloApp
- idleIcon : Icon
- mainPanel : JPanel
- menuBar : JMenuBar

Output - CiaoMondoGUI (run)

```
run:
BUILD SUCCESSFUL (total time: 15 seconds)
```

Tasks

Source Design

```

10 import org.jdesktop.application.FrameView;
11 import org.jdesktop.application.TaskMonitor;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import javax.swing.Timer;
15 import javax.swing.Icon;
16 import javax.swing.JDialog;
17 import javax.swing.JFrame;
18
19 /**
20  * The application's main frame.
21  */
22 public class BottoneSingoloView extends FrameView {
23
24     private BottoneSingoloApp control; //riferimento al controller
25
26     public BottoneSingoloView(SingleFrameApplication app) {
27         super(app);
28
29         control=(BottoneSingoloApp) app;//attilva il riferimento al controller
30
31         initComponents();
32
33         // status bar initialization - message timeout, idle icon and busy animatio

```

2 | 29 | 78 | INS

NetBeans IDE 7.0.1 window showing the development of a Java application named **BottoneSingolo**.

The main editor displays the source code for **BottoneSingoloApp.java**. The code defines a class `BottoneSingoloApp` that extends `SingleFrameApplication`. A red circle highlights the following line:

```
15 private BottoneSingoloView vista; //riferimento all'oggetto di vista
```

The `startup()` method is implemented as follows:

```
18 /**
19  * At startup create and show the main frame of the application.
20  */
21 @Override protected void startup() {
22     show(new BottoneSingoloView(this));
23 }
24 /**
```

The **Output - CiaoMondoGUI (run)** window shows the following output:

```
run:
BUILD SUCCESSFUL (total time: 15 seconds)
```

The **Navigator** window shows the class hierarchy for `BottoneSingoloApp`:

- `BottoneSingoloApp :: SingleFrameApplication`
  - `configureWindow(Window root)`
  - `getApplication() : BottoneSingoloApp`
  - `main(String[] args)`
  - `startup()`
  - `vista : BottoneSingoloView`

The **Files** view on the left shows the project structure:

- BottoneSingolo**
  - Source Packages
    - META-INF.services
    - bottesingolo
      - `BottoneSingoloAboutBox.java`
      - `BottoneSingoloApp.java`
      - `BottoneSingoloView.java`
    - bottesingolo.resources
    - bottesingolo.resources.busyicons
  - Libraries
  - CiaoMondoGUI
    - Source Packages
      - META-INF.services
      - ciaomondogui
        - `CiaoMondoGUIAboutBox.java`



NetBeans IDE 7.0.1 window showing the development of a Java application named **BottoneSingolo**.

The main editor displays the code for **BottoneSingoloApp.java**:

```

import org.jdesktop.application.SingleFrameApplication;

/**
 * The main class of the application.
 */
public class BottoneSingoloApp extends SingleFrameApplication {

    private BottoneSingoloView vista; //riferimento all'oggetto di vista

    /**
     * At startup create and show the main frame of the application.
     */
    @Override protected void startup() {
        vista = new BottoneSingoloView(this); //istanzia l'oggetto di vista
                                                //lo riferenzia con l'attributo vista
        show(vista);
    }

    /**
     * This method is to initialize the specified window by injecting resources.
     * Windows shown in our application come fully initialized from the GUI
     * builder, so this additional configuration is not needed.
     */
    @Override protected void configureWindow(java.awt.Window root) {

```

Annotations in the code:

- A red circle highlights the line: `private BottoneSingoloView vista; //riferimento all'oggetto di vista`.
- A red oval highlights the lines: `vista = new BottoneSingoloView(this); //istanzia l'oggetto di vista` and `show(vista);`.

The **start-up - Navigator** window shows the class hierarchy:

- BottoneSingoloApp** :: **SingleFrameApplication**
  - `configureWindow(Window root)`
  - `getApplication() : BottoneSingoloApp`
  - `main(String[] args)`
  - `startup()`
  - `vista : BottoneSingoloView`

The **Output - CiaoMondoGUI (run)** window shows the following output:

```

run:
BUILD SUCCESSFUL (total time: 15 seconds)

```

The status bar at the bottom indicates 2 files, 22 lines, 83 characters, and the keyboard layout is set to **INS**.