**NetBeans IDE Java Quick Start Tutorial**

Welcome to NetBeans IDE!

This tutorial provides a very simple and quick introduction to the NetBeans IDE workflow by walking you through the creation of a simple "Hello World" Java console application. Once you are done with this tutorial, you will have a general knowledge of how to create and run applications in the IDE.

This tutorial takes less than 10 minutes to complete.

After you finish this tutorial, you can move on to the learning trails, which are linked from the [Documentation, Training & Support](https://netbeans.org/kb/index.html) page. The learning trails provide comprehensive tutorials that highlight a wider range of IDE features and programming techniques for a variety of application types. If you do not want to do a "Hello World" application, you can skip this tutorial and jump straight to the learning trails.

**Contents**



* [Setting Up the Project](https://netbeans.org/kb/docs/java/quickstart.html#setup)
* [Adding Code to the Generated Source File](https://netbeans.org/kb/docs/java/quickstart.html#code)
* [Compiling and Running the Application](https://netbeans.org/kb/docs/java/quickstart.html#run)
* [Building and Deploying the Application](https://netbeans.org/kb/docs/java/quickstart.html#build)
* [Next Steps](https://netbeans.org/kb/docs/java/quickstart.html#nextsteps)

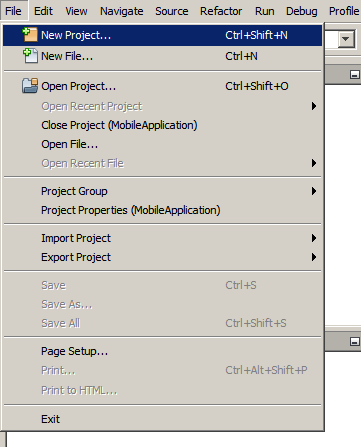
**To complete this tutorial, you need the following software and resources.**

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE](https://netbeans.org/downloads/index.html) | version 7.2, 7.3, 7.4, or 8.0, or higher |
| [Java Development Kit (JDK)](http://java.sun.com/javase/downloads/index.jsp) | version 6, 7, or 8 |

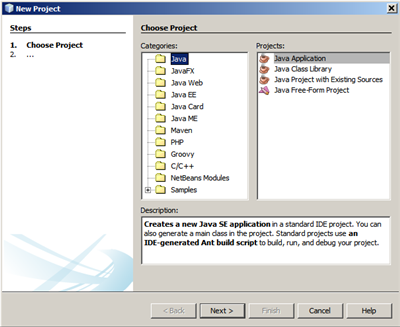
**Setting Up the Project**

To create an IDE project:

1. Start NetBeans IDE.
2. In the IDE, choose File > New Project, as shown in the figure below.

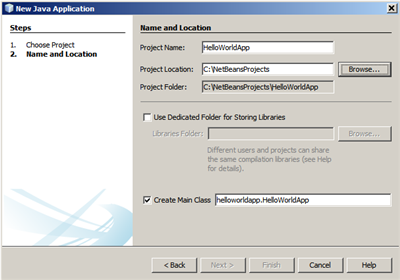


1. In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.

[](https://netbeans.org/images_www/articles/72/java/quickstart/proj-wizard.png)

1. In the Name and Location page of the wizard, do the following (as shown in the figure below):

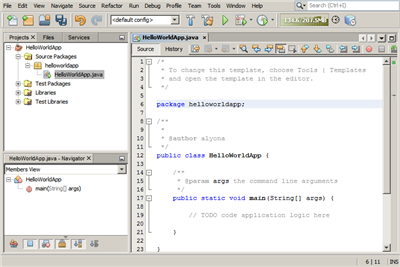
* + In the Project Name field, type HelloWorldApp.
  + Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
  + In the Create Main Class field, type helloworldapp.HelloWorldApp.

[](https://netbeans.org/images_www/articles/72/java/quickstart/proj-wizard2.png)

1. Click Finish.

The project is created and opened in the IDE. You should see the following components:

* The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
* The Source Editor window with a file called HelloWorldApp open.
* The Navigator window, which you can use to quickly navigate between elements within the selected class.

[](https://netbeans.org/images_www/articles/72/java/quickstart/proj-opened.png)

**Adding Code to the Generated Source File**

Because you have left the Create Main Class checkbox selected in the New Project wizard, the IDE has created a skeleton main class for you. You can add the "Hello World!" message to the skeleton code by replacing the line:

// TODO code application logic here

with the line:

System.out.println("Hello World!");

Save the change by choosing File > Save.

The file should look something like the following code sample.

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package helloworldapp;

/\*\*

\*

\* @author <your name>

\*/

public class HelloWorldApp {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

System.out.println("Hello World!");

}

}

**Compiling and Running the Program**

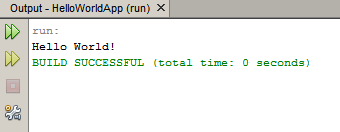
Because of the IDE's Compile on Save feature, you do not have to manually compile your project in order to run it in the IDE. When you save a Java source file, the IDE automatically compiles it.

The Compile on Save feature can be turned off in the Project Properties window. Right-click your project, select Properties. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

**To run the program:**

* Choose Run > Run Project.

The next figure shows what you should now see.



Congratulations! Your program works!

If there are compilation errors, they are marked with red glyphs in the left and right margins of the Source Editor. The glyphs in the left margin indicate errors for the corresponding lines. The glyphs in the right margin show all of the areas of the file that have errors, including errors in lines that are not visible. You can mouse over an error mark to get a description of the error. You can click a glyph in the right margin to jump to the line with the error.

**Building and Deploying the Application**

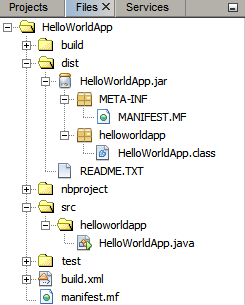
Once you have written and test run your application, you can use the Clean and Build command to build your application for deployment. When you use the Clean and Build command, the IDE runs a build script that performs the following tasks:

* Deletes any previously compiled files and other build outputs.
* Recompiles the application and builds a JAR file containing the compiled files.

**To build your application:**

* Choose Run > Clean and Build Project.

You can view the build outputs by opening the Files window and expanding the HelloWorldApp node. The compiled bytecode file HelloWorldApp.class is within the build/classes/helloworldapp subnode. A deployable JAR file that contains the HelloWorldApp.class is within the dist node.



You now know how to accomplish some of the most common programming tasks in the IDE.

For information on how to run the application from the command line, see the [Starting Your Java Application](https://netbeans.org/kb/docs/java/javase-deploy.html#startapp) chapter of the Packaging and Distributing Java Applications tutorial.

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20NetBeans%20IDE%20Java%20Quick%20Start%20Tutorial)

**See Also**

For information on creating and working with standard and free-form Java projects, see [Creating Java Projects](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG366) in *Developing Applications with NetBeans IDE*.

To learn more about the IDE workflow for developing Java applications, including classpath management, see [Developing General Java Applications](https://netbeans.org/kb/docs/java/javase-intro.html).

For detailed instructions on how to compile and run a simple "Hello World!" application on your operating system, see the [The "Hello World" Application](http://docs.oracle.com/javase/tutorial/getStarted/cupojava/index.html) lesson of the Java Tutorials.

To find information specific to the kind of applications you are developing, use the NetBeans IDE learning trail for that type of application. Each learning trail contains a series of tutorials and guides that range in scope from basic to advanced. The following learning trails are available:

|  |  |
| --- | --- |
| * [General Java Development](https://netbeans.org/kb/trails/java-se.html) * [Integration with External Tools and Services](https://netbeans.org/kb/trails/tools.html) * [Java GUI Applications](https://netbeans.org/kb/trails/matisse.html) * [Web Services Applications](https://netbeans.org/kb/trails/web.html) * [Java EE & Java Web Applications](https://netbeans.org/kb/trails/java-ee.html) | * [PHP and HTML5 Applications](https://netbeans.org/kb/trails/php.html) * [NetBeans Platform and Module Development](https://netbeans.org/kb/trails/platform.html) * [C/C++ Applications](https://netbeans.org/kb/trails/cnd.html) * [Mobile Applications](https://netbeans.org/kb/trails/mobility.html) |

# Introduction to GUI Building

Contributed by Saleem Gul and Tomas Pavek

This beginner tutorial teaches you how to create a simple graphical user interface and add simple back-end functionality. In particular we will show how to code the behavior of buttons and fields in a Swing form.

We will work through the layout and design of a GUI and add a few buttons and text fields. The text fields will be used for receiving user input and also for displaying the program output. The button will initiate the functionality built into the front end. The application we create will be a simple but functional calculator.

For a more comprehensive guide to the GUI Builder's design features, including video demonstrations of the various design features, see [Designing a Swing GUI in NetBeans IDE](https://netbeans.org/kb/docs/java/quickstart-gui.html).

**Expected duration: 20 minutes**

### Contents



* [Exercise 1: Creating a Project](https://netbeans.org/kb/docs/java/gui-functionality.html#Exercise_1)
* [Exercise 2: Building the Front End](https://netbeans.org/kb/docs/java/gui-functionality.html#Exercise_2)
* [Exercise 3: Adding Functionality](https://netbeans.org/kb/docs/java/gui-functionality.html#Exercise_3)
* [Exercise 4: Running the Program](https://netbeans.org/kb/docs/java/gui-functionality.html#Exercise_4)
* [How Event Handling Works](https://netbeans.org/kb/docs/java/gui-functionality.html#how)
* [See Also](https://netbeans.org/kb/docs/java/gui-functionality.html#seealso)

**To complete this tutorial, you need the following software and resources.**

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE with Java SE](http://netbeans.org/downloads/) | version 6.9 or higher |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | version 6, 7 or 8 |

## 

## Exercise 1: Creating a Project

The first step is to create an IDE project for the application that we are going to develop. We will name our project NumberAddition.

1. Choose File > New Project. Alternatively, you can click the New Project icon in the IDE toolbar.
2. In the Categories pane, select the Java node. In the Projects pane, choose Java Application. Click Next.
3. Type NumberAddition in the Project Name field and specify a path, for example, in your home directory, as the project location.
4. (Optional) Select the Use Dedicated Folder for Storing Libraries checkbox and specify the location for the libraries folder. See [Sharing a Library with Other Users](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG455) in *Developing Applications with NetBeans IDE* for more information.
5. Deselect the Create Main Class checkbox if it is selected.
6. Click Finish.

## Exercise 2: Building the Front End

To proceed with building our interface, we need to create a Java container within which we will place the other required GUI components. In this step we'll create a container using the JFrame component. We will place the container in a new package, which will appear within the Source Packages node.

### Create a JFrame container

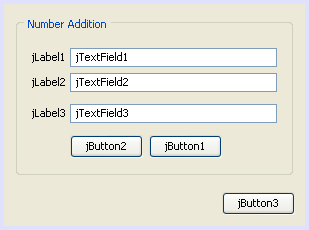
1. In the Projects window, right-click the NumberAddition node and choose New > Other.
2. In the New File dialog box, choose the Swing GUI Forms category and the JFrame Form file type. Click Next.
3. Enter NumberAdditionUI as the class name.
4. Enter my.numberaddition as the package.
5. Click Finish.

The IDE creates the NumberAdditionUI form and the NumberAdditionUI class within the NumberAddition application, and opens the NumberAdditionUI form in the GUI Builder. The my.NumberAddition package replaces the default package.

### Adding Components: Making the Front End

Next we will use the Palette to populate our application's front end with a JPanel. Then we will add three JLabels, three JTextFields, and three JButtons. If you have not used the GUI Builder before, you might find information in the [Designing a Swing GUI in NetBeans IDE](https://netbeans.org/kb/docs/java/quickstart-gui.html) tutorial on positioning components useful.

Once you are done dragging and positioning the aforementioned components, the JFrame should look something like the following screenshot.



If you do not see the Palette window in the upper right corner of the IDE, choose Window > Palette.

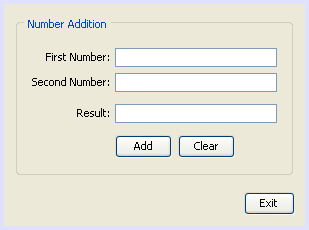
1. Start by selecting a Panel from the Swing Containers category on Palette and drop it onto the JFrame.
2. While the JPanel is highlighted, go to the Properties window and click the ellipsis (...) button next to Border to choose a border style.
3. In the Border dialog, select TitledBorder from the list, and type in Number Addition in the Title field. Click OK to save the changes and exit the dialog.
4. You should now see an empty titled JFrame that says Number Addition like in the screenshot. Look at the screenshot and add three JLabels, three JTextFields and three JButtons as you see above.

### Renaming the Components

In this step we are going to rename the display text of the components that were just added to the JFrame.

1. Double-click jLabel1 and change the text property to First Number:.
2. Double-click jLabel2 and change the text to Second Number:.
3. Double-click jLabel3 and change the text to Result:.
4. Delete the sample text from jTextField1. You can make the display text editable by right-clicking the text field and choosing Edit Text from the popup menu. You may have to resize the jTextField1 to its original size. Repeat this step for jTextField2 and jTextField3.
5. Rename the display text of jButton1 to Clear. (You can edit a button's text by right-clicking the button and choosing Edit Text. Or you can click the button, pause, and then click again.)
6. Rename the display text of jButton2 to Add.
7. Rename the display text of jButton3 to Exit.

Your Finished GUI should now look like the following screenshot:



## Exercise 3: Adding Functionality

In this exercise we are going to give functionality to the Add, Clear, and Exit buttons. The jTextField1 and jTextField2 boxes will be used for user input and jTextField3 for program output - what we are creating is a very simple calculator. Let's begin.

### Making the Exit Button Work

In order to give function to the buttons, we have to assign an event handler to each to respond to events. In our case we want to know when the button is pressed, either by mouse click or via keyboard. So we will use ActionListener responding to ActionEvent.

1. Right click the Exit button. From the pop-up menu choose Events > Action > actionPerformed. Note that the menu contains many more events you can respond to! When you select the actionPerformed event, the IDE will automatically add an ActionListener to the Exit button and generate a handler method for handling the listener's actionPerformed method.
2. The IDE will open up the Source Code window and scroll to where you implement the action you want the button to do when the button is pressed (either by mouse click or via keyboard). Your Source Code window should contain the following lines:
3. private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
4. //TODO add your handling code here:

}

1. We are now going to add code for what we want the Exit Button to do. Replace the TODO line with System.exit(0);. Your finished Exit button code should look like this:
2. private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
3. System.exit(0);

}

### Making the Clear Button Work

1. Click the Design tab at the top of your work area to go back to the Form Design.
2. Right click the Clear button (jButton1). From the pop-up menu select Events > Action > actionPerformed.
3. We are going to have the Clear button erase all text from the jTextFields. To do this, you will add some code like above. Your finished source code should look like this:
4. private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){
5. jTextField1.setText("");
6. jTextField2.setText("");
7. jTextField3.setText("");

}

The above code changes the text in all three of our JTextFields to nothing, in essence it is overwriting the existing Text with a blank.

### Making the Add Button Work

The Add button will perform three actions.

1. It is going to accept user input from jTextField1 and jTextField2 and convert the input from a type String to a float.
2. It will then perform addition of the two numbers.
3. And finally, it will convert the sum to a type String and place it in jTextField3.

Lets get started!

1. Click the Design tab at the top of your work area to go back to the Form Design.
2. Right-click the Add button (jButton2). From the pop-up menu, select Events > Action > actionPerformed.
3. We are going to add some code to have our Add button work. The finished source code shall look like this:
4. private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){
5. // First we define float variables.
6. float num1, num2, result;
7. // We have to parse the text to a type float.
8. num1 = Float.parseFloat(jTextField1.getText());
9. num2 = Float.parseFloat(jTextField2.getText());
10. // Now we can perform the addition.
11. result = num1+num2;
12. // We will now pass the value of result to jTextField3.
13. // At the same time, we are going to
14. // change the value of result from a float to a string.
15. jTextField3.setText(String.valueOf(result));

}

Our program is now complete we can now build and run it to see it in action.

## Exercise 4: Running the Program

**To run the program in the IDE:**

1. Choose Run > Run Project (Number Addition) (alternatively, press F6).

**Note:** If you get a window informing you that Project NumberAddition does not have a main class set, then you should select my.NumberAddition.NumberAdditionUI as the main class in the same window and click the OK button.

**To run the program outside of the IDE:**

1. Choose Run > Clean and Build Main Project (Shift-F11) to build the application JAR file.
2. Using your system's file explorer or file manager, navigate to the NumberAddition/dist directory.

**Note:** The location of the NumberAddition project directory depends on the path you specified while creating the project in step 3 of the [Exercise 1: Creating a Project](https://netbeans.org/kb/docs/java/gui-functionality.html#Exercise_1) section.

1. Double-click the NumberAddition.jar file.

After a few seconds, the application should start.

**Note:** If double-clicking the JAR file does not launch the application, see [this article](https://netbeans.org/kb/articles/javase-deploy.html#troubleshooting) for information on setting JAR file associations in your operating system.

You can also launch the application from the command line.

**To launch the application from the command line:**

1. On your system, open up a command prompt or terminal window.
2. In the command prompt, change directories to the NumberAddition/dist directory.
3. At the command line, type the following statement:

java -jar NumberAddition.jar

**Note:** Make sure my.NumberAddition.NumberAdditionUI is set as the main class before running the application. You can check this by right-clicking the NumberAddition project node in the Projects pane, choosing Properties in the popup menu, and selecting the Run category in the Project Properties dialog box. The Main Class field should display my.numberaddition.NumberAdditionUI.

## How Event Handling Works

This tutorial has showed how to respond to a simple button event. There are many more events you can have your application respond to. The IDE can help you find the list of available events your GUI components can handle:

1. Go back to the file NumberAdditionUI.java in the Editor. Click the Design tab to see the GUI's layout in the GUI Builder.
2. Right-click any GUI component, and select Events from the pop-up menu. For now, just browse the menu to see what's there, you don't need to select anything.
3. Alternatively, you can select Properties from the Window menu. In the Properties window, click the Events tab. In the Events tab, you can view and edit events handlers associated with the currently active GUI component.
4. You can have your application respond to key presses, single, double and triple mouse clicks, mouse motion, window size and focus changes. You can generate event handlers for all of them from the Events menu. The most common event you will use is an Action event. (Learn [best practices for Event handling](http://java.sun.com/docs/books/tutorial/uiswing/events/generalrules.html#twokinds) from Sun's [Java Events Tutorial](http://java.sun.com/docs/books/tutorial/uiswing/events/index.html).)

How does event handling work? Every time you select an event from the Event menu, the IDE automatically creates a so-called event listener for you, and hooks it up to your component. Go through the following steps to see how event handling works.

1. Go back to the file NumberAdditionUI.java in the Editor. Click the Source tab to see the GUI's source.
2. Scroll down and note the methods jButton1ActionPerformed(), jButton2ActionPerformed(), and jButton3ActionPerformed() that you just implemented. These methods are called event handlers.
3. Now scroll to a method called initComponents(). If you do not see this method, look for a line that says Generated Code; click the + sign next to it to expand the collapsed initComponents() method.
4. First, note the blue block around the initComponents() method. This code was auto-generated by the IDE and you cannot edit it.
5. Now, browse through the initComponents() method. Among other things, it contains the code that initializes and places your GUI components on the form. This code is generated and updated automatically while you place and edit components in the Design view.
6. In initComponents(), scroll down to where it reads

jButton3.setText("Exit");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

This is the spot where an event listener object is added to the GUI component; in this case, you register an ActionListener to the jButton3. The ActionListener interface has an actionPerformed method taking ActionEvent object which is implemented simply by calling your jButton3ActionPerformed event handler. The button is now listening to action events. Everytime it is pressed an ActionEvent is generated and passed to the listener's actionPerformed method which in turn executes code that you provided in the event handler for this event.

Generally speaking, to be able to respond, each interactive GUI component needs to register to an event listener and needs to implement an event handler. As you can see, NetBeans IDE handles hooking up the event listener for you, so you can concentrate on implementing the actual business logic that should be triggered by the event.

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20Introduction%20to%20GUI%20Building)

## See Also

* [Implementing Java GUIs](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG920) in *Developing Applications with NetBeans IDE*
* [Designing a Swing GUI in NetBeans IDE](https://netbeans.org/kb/docs/java/quickstart-gui.html)
* [Handling Images in A GUI Application](https://netbeans.org/kb/docs/java/gui-image-display.html)
* [Using Hibernate in a Java Swing Application](https://netbeans.org/kb/docs/java/hibernate-java-se.html)
* [Java GUI Applications Learning Trail](https://netbeans.org/kb/trails/matisse.html)
* [General Java Development Learning Trail](https://netbeans.org/kb/trails/java-se.html)
* [GUI Builder - Frequently Asked Questions](http://wiki.netbeans.org/wiki/view/NetBeansUserFAQ#section-NetBeansUserFAQ-GUIEditorMatisse)
* [Lesson: Writing Event Listeners](http://download.oracle.com/javase/tutorial/uiswing/events/index.html) from [The Java™ Tutorials](http://download.oracle.com/javase/tutorial/index.html).

# Designing a Swing GUI in NetBeans IDE

This tutorial guides you through the process of creating the graphical user interface (GUI) for an application called ContactEditor using the NetBeans IDE GUI Builder. In the process you will layout a GUI front-end that enables you to view and edit contact information of individuals included in an employee database.

In this tutorial you will learn how to: use the GUI Builder Interface, create a GUI Container, add, resize, and align components, adjust component anchoring, set component auto-resizing behavior, edit component properties.

This tutorial takes about 30 minutes to complete.

### Contents

* [Getting Started](https://netbeans.org/kb/docs/java/quickstart-gui.html#getting_started)
  + [Creating a Project](https://netbeans.org/kb/docs/java/quickstart-gui.html#project)
  + [Creating a JFrame Container](https://netbeans.org/kb/docs/java/quickstart-gui.html#container)
* [Getting Familiar with the GUI Builder](https://netbeans.org/kb/docs/java/quickstart-gui.html#getting_familiar)
* [Key Concepts](https://netbeans.org/kb/docs/java/quickstart-gui.html#key_concepts)
  + [Free Design](https://netbeans.org/kb/docs/java/quickstart-gui.html#design)
  + [Automatic Component Positioning (Snapping)](https://netbeans.org/kb/docs/java/quickstart-gui.html#snapping)
  + [Visual Feedback](https://netbeans.org/kb/docs/java/quickstart-gui.html#feedback)
* [First Things First](https://netbeans.org/kb/docs/java/quickstart-gui.html#first_things)
  + [Adding Components: The Basics](https://netbeans.org/kb/docs/java/quickstart-gui.html#adding_components)
  + [Adding Individual Components to the Form](https://netbeans.org/kb/docs/java/quickstart-gui.html#individual)
  + [Adding Multiple Components to the Form](https://netbeans.org/kb/docs/java/quickstart-gui.html#multiple)
  + [Inserting Components](https://netbeans.org/kb/docs/java/quickstart-gui.html#inserting)
* [Moving Forward](https://netbeans.org/kb/docs/java/quickstart-gui.html#moving_forward)
  + [Component Alignment](https://netbeans.org/kb/docs/java/quickstart-gui.html#aligning_components)
  + [Baseline Alignment](https://netbeans.org/kb/docs/java/quickstart-gui.html#baseline_alignment)
* [Reviewing What We've Learned](https://netbeans.org/kb/docs/java/quickstart-gui.html#reviewing)
  + [Adding, Aligning, and Anchoring](https://netbeans.org/kb/docs/java/quickstart-gui.html#adding_aligning_anchoring)
  + [Component Sizing](https://netbeans.org/kb/docs/java/quickstart-gui.html#component_sizing)
  + [Indentation](https://netbeans.org/kb/docs/java/quickstart-gui.html#indentation)
* [Making the Final Adjustments](https://netbeans.org/kb/docs/java/quickstart-gui.html#adjusting_form)
* [Previewing Your GUI](https://netbeans.org/kb/docs/java/quickstart-gui.html#previewing_form)
* [Deploying GUI Applications](https://netbeans.org/kb/docs/java/quickstart-gui.html#deploying)
* [See Also](https://netbeans.org/kb/docs/java/quickstart-gui.html#seealso)

**To complete this tutorial, you need the following software and resources.**

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE](http://netbeans.org/downloads/index.html) | version 6.9 and more recent |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | version 6, 7 or 8 |

## Getting Started

The IDE's GUI Builder makes it possible to build professional-looking GUIs without an intimate understanding of layout managers. You can lay out your forms by simply placing components where you want them.

For descriptions of the GUI Builder's visual feedback, you can use the [GUI Builder Visual Feedback Legend](https://netbeans.org/kb/docs/java/quickstart-gui-legend.html).

### Creating a Project

Because all Java development in the IDE takes place within projects, we first need to create a new ContactEditor project within which to store sources and other project files. An IDE project is a group of Java source files plus its associated meta data, including project-specific properties files, an Ant build script that controls the build and run settings, and a project.xml file that maps Ant targets to IDE commands. While Java applications often consist of several IDE projects, for the purposes of this tutorial, we will build a simple application which is stored entirely in a single project.

|  |
| --- |
| To create a new ContactEditor application project:   1. Choose File > New Project. Alternately, you can click the New Project icon in the IDE toolbar. 2. In the Categories pane, select the Java node and in the Projects pane, choose Java Application. Click Next. 3. Enter ContactEditor in the Project Name field and specify the project location. 4. Leave the Use Dedicated Folder for Storing Libraries checkbox unselected. 5. Ensure that the Set as Main Project checkbox is selected and clear the Create Main Class field. 6. Click Finish.   The IDE creates the ContactEditor folder on your system in the designated location. This folder contains all of the project's associated files, including its Ant script, folders for storing sources and tests, and a folder for project-specific metadata. To view the project structure, use the IDE's Files window. |

### Creating a JFrame Container

After creating the new application, you may have noticed that the Source Packages folder in the Projects window contains an empty <default package> node. To proceed with building our interface, we need to create a Java container within which we will place the other required GUI components. In this step we'll create a container using the JFrame component and place the container in a new package.

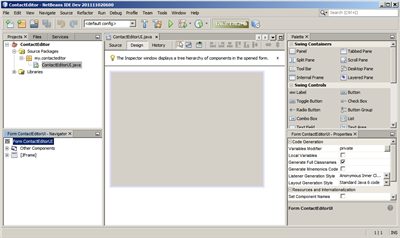
|  |
| --- |
| To add a JFrame container:   1. In the Projects window, right-click the ContactEditor node and choose New > JFrame Form.  Alternatively, you can find a JFrame form by choosing New > Other > Swing GUI Forms > JFrame Form. 2. Enter ContactEditorUI as the Class Name. 3. Enter my.contacteditor as the package. 4. Click Finish.   The IDE creates the ContactEditorUI form and the ContactEditorUI class within the ContactEditorUI.java application and opens the ContactEditorUI form in the GUI Builder. Notice that the my.contacteditor package replaces the default package. |

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Getting Familiar with the GUI Builder

Now that we've set up a new project for our application, let's take a minute to familiarize ourselves with the GUI Builder's interface.

**Note:** To explore the GUI Builder interface with an interactive demo, view the [Exploring GUI Builder (.swf)](http://bits.netbeans.org/media/quickstart-gui-explore.swf) screencast.

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/01_gb_ui.png)

When we added the JFrame container, the IDE opened the newly-created ContactEditorUI form in an Editor tab with a toolbar containing several buttons, as shown in the preceding illustration. The ContactEditor form opened in the GUI Builder's Design view and three additional windows appeared automatically along the IDE's edges, enabling you to navigate, organize, and edit GUI forms as you build them.

The GUI Builder's various windows include:

* **Design Area.** The GUI Builder's primary window for creating and editing Java GUI forms. The toolbar's Source button enables you to view a class's source code, the Design button allows you to view a graphical view of the GUI components, the History button allows you to access the local history of changes of the file. The additional toolbar buttons provide convenient access to common commands, such as choosing between Selection and Connection modes, aligning components, setting component auto-resizing behavior, and previewing forms.
* **Navigator.** Provides a representation of all the components, both visual and non-visual, in your application as a tree hierarchy. The Navigator also provides visual feedback about what component in the tree is currently being edited in the GUI Builder as well as allows you to organize components in the available panels.
* **Palette.** A customizable list of available components containing tabs for JFC/Swing, AWT, and JavaBeans components, as well as layout managers. In addition, you can create, remove, and rearrange the categories displayed in the Palette using the customizer.
* **Properties Window.** Displays the properties of the component currently selected in the GUI Builder, Navigator window, Projects window, or Files window.

If you click the Source button, the IDE displays the application's Java source code in the Editor with sections of code that are automatically generated by the GUI Builder indicated by grey areas (they become blue when selected), called Guarded Blocks. Guarded blocks are protected areas that are not editable in Source view. You can only edit code appearing in the white areas of the Editor when in Source view. If you need to make changes to the code within a Guarded Block, clicking the Design button returns the IDE's Editor to the GUI Builder where you can make the necessary adjustments to the form. When you save your changes, the IDE updates the file's sources.

**Note:** For advanced developers, the Palette Manager is available that enables you to add custom components from JARs, libraries, or other projects to the Palette. To add custom components through the Palette Manager, choose Tools > Palette > Swing/AWT Components.

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Key Concepts

The IDE's GUI Builder solves the core problem of Java GUI creation by streamlining the workflow of creating graphical interfaces, freeing developers from the complexities of Swing layout managers. It does this by extending the current NetBeans IDE GUI Builder to support a straightforward "Free Design" paradigm with simple layout rules that are easy to understand and use. As you lay out your form, the GUI Builder provides visual guidelines suggesting optimal spacing and alignment of components. In the background, the GUI Builder translates your design decisions into a functional UI that is implemented using the new GroupLayout layout manager and other Swing constructs. Because it uses a dynamic layout model, GUI's built with the GUI Builder behave as you would expect at runtime, adjusting to accommodate any changes you make without altering the defined relationships between components. Whenever you resize the form, switch locales, or specify a different look and feel, your GUI automatically adjusts to respect the target look and feel's insets and offsets.

### Free Design

In the IDE's GUI Builder, you can build your forms by simply putting components where you want them as though you were using absolute positioning. The GUI Builder figures out which layout attributes are required and then generates the code for you automatically. You need not concern yourself with insets, anchors, fills, and so forth.

### Automatic Component Positioning (Snapping)

As you add components to a form, the GUI Builder provides visual feedback that assists in positioning components based on your operating system's look and feel. The GUI Builder provides helpful inline hints and other visual feedback regarding where components should be placed on your form, automatically snapping components into position along guidelines. It makes these suggestions based on the positions of the components that have already been placed in the form, while allowing the padding to remain flexible such that different target look and feels render properly at runtime.

### Visual Feedback

The GUI Builder also provides visual feedback regarding component anchoring and chaining relationships. These indicators enable you to quickly identify the various positioning relationships and component pinning behavior that affect the way your GUI will both appear and behave at runtime. This speeds the GUI design process, enabling you to quickly create professional-looking visual interfaces that work.

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## First Things First

Now that you have familiarized yourself with the GUI builder's interface, it's time to begin developing the UI of our ContactEditor application. In this section we'll take a look at using the IDE's Palette to add the various GUI components that we need to our form.

Thanks to the IDE's Free Design paradigm, you no longer have to struggle with layout managers to control the size and position of the components within your containers. All you need to do is drag and drop the components you need to your GUI form as shown in the illustrations that follow.

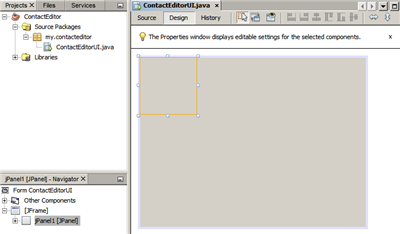
**Note:** Refer to the [Adding individual and multiple components (.swf)](http://bits.netbeans.org/media/quickstart-gui-add.swf) screencast for an interactive demo on the section below.

### Adding Components: The Basics

Though the IDE's GUI Builder simplifies the process of creating Java GUIs, it is often helpful to sketch out the way you want your interface to look before beginning to lay it out. Many interface designers consider this a "best practice" technique, however, for the purposes of this tutorial you can simply peek at how our completed form should look by jumping ahead to the [Previewing your GUI](https://netbeans.org/kb/docs/java/quickstart-gui.html#previewing_form) section.

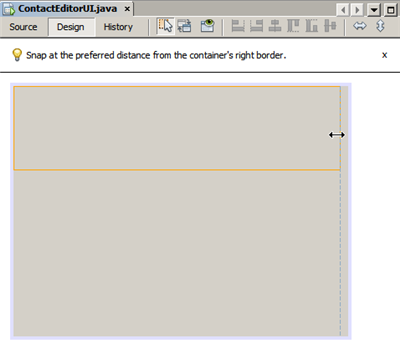
Since we've already added a JFrame as our form's top-level container, the next step is to add a couple of JPanels which will enable us to cluster the components of our UI using titled borders. Refer to the following illustrations and notice the IDE's "drag and drop" behavior when accomplishing this.

|  |
| --- |
| To add a JPanel:   1. In the Palette window, select the Panel component from the Swing Containers category by clicking and releasing the mouse button. 2. Move the cursor to the upper left corner of the form in the GUI Builder. When the component is located near the container's top and left edges, horizontal and vertical alignment guidelines appear indicating the preferred margins. Click in the form to place the JPanel in this location.   The JPanel component appears in the ContactEditorUI form with orange highlighting signifying that it is selected. After releasing the mouse button, small indicators appear to show the component's anchoring relationships and a corresponding JPanel node is displayed in the Navigator window, as shown in the following illustration. |

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/02_add_panels_1.png)

Next, we need to resize the JPanel to make room for the components we'll place within it a little later, but let's take a minute to point out another of the GUI Builder's visualization features first. In order to do this we need to deselect the JPanel we just added. Because we haven't added a title border yet, the panel disappears. Notice, however, that when you pass the cursor over the JPanel, its edges change to light gray so that its position can be clearly seen. You need only to click anywhere within the component to reselect it and cause the resize handles and anchoring indicators to reappear.

|  |
| --- |
| To resize the JPanel:   1. Select the JPanel you just added. The small square resize handles reappear around the component's perimeter. 2. Click and hold the resize handle on the right edge of the JPanel and drag until the dotted alignment guideline appears near the form's edge. 3. Release the resize handle to resize the component.   The JPanel component is extended to span between the container's left and right margins in accordance with the recommended offset, as shown in the following illustration. |

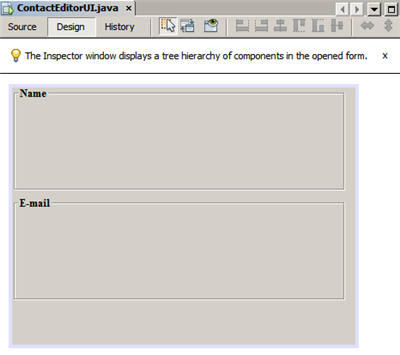
[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/02_add_panels_2.png)

Now that we've added a panel to contain our UI's Name information, we need to repeat the process to add another directly below the first for the E-mail information. Referring to the following illustrations, repeat the previous two tasks, paying attention to the GUI Builder's suggested positioning. Notice that the suggested vertical spacing between the two JPanels is much narrower than that at the edges. Once you have added the second JPanel, resize it such that it fills the form's remaining vertical space.

|  |  |
| --- | --- |
| [Another JPanel added](https://netbeans.org/images_www/articles/80/java/quickstart-gui/02_add_panels_3.png) | [Another JPanel selected](https://netbeans.org/images_www/articles/80/java/quickstart-gui/02_add_panels_4.png) |
| [Another JPanel resized](https://netbeans.org/images_www/articles/80/java/quickstart-gui/02_add_panels_5.png) |  |

Because we want to visually distinguish the functions in the upper and lower sections of our GUI, we need to add a border and title to each JPanel. First we'll accomplish this using the Properties window and then we'll try it using the pop-up menu.

|  |
| --- |
| To add title borders to the JPanels:   1. Select the top JPanel in the GUI Builder. 2. In the Properties window, click the ellipsis button (...) next to the Border property. 3. In the JPanel Border editor that appears, select the TitledBorder node in the Available Borders pane. 4. In the Properties pane below, enter Name for the Title property. 5. Click the ellipsis (...) next to the Font property, select Bold for the Font Style, and enter 12 for the Size. Click OK to exit the dialogs. 6. Select the bottom JPanel and repeat steps 2 through 5, but this time right-click the JPanel and access the Properties window using the pop-up menu. Enter E-mail for the Title property.   Titled borders are added to both JPanel components. |

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/02_add_borders.png)

### Adding Individual Components to the Form

Now we need to start adding the components that will present the actual contact information in our contact list. In this task we'll add four JTextFields that will display the contact information and the JLabels that will describe them. While accomplishing this, notice the horizontal and vertical guidelines that the GUI Builder displays, suggesting the preferred component spacing as defined by your operating system's look and feel. This ensures that your GUI is automatically rendered respecting the target operating system's look and feel at runtime.

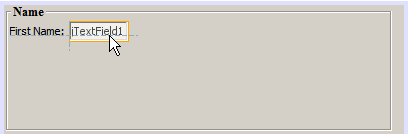
|  |
| --- |
| To add a JLabel to the form:   1. In the Palette window, select the Label component from the Swing Controls category. 2. Move the cursor over the Name JPanel we added earlier. When the guidelines appear indicating that the JLabel is positioned in the top left corner of the JPanel with a small margin at the top and left edges, click to place the label.   The JLabel is added to the form and a corresponding node representing the component is added to the Inspector window. |

Before going further, we need to edit the display text of the JLabel we just added. Though you can edit component display text at any point, the easiest way is to do this as you add them.

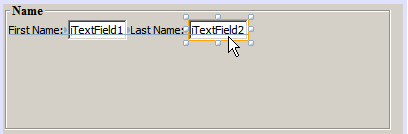
|  |
| --- |
| To edit the display text of a JLabel:   1. Double-click the JLabel to select its display text. 2. Type First Name: and press Enter.   The JLabel's new name is displayed and the component's width adjusts as a result of the edit. |

Now we'll add a JTextField so we can get a glimpse of the GUI Builder's baseline alignment feature.

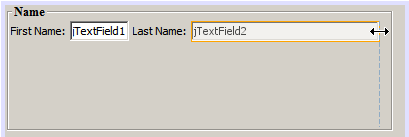
|  |
| --- |
| To add a JTextField to the form:   1. In the Palette window, select the Text Field component from the Swing Controls category. 2. Move the cursor immediately to the right of the First Name: JLabel we just added. When the horizontal guideline appears indicating that the JTextField's baseline is aligned with that of the JLabel and the spacing between the two components is suggested with a vertical guideline, click to position the JTextField.   The JTextField snaps into position in the form aligned with the JLabel's baseline, as shown in the following illustration. Notice that the JLabel shifted downward slightly in order to align with the taller text field's baseline. As usual, a node representing the component is added to the Navigator window. |



Before proceeding further, we need to add an additional JLabel and JTextField immediately to the right of those we just added, as shown in the following illustration. This time enter Last Name: as the JLabel's display text and leave the JTextFields' placeholder text as it is for now.



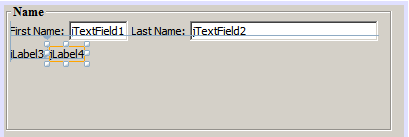
|  |
| --- |
| To resize a JTextField:   1. Select the JTextField we just added to the right of the Last Name: JLabel. 2. Drag the JTextField's right edge resize handle toward the right edge of the enclosing JPanel. 3. When the vertical alignment guidelines appear suggesting the margin between the text field and right edge of the JPanel, release the mouse button to resize the JTextField.   The JTextField's right edge snaps into alignment with the JPanel's recommended edge margin, as shown in the following illustration. |



### Adding Multiple Components to the Form

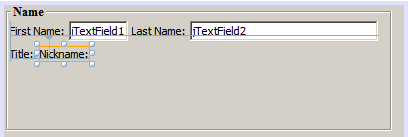
Now we'll add the Title: and Nickname: JLabels that describe two JTextFields that we're going to add in a minute. We'll drag and drop the components while pressing the Shift key, to quickly add them to the form. While accomplishing this, again notice that the GUI Builder displays horizontal and vertical guidelines suggesting the preferred component spacing.

|  |
| --- |
| To add multiple JLabels to the form:   1. In the Palette window, select the Label component from the Swing Controls category by clicking and releasing the mouse button. 2. Move the cursor over the form directly below the First Name: JLabel we added earlier. When the guidelines appear indicating that the new JLabel's left edge is aligned with that of the JLabel above and a small margin exists between them, shift-click to place the first JLabel. 3. While still pressing the Shift key, place another JLabel immediately to the right of the first. Make certain to release the Shift key prior to positioning the second JLabel. If you forget to release the Shift key prior to positioning the last JLabel, simply press the Escape key.   The JLabels are added to the form creating a second row, as shown in the following illustration. Nodes representing each component are added to the Navigator window. |



Before moving on, we need to edit the JLabels' name so that we'll be able to see the effect of the alignments we'll set later.

|  |
| --- |
| To edit the display text of JLabels:   1. Double-click the first JLabel to select its display text. 2. Type Title: and press Enter. 3. Repeat steps 1 and 2, entering Nickname: for the second JLabel's name property.   The JLabels' new names are displayed in the form and are shifted as a result of their edited widths, as shown in the following illustration. |



### Inserting Components

**Note:** Refer to the [Inserting components (.swf)](http://bits.netbeans.org/media/quickstart-gui-insert.swf) screencast for an interactive demo on the section below.

Often it is necessary to add a component between components that are already placed in a form. Whenever you add a component between two existing components, the GUI Builder automatically shifts them to make room for the new component. To demonstrate this, we'll insert a JTextField between the JLabels we added previously, as shown in the following two illustrations.

|  |
| --- |
| To insert a JTextField between two JLabels:   1. In the Palette window, select the Text Field component from the Swing Controls category. 2. Move the cursor over the Title: and Nickname: JLabels on the second row such that the JTextField overlaps both and is aligned to their baselines. If you encounter difficulty positioning the new text field, you can snap it to the left guideline of the Nickname JLabel as shown in the first image below. 3. Click to place the JTextField between the Title: and Nickname: JLabels.   The JTextField snaps into position between the two JLabels. The rightmost JLabel shifts toward the right of the JTextField to accommodate the suggested horizontal offset. |

|  |  |
| --- | --- |
| JTextField overlaps Nickname: JLabel | JTextField between Tiltle: and Nickname: JLabels |

We still need to add one additional JTextField to the form that will display each contact's nickname on the right side of the form.

|  |
| --- |
| To add a JTextField:   1. In the Palette window, select the Text Field component from the Swing category. 2. Move the cursor to the right of the Nickname label and click to place the text field.   The JTextField snaps into position next to the JLabel on its left. |

|  |
| --- |
| To resize a JTextField:   1. Drag the resize handles of the Nickname: label's JTextField you added in the previous task toward the right of the enclosing JPanel. 2. When the vertical alignment guidelines appear suggesting the margin between the text field and JPanel edges, release the mouse button to resize the JTextField.   The JTextField's right edge snaps into alignment with the JPanel's recommended edge margin and the GUI Builder infers the appropriate resizing behavior.   1. Press Ctrl-S to save the file. |

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Moving Forward

Alignment is one of the most fundamental aspects of creating professional-looking GUIs. In the previous section we got a glimpse of the IDE's alignment features while adding the JLabel and JTextField components to our ContactEditorUI form. Next, we'll take a more in depth look at the GUI Builder's alignment features as we work with the various other components we need for our application.

### Component Alignment

**Note:** Refer to the [Aligning and anchoring components (.swf)](http://bits.netbeans.org/media/quickstart-gui-align.swf) screencast for an interactive demo on the sections below.

Every time you add a component to a form, the GUI Builder effectively aligns them, as evidenced by the alignment guidelines that appear. It is sometimes necessary, however, to specify different relationships between groups of components as well. Earlier we added four JLabels that we need for our ContactEditor GUI, but we didn't align them. Now we'll align the two columns of JLabels so that their right edges line up.

|  |
| --- |
| To align components:   1. Hold down the Ctrl key and click to select the First Name: and Title: JLabels on the left side of the form. 2. Click the Align Right in Column button (https://netbeans.org/images_www/articles/80/java/quickstart-gui/align_r.png) in the toolbar. Alternately, you can right-click either one and choose Align > Right in Column from the pop-up menu. 3. Repeat this for the Last Name: and Nickname: JLabels as well.   The JLabels' positions shift such that the right edges of their display text are aligned. The anchoring relationships are updated, indicating that the components have been grouped. |

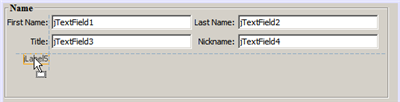
Before we're finished with the JTextFields we added earlier, we need to make sure that the two JTextFields we inserted between the JLabels are set to resize correctly. Unlike the two JTextFields that we stretched to the right edge of our form, inserted components' resizeability behavior isn't automatically set.

|  |
| --- |
| To set component resizeability behavior:   1. Control-click the two inserted JTextField components to select them in the GUI Builder. 2. With both JTextFields selected, right-click either one of them and choose Auto Resizing > Horizontal from the pop-up menu.   The JTextFields are set to resize horizontally at runtime. The alignment guidelines and anchoring indicators are updated, providing visual feedback of the component relationships. |

|  |
| --- |
| To set components to be the same size:   1. Control-click all four of the JTextFields in the form to select them. 2. With the JTextFields selected, right-click any one of them and choose Set Same Size > Same Width from the pop-up menu.   The JTextFields are all set to the same width and indicators are added to the top edge of each, providing visual feedback of the component relationships. |

Now we need to add another JLabel describing the JComboBox that will enable users to select the format of the information our ContactEditor application will display.

|  |
| --- |
| To align a JLabel to a component group:   1. In the Palette window, select the Label component from the Swing category. 2. Move the cursor below the First Name and Title JLabels on the left side of the JPanel. When the guideline appears indicating that the new JLabel's right edge is aligned with the right edges of the component group above (the two JLabels), click to position the component.   The JLabel snaps into a right-aligned position with the column of JLabels above, as shown in the following illustration. The GUI Builder updates the alignment status lines indicating the component's spacing and anchoring relationships. |

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/06_align_1.png)

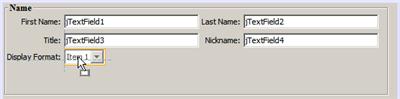
As in the previous examples, double-click the JLabel to select its display text and then enter Display Format: for the display name. Notice that when the JLabel snaps into position, the other components shift to accommodate the longer display text.

### Baseline Alignment

Whenever you add or move components that include text (JLabels, JTextFields, and so forth), the IDE suggests alignments which are based on the baselines of the text in the components. When we inserted the JTextField earlier, for example, its baseline was automatically aligned to the adjacent JLabels.

Now we'll add the combo box that will enable users to select the format of the information that our ContactEditor application will display. As we add the JComboBox, we'll align its baseline to that of the JLabel's text. Notice once again the baseline alignment guidelines that appear to assist us with the positioning.

|  |
| --- |
| To align the baselines of components:   1. In the Palette window, select the Combo Box component from the Swing Controls category. 2. Move the cursor immediately to the right of the JLabel we just added. When the horizontal guideline appears indicating that the JComboBox's baseline is aligned with the baseline of the text in the JLabel and the spacing between the two components is suggested with a vertical guideline, click to position the combo box.   The component snaps into a position aligned with the baseline of the text in the JLabel to its left, as shown in the following illustration. The GUI Builder displays status lines indicating the component's spacing and anchoring relationships. |

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/06_align_2.png)

|  |
| --- |
| To resize the JComboBox:   1. Select the ComboBox in the GUI Builder. 2. Drag the resize handle on the JComboBox's right edge toward the right until the alignment guidelines appear suggesting the preferred offset between the JComboBox and JPanel edges.   As shown in the following illustration, the JComboBox's right edge snaps into alignment with the JPanel's recommended edge margin and the component's width is automatically set to resize with the form.  [Combo Box resized](https://netbeans.org/images_www/articles/80/java/quickstart-gui/06_align_3.png)   1. Press Ctrl-S to save the file. |

Editing component models is beyond the scope of this tutorial, so for the time being we'll leave the JComboBox's placeholder item list as it is.

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Reviewing What We've Learned

We've got off to a good start building our ContactEditor GUI, but let's take a minute to recap what we've learned while we add a few more of the components our interface requires.

Until now we've concentrated on adding components to our ContactEditor GUI using the IDE's alignment guidelines to help us with positioning. It is important to understand, however, that another integral part of component placement is anchoring. Though we haven't discussed it yet, you've already taken advantage of this feature without realizing it. As mentioned previously, whenever you add a component to a form, the IDE suggests the target look and feel's preferred positioning with guidelines. Once placed, new components are also anchored to the nearest container edge or component to ensure that component relationships are maintained at runtime. In this section, we'll concentrate on accomplishing the tasks in a more streamlined fashion while pointing out the work the GUI builder is doing behind the scenes.

### Adding, Aligning, and Anchoring

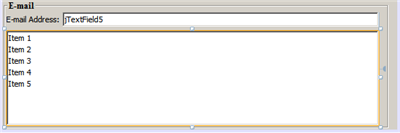
The GUI Builder enables you to lay out your forms quickly and easily by streamlining typical workflow gestures. Whenever you add a component to a form, the GUI Builder automatically snaps them into the preferred positions and sets the necessary chaining relationships so you can concentrate on designing your forms rather than struggling with complicated implementation details.

|  |
| --- |
| To add, align, and edit the display text of a JLabel:   1. In the Palette window, select the Label component from the Swing Controls category. 2. Move the cursor over the form immediately below the bottom JPanel's E-mail title. When the guidelines appear indicating that it's positioned in the top left corner of the JPanel with a small margin at the top and left edges, click to place the JLabel. 3. Double-click the JLabel to select its display text. Then type E-mail Address: and press Enter.   The JLabel snaps into the preferred position in the form, anchored to the top and left edges of the enclosing JPanel. Just as before, a corresponding node representing the component is added to the Navigator window. |

|  |
| --- |
| To add a JTextField:   1. In the Palette window, select the Text Field component from the Swing Controls category. 2. Move the cursor immediately to the right of the E-mail Address label we just added. When the guidelines appear indicating that the JTextField's baseline is aligned with the baseline of the text in the JLabel and the margin between the two components is suggested with a vertical guideline, click to position the text field.   The JTextField snaps into position on the right of the E-mail Address: JLabel and is chained to the JLabel. Its corresponding node is also added to the Inspector window.   1. Drag the resize handle of the JTextField toward the right of the enclosing JPanel until the alignment guidelines appear suggesting the offset between the JTextField and JPanel edges.   The JTextField's right edge snaps to the alignment guideline indicating the preferred margins. |

Now we need to add the JList that will display our ContactEditor's entire contact list.

|  |
| --- |
| To add and resize a JList:   1. In the Palette window, select the List component from the Swing Controls category. 2. Move the cursor immediately below the E-mail Address JLabel we added earlier. When the guidelines appear indicating that the JList's top and left edges are aligned with the preferred margins along the JPanel's left edge and the JLabel above, click to position the JList. 3. Drag the JList's right resize handle toward the right of the enclosing JPanel until the alignment guidelines appear indicating that it is the same width as the JTextField above.   The JList snaps into the position designated by the alignment guidelines and its corresponding node is displayed in the Inspector window. Notice also that the form expands to accommodate the newly added JList. |

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/06_align_4.png)

Since JLists are used to display long lists of data, they typically require the addition of a JScrollPane. Whenever you add a component which requires a JScrollPane, the GUI Builder automatically adds it for you. Because JScrollPanes are non-visual components, you have to use the Inspector window in order to view or edit any JScrollPanes that the GUI Builder created.

### Component Sizing

**Note:** Refer to the [Resizing and indenting components (.swf)](http://bits.netbeans.org/media/quickstart-gui-resize.swf) screencast for an interactive demo on the sections below.

It is often beneficial to set several related components, such as buttons in modal dialogues, to be the same size for visual consistency. To demonstrate this we'll add four JButtons to our ContactEditor form that will allow us to add, edit, and remove individual entries from our contact list, as shown in the following illustrations. Afterwards, we'll set the four buttons to be the same size so they can be easily recognized as offering related functionality.

|  |  |  |
| --- | --- | --- |
| To add, align, and edit the display text of multiple buttons:   1. In the Palette window, select the Button component from the Swing Controls category. 2. Move the JButton over the right edge of the E-mail Address JTextField in the lower JPanel. When the guidelines appear indicating that the JButton's baseline and right edge are aligned with that of the JTextField, shift-click to place the first button along the JFrame's right edge. The JTextField's width shrinks to accommodate the JButton when you release the mouse button.  |  |  | | --- | --- | | [JButton aligned](https://netbeans.org/images_www/articles/80/java/quickstart-gui/buttons_1.png) | [JButton added](https://netbeans.org/images_www/articles/80/java/quickstart-gui/buttons_2.png) |  1. Move the cursor over the top right corner of the JList in the lower JPanel. When the guidelines appear indicating that the JButton's top and right edges are aligned with that of the JList, shift-click to place the second button along the JFrame's right edge.   [Second JButton added](https://netbeans.org/images_www/articles/80/java/quickstart-gui/buttons_3.png)   1. Add two additional JButtons below the two we already added to create a column. Make certain to position the JButtons such that the suggested spacing is respected and consistent. If you forget to release the Shift key prior to positioning the last JButton, simply press the Escape key.   [Two additional JButtons added](https://netbeans.org/images_www/articles/80/java/quickstart-gui/buttons_4.png)   1. Set the display text for each JButton. (You can edit a button's text by right-clicking the button and choosing Edit Text. Or you can click the button, pause, and then click again.) Enter Add for the top button, Edit for the second, Remove for the third, and As Default for the fourth.   The JButton components snap into the positions designated by the alignment guidelines. The width of the buttons changes to accommodate the new names.  [JButtons renamed](https://netbeans.org/images_www/articles/80/java/quickstart-gui/buttons_5.png) |

Now that the buttons are positioned where we want them, we'll set the four buttons to be the same size for visual consistency as well as to clarify that they are related functionally.

|  |
| --- |
| To set components to the same size:   1. Select all four JButtons by pressing the Control key while making your selection. 2. Right-click one of them and choose Same Size > Same Width from the pop-up menu.   The JButtons are set to the same size as the button with the longest name.  [JButtons set to the same size](https://netbeans.org/images_www/articles/80/java/quickstart-gui/buttons_6.png) |

### Indentation

Often it is necessary to cluster multiple components under another component such that it is clear they belong to a group of related functions. One typical case, for example, is placing several related checkboxes below a common label. The GUI Builder enables you to accomplish indenting easily by providing special guidelines suggesting the preferred offset for your operating system's look and feel.

In this section we'll add a few JRadioButtons below a JLabel that will allow users to customize the way the application displays data. Refer to the following illustrations while accomplishing this or click the View Demo link following the procedure to view an interactive demonstration.

|  |
| --- |
| To indent JRadioButtons below a JLabel:   1. Add a JLabel named Mail Format to the form below the JList. Make certain the label is left aligned with the JList above. 2. In the Palette window, select the Radio Button component from the Swing category. 3. Move the cursor below the JLabel that we just added. When the guidelines appear indicating that the JRadioButton's left edge is aligned with that of the JLabel, move the JRadioButton slightly to the right until secondary indentation guidelines appear. Shift-click to place the first radio button.   [Secondary Indentation Guidelines for JRadioButton](https://netbeans.org/images_www/articles/80/java/quickstart-gui/07_indent_1.png)   1. Move the cursor to the right of the first JRadioButton. Shift-click to place the second and third JRadioButtons, being careful to respect the suggested component spacing. Make certain to release the Shift key prior to positioning the last JRadioButton. 2. Set the display text for each JRadioButton. (You can edit a button's text by right-clicking the button and choosing Edit Text. Or you can click the button, pause, and then click again.) Enter HTML for the left radio button, Plain Text for the second, and Custom for the third.   Three JRadioButtons are added to the form and indented below the Mail Format JLabel.  [Three JRadioButtons added and renamed](https://netbeans.org/images_www/articles/80/java/quickstart-gui/07_indent_3.png) |

Now we need to add the three JRadioButtons to a ButtonGroup to enable the expected toggle behavior in which only one radio button can be selected at a time. This will, in turn, ensure that our ContactEditor application's contact information will be displayed in the mail format of our choosing.

|  |
| --- |
| To add JRadioButtons to a ButtonGroup:   1. In the Palette window, select the Button Group component from the Swing Controls category. 2. Click anywhere in the GUI Builder design area to add the ButtonGroup component to the form. Notice that the ButtonGroup does not appear in the form itself, however, it is visible in the Navigator's Other Components area. 3. Select all three of the JRadioButtons in the form. 4. In the Properties window, choose buttonGroup1 from the buttonGroup property combo box.   Three JRadioButtons are added to the button group.  [Three JRadioButtons being added to button group](https://netbeans.org/images_www/articles/80/java/quickstart-gui/07_group.png)   1. Press Ctrl-S to save the file. |

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Making the Final Adjustments

We've managed to rough out our ContactEditor application's GUI, but there are still a few things remaining to do. In this section, we'll take a look at a couple of other typical layout tasks that the GUI Builder streamlines.

### Finishing Up

Now we need to add the buttons that will enable users to confirm the information they enter for an individual contact and add it to the contact list or cancel, leaving the database unchanged. In this step, we'll add the two required buttons and then edit them so that they appear the same size in our form even though their display text are different lengths.

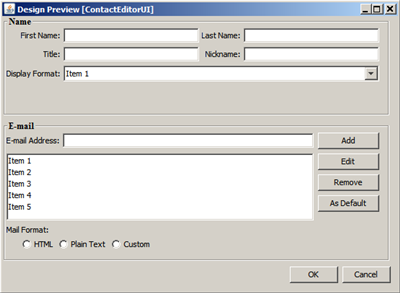
|  |
| --- |
| To add and edit the display text of buttons:   1. If the lower JPanel is extended to the bottom edge of the JFrame form, drag the bottom edge of the JFrame down. This gives you space between the edge of the JFrame and the edge of the JPanel for your OK and Cancel buttons. 2. In the Palette window, select the Button component from the Swing Controls category. 3. Move the cursor over the form below the E-mail JPanel. When the guidelines appear indicating that the JButton's right edge is aligned with the lower right corner of the JFrame, click to place the button.   [Button added below JPanel](https://netbeans.org/images_www/articles/80/java/quickstart-gui/08_cancel.png)   1. Add another JButton to the left of the first, making certain to place it using the suggested spacing along the JFrame's bottom edge. 2. Set the display text for each JButton. Enter OK for the left button and Cancel for right one. Notice that the width of the buttons changes to accommodate the new names. 3. Set the two JButtons to be the same size by selecting both, right-clicking either, and choosing Same Size > Same Width from the pop-up menu.   [Buttons set to the same size](https://netbeans.org/images_www/articles/80/java/quickstart-gui/08_same_size.png)  The JButton components appear in the form and their corresponding nodes are displayed in the Navigator window. The JButton components' code is also added to the form's source file which is visible in the Editor's Source view. Each of the JButtons are set to the same size as the button with the longest name.   1. Press Ctrl-S to save the file. |

The last thing we need to do is delete the placeholder text in the various components. Note that while removing placeholder text after roughing out a form can be a helpful technique in avoiding problems with component alignments and anchoring relationships, most developers typically remove this text in the process of laying out their forms. As you go through the form, select and delete the placeholder text for each of the JTextFields. We'll leave the placeholder items in both the JComboBox and JList for a later tutorial.

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Previewing Your GUI

Now that you have successfully built the ContactEditor GUI, you can try your interface to see the results. You can preview your form as you work by clicking the Preview Form button (https://netbeans.org/images_www/articles/80/java/quickstart-gui/test_form.png) in the GUI Builder's toolbar. The form opens in its own window, allowing you to test it prior to building and running.

[](https://netbeans.org/images_www/articles/80/java/quickstart-gui/08_preview_gui.png)

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

## Deploying GUI Applications

In order for the interfaces you create with the GUI Builder to work outside of the IDE, the application must be compiled against classes for the GroupLayout layout manager and also have those classes available at runtime. These classes are included in Java SE 6, but not in Java SE 5. If you develop the application to run on Java SE 5, your application needs to use the Swing Layout Extensions library.

If you are running the IDE on JDK 5, the IDE automatically generates your application code to use the Swing Layout Extensions library. When you deploy the application, you need to include the Swing Layout Extensions library with the application. When you build the application (Build > Build Main Project), the IDE automatically provides a copy of the library's JAR file in the application's dist/lib folder. The IDE also adds each of the JAR files that are in the dist folder to the Class-Path element in the application JAR file's manifest.mf file.

If you are running the IDE on JDK 6, the IDE generates your application code to use the GroupLayout classes that are in Java SE 6. This means that you can deploy the application to run on systems with Java SE 6 installed and you do not need to package your application with the Swing Layout Extensions library.

**Note:** If you create your application using JDK 6 but you need the application to also run on Java SE 5, you can have the IDE generate its code to use the Swing Layout Extensions library instead of the classes in Java SE 6. Open the ContactEditorUI class in the GUI Editor. In the Navigator, right-click the Form ContactEditorUI node and choose Properties in the popup menu. In the Properties dialog box, change the value of the Layout Generation Style property to Swing Layout Extensions Library.

### Distributing and Running Standalone GUI Applications

To prepare your GUI application for distribution outside of the IDE:

* Zip the project's dist folder into a ZIP archive. (The dist folder might also contain a lib folder, which you would also need to include.)

To run your application, right-click the project name and select Run in the context menu. In the Run Project dialog select the main class name (my.contacteditor.ContactEditorUI if speaking about the project you have just created) and click OK. Your application is up and running.

To run a standalone GUI application from the command line:

1. Navigate to the project's dist folder.
2. Type the following:

java -jar <jar\_name>.jar

**Note:** If you encounter the following error:

Exception in thread "main" java.lang.NoClassDefFoundError: org/jdesktop/layout/GroupLayout$Group

Ensure that the manifest.mf file references the currently installed version of the Swing Layout Extensions Library.

[top](https://netbeans.org/kb/docs/java/quickstart-gui.html#top)

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20Designing%20a%20Swing%20GUI%20in%20NetBeans%20IDE)

## See Also

You have now completed the Designing a Swing GUI tutorial. For information on adding functionality to the GUIs that you create, see:

* [Introduction to GUI Building](https://netbeans.org/kb/docs/java/gui-functionality.html)
* [Handling Images in a GUI Application](https://netbeans.org/kb/docs/java/gui-image-display.html)
* [GUI Builder FAQ](http://wiki.netbeans.org/wiki/view/NetBeansUserFAQ#section-NetBeansUserFAQ-GUIEditorMatisse)
* [Java GUI Applications Learning Trail](https://netbeans.org/kb/trails/matisse.html)
* [Implementing Java GUIs](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG920) in *Developing Applications with NetBeans IDE*

# Introduction to Profiling Java Applications in NetBeans IDE

NetBeans IDE includes a powerful profiling tool that can provide important information about the runtime behavior of your application. The NetBeans profiling tool easily enables you to monitor thread states, CPU performance, and memory usage of your application from within the IDE, and imposes relatively low overhead.

This introductory document is an overview of the profiling tools included in the IDE and a guide to help you quickly start profiling your NetBeans projects. This document is intended to demonstrate the various profiling tasks available in the IDE and the profiling results you can obtain when profiling a project. It does not cover all the profiling features included in the IDE, nor does it explore how to interpret profiling results to resolve specific performance issues you may have in your application.

In this document you will use the profiling tool to obtain profiling data about the Anagram Game sample application, a simple Java application that is included in the IDE. Though the Anagram Game is a very simple Java application project, you would follow the same steps to profile larger, more complex Java applications, as well as web and enterprise application projects.

This document shows how to use the IDE to profile the application and obtain the following profiling results:

* Runtime behavior of an application
* CPU time used by an application's methods
* Object creation

Additionally, this document demonstrates how to take and compare snapshots of profiling results.



**Contents**

* [Profiling for the First Time](https://netbeans.org/kb/docs/java/profiler-intro.html#firsttime)
  + [Creating the Sample Project](https://netbeans.org/kb/docs/java/profiler-intro.html#create)
  + [Using the Profiler for the First Time](https://netbeans.org/kb/docs/java/profiler-intro.html#calibrate)
* [Selecting a Profiling Task](https://netbeans.org/kb/docs/java/profiler-intro.html#select)
  + [Monitoring an Application](https://netbeans.org/kb/docs/java/profiler-intro.html#monitor)
  + [Analyzing Application Performance](https://netbeans.org/kb/docs/java/profiler-intro.html#cpu)
  + [Analyzing Memory Usage](https://netbeans.org/kb/docs/java/profiler-intro.html#memory)
* [Taking Snapshots](https://netbeans.org/kb/docs/java/profiler-intro.html#snapshot)
  + [Taking and Comparing Memory Snapshots](https://netbeans.org/kb/docs/java/profiler-intro.html#takesnap)

## Getting Started

### Prerequisites

This document assumes you have some basic knowledge of, or programming experience with, the following technologies:

* Java Programming
* NetBeans IDE

### Software Needed for the Tutorial

For this tutorial you need to have the following software installed on your computer:

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE](https://netbeans.org/downloads/index.html) | 7.3, 7.4, 8.0 |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | Version 7 or 8 |

**Notes.**

* The profiling tool is bundled with NetBeans IDE, and no special setup is necessary in order to begin profiling an application.
* If you are using NetBeans IDE 7.2 or earlier, see [Introduction to Profiling Java Applications in NetBeans IDE 7.2](https://netbeans.org/kb/72/java/profiler-intro.html)

## Profiling for the First Time

The first time that you use the profiling tool, the IDE needs to perform some initial operations to ensure that accurate profile results are obtained and to integrate the tool with the project. To demonstrate this, you will first create the AnagramGame project and then run the calibration. The IDE will automatically perform the integration the first time you profile the AnagramGame project.

### Creating the Sample Project

In this document you will profile the Anagram Game sample application. To do this you will first use the New Project wizard to create the sample application and set the project as the Main Project.

To create the Anagram Game application perform the following steps.

1. Choose File > New Project (Ctrl-Shift-N; ⌘-Shift-N on Mac) from the main menu.
2. In the New Projects wizard, select the Samples > Java category.
3. Select the Anagram Game project. Click Next.
4. Specify a location for the project. Click Finish.

When you click Finish, the IDE creates the Anagram Game sample project.

1. Choose Run > Set Main Project > AnagramGame from the main menu.

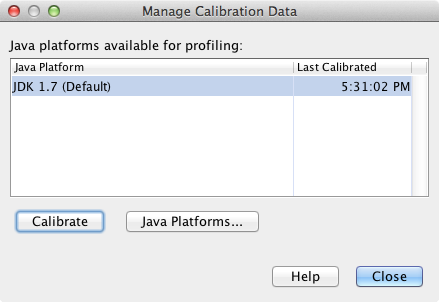
After you set the project as the main project you can see that the name of the Anagram Game project is in bold in the Projects window. By default, when using the IDE to profile a project, the IDE will profile the main project. If no project is set as the main project the IDE will profile the project that is selected in the Projects window.

### Using the Profiler for the First Time

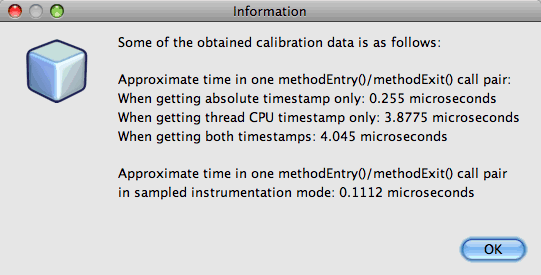
To achieve accurate profiling results, you must have calibration data for each Java platform that will be used for profiling. If you are running the profiling tool for the first time, or if valid calibration data is unavailable for the Java platform, the IDE will prompt you to run the calibration process for your platform.

The calibration only needs to be performed once. However, if you make any substantial changes to the configuration of the machine that could affect machine performance, you should run the calibration again. You can run the calibration at any time by performing the following steps.

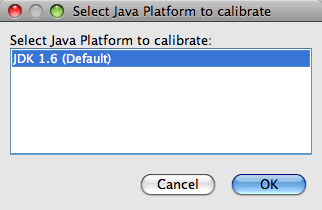
1. Choose **Advanced Commands** > **Manage Calibration Data** from the Profile menu.
2. Select the Java Platform. Click Calibrate.



A dialog box appears when the calibration operation is complete. You can click Show Details to view a dialog box with information about the calibration results. The calibration data for each Java platform is saved in the .nbprofile directory in your home directory.



**Note.** If you are using NetBeans IDE 7.4 or earlier, choose **Advanced Commands** > **Run Profiler Calibration** from the Profile menu and select the Java platform in the Select Java Platform to Calibrate dialog box.



## Selecting a Profiling Task

The IDE provides a number of internal settings that let you tune profiling to your needs. For example, you may decrease the profiling overhead at the cost of some reduction in the amount of generated information. However, it may take some time to understand the meaning and use of the numerous settings available. For many applications, the default settings specified for the profiling tasks are sufficient in most situations.

When profiling a project, you use the Select Profiling Task dialog box to choose a task according to the type of profiling information you want to obtain. The following table describes the profiling tasks and the profiling results obtained from running the task.

|  |  |
| --- | --- |
| **Profiling Task** | **Results** |
| [Monitor Application](https://netbeans.org/kb/docs/java/profiler-intro.html#monitor) | Choose this to obtain high-level information about properties of the target JVM, including thread activity and memory allocations. |
| [Analyze CPU Performance](https://netbeans.org/kb/docs/java/profiler-intro.html#cpu) | Choose this to obtain detailed data on application performance, including the time to execute methods and the number of times the method is invoked. |
| [Analyze Memory Usage](https://netbeans.org/kb/docs/java/profiler-intro.html#memory) | Choose this to obtain detailed data on object allocation and garbage collection. |

The Select Profiling Task dialog box is the main interface for running a profiling task. After you select a task, you can modify the task settings to fine tune the results you will obtain. For each of the profiling tasks, you can also create and save custom profiling tasks based on the task. When you create a custom profiling task, the custom task is listed in the Select Profiling Task dialog box so that you can easily find and run your custom settings later. When you create a custom profiling task, you can modify more advanced profiling settings by clicking **Advanced settings** in the Select Profiling Task dialog box.

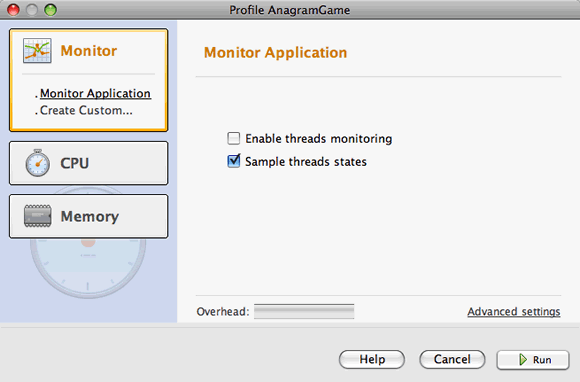
### Monitoring an Application

When you choose the Monitor task, the target application is started without any instrumentation. When monitoring an application, you obtain high-level information about several important properties of the target JVM. Because monitoring an application imposes very low overhead, you can run the application in this mode for extended periods of time.

To monitor the Anagram Game application, perform the following steps.

1. Confirm that the AnagramGame project is set as the main project.
2. Choose Profile > Profile Main Project from the main menu.

Alternatively, right-click the project node in the Projects window and choose Profile.

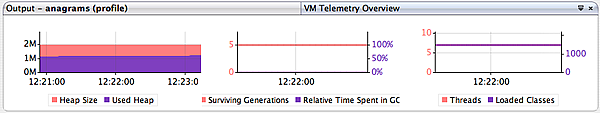
1. Select Monitor in the Select Profiling Task dialog box.
2. Select additional monitor options, if desired. Click Run.  
   

You can hover the cursor over an option to view a tooltip with details about the option.

When you click Run, the IDE launches the application and the Profiler window opens in the left pane of the IDE. The Profiler window contains the controls that enable you to do the following:

* Control the profiling task
* See the status of the current profiling task
* Display profiling results
* Manage profiling results snapshots
* See basic telemetry statistics

You can use the controls in the Profiler window or the main menu to open the windows where you can view the monitoring data. You can use the Telemetry Overview window to quickly get an overview of the monitoring data in real time. If you place the cursor over a graph, you can see more detailed statistics about the data displayed in the graph. You can double-click on any of the graphs in the Telemetry Overview window to open a larger and more detailed version of the graph.



If the overview does not open automatically you can choose Window > Profiling > VM Telemetry Overview to open the overview in the Output window. You can open the VM Telemetry Overview window and see monitoring data at any time during any profiling session.

### Analyzing CPU Performance

When you choose the CPU task, the IDE profiles the method-level CPU performance (execution time) of your application and processes the results in real-time. You can choose to analyze the performance by periodically taking a stack trace or by instrumenting the methods in the application. You can choose to instrument all the methods or limit the instrumentation to a part of the application code, even down to a specific code fragment.

To analyze CPU performance, you choose how the application is profiled by selecting one of the following options.

* **Quick (Sampled).** In this mode, the IDE samples the application and takes a stack trace periodically. This option is less precise than instrumenting methods, but the overhead is lower. This option can help you locate methods that you might want to instrument.
* **Advanced (Instrumented).** In this mode, the methods of the profiled application are instrumented. The IDE records when threads enter and exit project methods enabling you to see how much time is spent in each method. When entering a method, threads generate a "method entry" event. Threads generate a corresponding "method exit" event when exiting the method. The timestamps for both of these events are recorded. This data is processed in real time.

You can choose to instrument all the methods in the application or limit the instrumentation to a subset of the application's code by specifying one or more **root methods**. You can specify a root method using the popup menu in the source code or by clicking **customize** to open the Edit Profiling Roots dialog box.

A root method is the method, class or package in your source code that you specify as an instrumentation root. Profiling data is collected when one of the application's threads enters and leaves the instrumentation root. No profiling data is collected until one of the application's threads enters the root method. Specifying a root method can significantly reduce the profiling overhead. For some applications, specifying a root method may be the only way to obtain any detailed and/or realistic performance data because profiling the entire application may generate so much profiling data that the application becomes unusable or may even cause the application to crash.

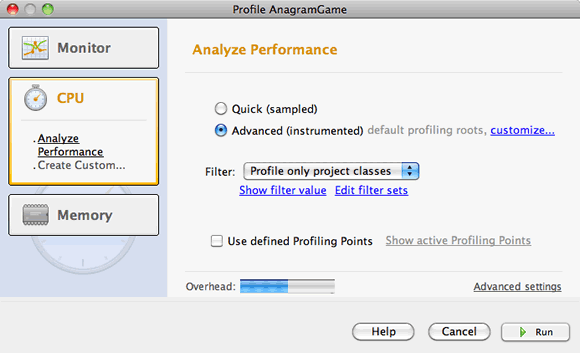
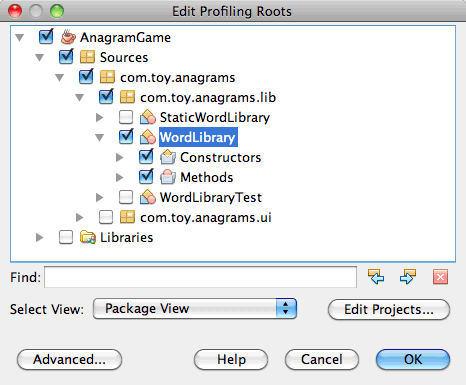
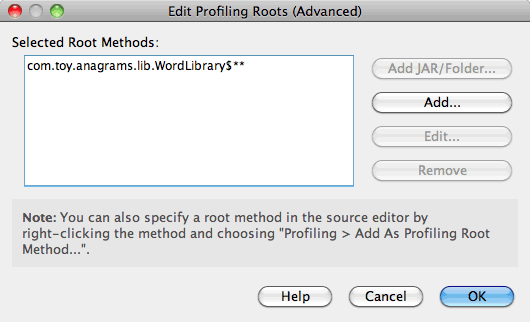
**Note:** The Quick profile mode is not available in NetBeans IDE 7.0 and earlier. You can only use instrumentation to obtain profiling results, but you can choose to instrument the entire application or limit instrumentation to part of the application by specifying one or more root methods.

You can further fine tune how much code is profiled by using a filter to limit the sources that are instrumented.

You will now use the IDE to analyze the CPU performance of the Anagram Game application. You will choose the Part of Application option and then select WordLibrary.java as the profiling root. By selecting this class as the profiling root, you limit the profiling to the methods in this class.

1. Click the Stop button in the Profiler window to stop the previous profiling session (if still running).
2. Choose Profile > Profile Main Project from the main menu.
3. Select CPU in the Select Profiling Task dialog box.
4. Select **Advanced (Instrumented)**.

To use this option you also need to specify a profiling root method.

1. Click **customize** to open the Edit Profiling Roots dialog box.  
   
2. In the Edit Profiling Roots dialog box, expand the AnagramGame node and select Sources/com.toy.anagrams.lib/WordLibrary. When profiling a project you can specify multiple root methods.  
   
3. Click the Advanced button to open the Edit Profiling Roots (Advanced) dialog box which provides more advanced options for adding, editing and removing root methods.  
   

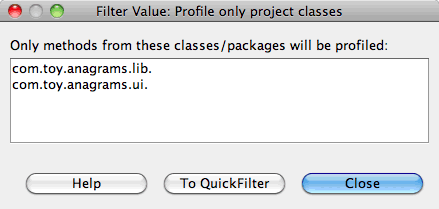
You can see that WordLibrary is listed as the root method. Click OK to close the Edit Profiling Roots (Advanced) dialog box.

1. Click OK to close the Edit Profiling Roots dialog box.

After you select the profiling root you can click **edit** in the Select Profiling Task dialog to modify the selected root method.

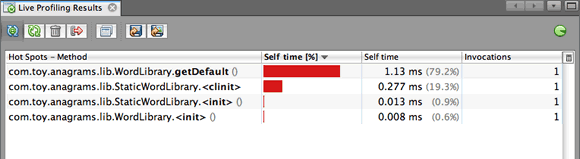
1. Select **Profile only project classes** for the Filter value.

The filter enables you to limit the classes that are instrumented. You can choose from the IDE's predefined profiling filters or create your own custom filters. You can click **Show filter value** to see a list of the classes that will be profiled when the selected filter is applied.



1. Click Run in the Select Profiling Task dialog box to start the profiling session.

When you click Run, the IDE launches the application and starts the profiling session. To view the profiling results, click Live Results in the Profiler window to open the Live Results window. The Live Results window displays the profiling data collected thus far. The data displayed is refreshed every few seconds by default. When analyzing CPU performance, the Live Results window displays information on the time spent in each method and the number of invocations of each method. You can see that in the Anagram Game application only the selected root methods are invoked initially.



You can quickly navigate to the source code containing any of the listed methods by right-clicking the name of the method and choosing Go To Source. When you click Go To Source the class opens in the Source editor.

### Analyzing Memory Usage

The Analyze Memory Usage task gives you data on objects that have been allocated in the target application such as the number, type and location of the allocated objects.

To analyze memory performance, you choose how much data you want to obtain by selecting one of the following options.

* **Quick.** When this option is selected, the profiler samples the application to provide data that is limited to the live objects. This option only tracks live objects and does not track allocations when instrumenting. It is not possible to record stack traces or to use profiling points if you select this option. This option incurs a significantly lower overhead than the Advanced option.
* **Advanced.** When this option is selected application you can obtain information about the number, type, and location of allocated objects. All classes that are currently loaded by the target JVM (and each new class as it is loaded) are instrumented to produce information about object allocations. You need to select this option if you want to use profiling points when analyzing memory usage or if you want to record the call stack. This options incurs a greater profiling overhead than the Quick option.

If you select the Advanced option you can also set the following options.

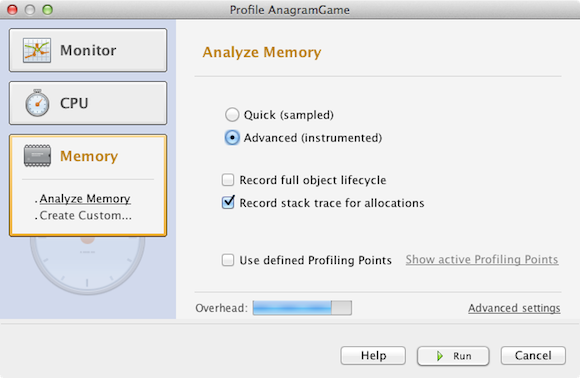
* **Record Full Object Lifestyle.** Select this option to record all information for each object, including the number of generations survived.
* **Record Stack Trace for Allocations.** Select this option to record the full call stack. This option enables you to view the reverse call tree for method calls when viewing a memory snapshot.
* **Use defined Profiling Points.** Select this option to enable profiling points. Disabled profiling points are ignored. When deselected, all profiling points in the project are ignored.

The Overhead meter in the Select Profiling Tasks window gives a rough approximation of the increase or decrease in the profiling overhead according to the profiling options that you select.

In this exercise you will use the IDE to analyze the memory performance of the Anagram Game application. You will choose the **Advanced** option and select the **Record Stack Trace for Allocations** so that the IDE records the full call stack. By selecting this option, when you take the memory snapshot you will be able to view a reverse call tree.

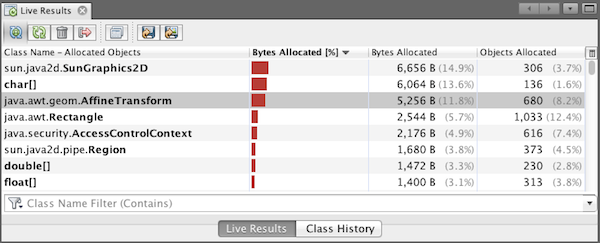
1. Click the Stop button in the Profiler window to stop the previous profiling session (if still running) and stop the Anagram Game application.
2. Choose Profile > Profile Main Project from the main menu.
3. Select Memory in the Select Profiling Task dialog box.
4. Select **Advanced**.
5. Select **Record Stack Trace for Allocations**. Click Run to start the profiling session.

Notice that when you selected this option the Overhead meter increased substantially, but the application is small enough that the performance hit should be manageable.



When you click Run, the IDE launches the application and starts the profiling session. To view the profiling results, click Live Results in the Profiler window to open the Live Results window. The Live Results window displays information on the size and number of objects that are allocated in our project.

By default the results are sorted and displayed by the number of Live Bytes, but you can click a column header to change how the results are displayed. You can also filter the results by typing a class name in the filter box below the list.



## Taking Snapshots

When a profiling session is in progress, you can capture profiling results by taking a snapshot. A snapshot captures the profiling data at the moment you take the snapshot. However, snapshots differ from live profiling results in the following ways:

* Snapshots can be examined when no profiling session is running.
* Snapshots contain a more detailed record of profiling data than live results.
* Snapshots can be easily compared (memory snapshots).

Because a profiling session does not have to be in progress to open a project's snapshots, you can open a project's snapshot at any time by selecting the snapshot in the list of saved snapshots in the Profiler window and clicking Open.

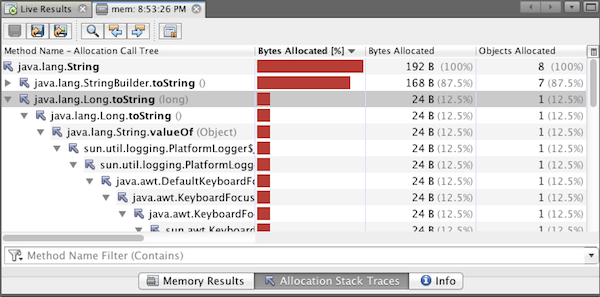
### Taking and Comparing Memory Snapshots

For the Anagram Game application, you can take a snapshot of the results to see the allocation stack trace for the objects of type String. You can then take another snapshot and compare the two. By comparing memory snapshots you can see what objects have been created or released from the heap in the interval between when you took the two snapshots. The snapshots must be comparable, which means that the profiling type (e.g., Allocations vs. Liveness) and the number of tracked objects must match.

In this exercise you will take and save a snapshot to your project. You will then take a second snapshot and compare the second snapshot to the saved snapshot.

1. Make sure the profiling session is still running.  
   (If you stopped the profiling session, repeat the steps to analyze memory performance and open the Live Results window.)
2. Right-click the line containing java.lang.String in the Live Results window and choose Take Snapshot and Show Allocation Stack Traces.

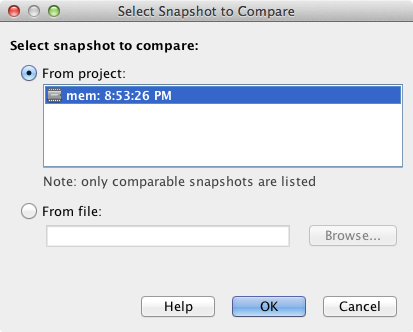
You can use the filter in the Live Results window to help you find the line.



The IDE takes a memory snapshot and opens the snapshot in the Allocation Stack Traces tab. In the Allocation Stack Traces tab you can explore the reverse call tree for the methods that instantiated the selected object.

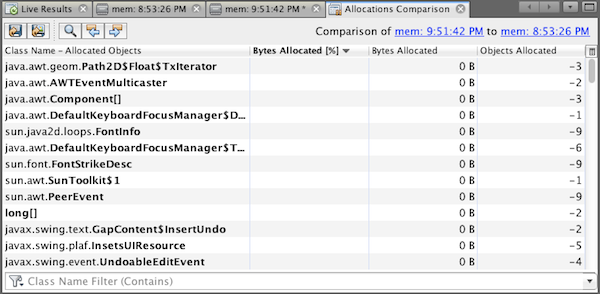
1. Click the Save Snapshot to Project button in the snapshot toolbar (Ctrl-S; ⌘-S on Mac) to save the memory snapshot to your project. When you save the snapshot to your project, the snapshot is added to the list of Anagram Game's saved snapshots in the Profiler window. By default, snapshots are physically saved in the nbproject/private/profiler directory of your project. Saved snapshots are appended with the .nps suffix.

**Note.** You can save snapshots anywhere on your filesystem, however only snapshots saved in the default location in your project will be listed in the Profiler window. You can also click the "Save current view to image" button in the snapshot's toolbar to save the snapshot as an image files (.png) that can be viewed outside of the IDE.

1. Take another snapshot by clicking the Take Snapshot of Collected Results button in the Live Results toolbar (or alternatively, click the Take Snapshot button in the Profiler window). Save the snapshot.
2. In the window of one of the memory snapshots, click the Compute Difference button ( Compare Snapshot button ) in the snapshot toolbar. Alternatively, choose Profile > Compare Memory Snapshots from the main menu.
3. In the Select Snapshot to Compare, select one of the open snapshots from the list. Click OK.  
   

When a memory snapshot is open, you can compare it to other comparable memory snapshots. You can compare the snapshot to unsaved snapshots that are currently open or to snapshots that are saved to the project or elsewhere on your system.

When you click OK, the Liveness Comparison window opens displaying the differences between the two memory snapshots.



The snapshot comparison looks similar to a memory snapshot but only displays the differences between the two compared snapshots. When you look at the numbers in the columns you can see that a plus sign ( + ) indicates that the value increased and a minus sign ( - ) indicates that the value decreased. In the Live Bytes column a graphical bar enables you to easily see the difference in the bytes allocated. If the left half of the cell in that column is green it means that the number of allocated bytes for that object is lower when the second snapshot was taken than it was when the first was taken. If the right half of the cell is red it means that the number of allocated bytes is higher in the second snapshot than in the first.

**Note.** You can also set Take Snapshot profiling points for more precise control over when snapshots are taken. For more on how to take snapshots using profiling points, see [Using Profiling Points in NetBeans IDE](https://netbeans.org/kb/docs/java/profiler-profilingpoints.html).

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20Introduction%20to%20Profiling)

## See Also

This concludes the introduction to profiling an application using NetBeans IDE. This document demonstrated the basics of how to use the IDE to profile a simple NetBeans project and view the profiling results. The steps outlined above can be applied when profiling most projects. Profiling more complex projects such as enterprise applications and free-form projects may require additional configuration steps.

For more detailed information about profiling settings and features not covered in this document, please consult the documentation included in the product and available from the Help menu item.

**NetBeans IDE PHP Quick Start Tutorial**

This document provides general guidelines for preparing the environment for PHP development, setting up a PHP project, and developing and running your first PHP application in the NetBeans IDE for PHP.

**Contents**



* [Component Installation](https://netbeans.org/kb/docs/php/quickstart.html#componentInstallation)
* [Setting up a PHP project in the NetBeans IDE for PHP](https://netbeans.org/kb/docs/php/quickstart.html#setUpProject)
* [Running Your First PHP Project](https://netbeans.org/kb/docs/php/quickstart.html#yurFirstPhpProject)
* [Using Database Servers with NetBeans IDE for PHP](https://netbeans.org/kb/docs/php/quickstart.html#usingDBServers)
* [Next Steps](https://netbeans.org/kb/docs/php/quickstart.html#nextSteps)

**To follow this tutorial, you need the following software and resources.**

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE](https://netbeans.org/downloads/index.html) | PHP download bundle |
| A PHP engine | Version 5 |
| A web server | [Apache HTTP Server 2.2](http://httpd.apache.org/download.cgi) is recommended. |
| A PHP debugger | [XDebug 2.0 or later](http://www.xdebug.org) |

You can install the PHP engine, web server and database separately or use AMP (**A**pache, **M**ySQL, **P**HP) packages.

**Installation and Configuration**

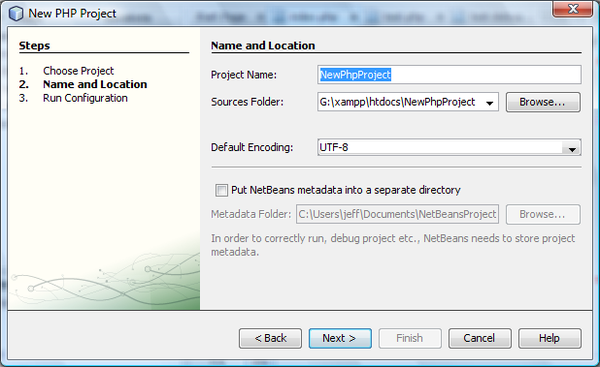
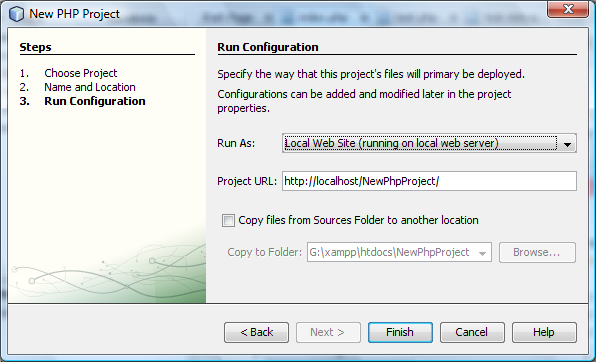
The following documents contain instructions for one or two ways to set up a PHP web stack on your operating system. These instructions are not definitive. The web stack consists of third-party software, your environment may differ, and you might prefer a different AMP package or another way to set up PHP. You might need to supplement our instructions with your own investigations.

* [Configuring PHP Development Environment in Windows](https://netbeans.org/kb/docs/php/configure-php-environment-windows.html)
* [Configuring PHP Development Environment in the Ubuntu Linux Distribution](https://netbeans.org/kb/docs/php/configure-php-environment-ubuntu.html)
* [Configuring PHP Development Environment in Mac Operating System (Mac OS X)](https://netbeans.org/kb/docs/php/configure-php-environment-mac-os.html)

**Setting up a PHP Project in the NetBeans IDE for PHP**

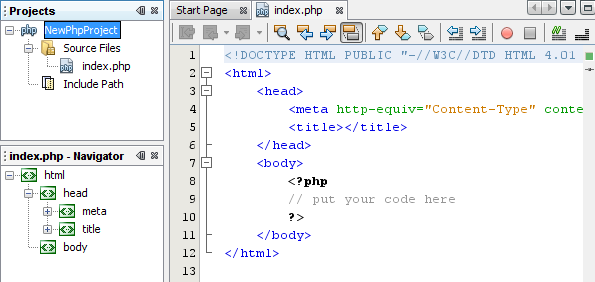
For help in installing and starting NetBeans IDE, please see the [installation documentation](https://netbeans.org/community/releases/73/install.html).

To start PHP development in the NetBeans IDE for PHP, you first need to create a project. A project contains the information on the location of the project files and the way you want to run and debug your application (run configuration).

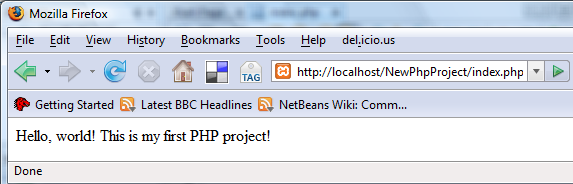
1. Start the IDE, switch to the Projects window, and choose File > New Project. The Choose Project panel opens.
2. In the Categories list, choose PHP.
3. In the Projects area, choose PHP Application and click Next. The New PHP Project > Name and Location panel opens.   
   
4. In the Project Name text field, enter NewPHPProject.
5. In the Sources Folder field, browse for your PHP document root and create a subfolder there called NewPHPProject. The document root is the folder where the web server looks for files to open in the browser. The document root is specified in the web server [configuration file](https://netbeans.org/kb/trails/php.html#configuration). For example, on Xampp, the document root is XAMPP\_HOME/htdocs.
6. Leave all other fields with their default values. Click Next. The Run Configuration window opens.   
   
7. In the Run As drop-down list, select Local Web Site. The project will run on your local Apache server. Your other options are to run the project remotely via FTP and to run it from the command line.
8. Leave the Project URL at default.
9. Click Finish. The IDE creates the project.

Learn more about [Setting up a PHP project in NetBeans](https://netbeans.org/kb/docs/php/project-setup.html).

**Running Your First PHP Project**

1. Start the IDE, choose File > Open Project. The Open Project dialog box opens.
2. Select NewPHPProject and click Open Project. The NewPHPProject tree appears in the Projects window and the project's index.php file opens in the editor and in the Navigator window.   
   
3. Enter the following code inside the <?php ?> block:

echo "Hello, world! This is my first PHP project!";

1. To run the project, position the cursor on the NewPHPProject node and choose Run from the context menu. The figure below shows what you should see in the browser window:  
     
   Congratulations! Your program works!

**Using Database Servers with NetBeans IDE for PHP**

You can use various database servers with the NetBeans IDE for PHP, although the most popular is the MySQL server. Downloads are available [here](http://dev.mysql.com/downloads/mysql/5.1.html).   
**Note:** The recommended version of the product is MySQL Server 5.0.  
See also:

* [Setting Up the MySQL Database Server in the Windows Operating System](https://netbeans.org/kb/docs/ide/install-and-configure-mysql-server.html)
* [Creating a Sample Database with Test Data](https://netbeans.org/kb/docs/php/wish-list-lesson1.html)
* [MySQL and the NetBeans IDE](https://netbeans.org/kb/articles/mysql.html)
* [Connecting to a MySQL Database](https://netbeans.org/kb/docs/ide/mysql.html)

**Next Steps**

Meanwhile, to find information specific to the kind of applications you are developing, use the NetBeans IDE learning trail for that type of application. Each learning trail contains a series of tutorials and guides that range in scope from basic to advanced. The following learning trails are available:

|  |  |
| --- | --- |
| * [PHP Programming](https://netbeans.org/kb/trails/php.html) * [Basic Java Programming](https://netbeans.org/kb/trails/java-se.html) * [Java GUI Applications](https://netbeans.org/kb/trails/matisse.html) * [Web Services Applications](https://netbeans.org/kb/trails/web.html) * [Java EE & Java Web Applications](https://netbeans.org/kb/trails/java-ee.html) | * [Mobile Applications](https://netbeans.org/kb/trails/mobility.html) * [NetBeans Plug-ins and Rich-Client Applications](https://netbeans.org/kb/trails/platform.html) * [C/C++ Applications](https://netbeans.org/kb/trails/cnd.html) |

# Creating a Database Driven Application With NetBeans IDE PHP Editor

#### Tutorial contents:

1. **=> Creating a Database Driven Application With PHP - Main page**
2. Creating the Database
   1. [Creating a MySQL Database](https://netbeans.org/kb/docs/php/wish-list-lesson1.html)
   2. [Creating Oracle Database Tables](https://netbeans.org/kb/docs/php/wish-list-oracle-lesson1.html)
3. [Designing the Application. Reading from the Database](https://netbeans.org/kb/docs/php/wish-list-lesson2.html)
4. [Creating a New Application User](https://netbeans.org/kb/docs/php/wish-list-lesson3.html)
5. [Optimizing the Code](https://netbeans.org/kb/docs/php/wish-list-lesson4.html)
6. [Adding Security. Implementing Application User Logon](https://netbeans.org/kb/docs/php/wish-list-lesson5.html)
7. [Adding a New Wish to the Database](https://netbeans.org/kb/docs/php/wish-list-lesson6.html)
8. [Updating and Deleting Entries in the Database](https://netbeans.org/kb/docs/php/wish-list-lesson7.html)
9. [Making the Application Look Better Using the CSS Technology](https://netbeans.org/kb/docs/php/wish-list-lesson8.html)
10. [Deploying the Application on a Remote Web Server](https://netbeans.org/kb/docs/php/wish-list-lesson9.html)



In this tutorial, you use the PHP support in the NetBeans IDE to create and run a simple web application. The example shows how to create a wish list. After you create and deploy the application, it enables you to share information on wish lists with your friends, such as lists for wedding, birthday, or holiday gifts. The information is stored in the database that you create in Lesson 1.

Any registered user is able to view the wish lists of other users, and you develop this functionality in Lesson 2.

The application supports registration of new users, a functionality that you develop in Lesson 3.

In Lesson 4 you receive tips for making the code easier to maintain.

Lesson 5 provides you with possible solutions for security issues.

In Lessons 6 and 7 you develop functionality that enables each registered user to edit their own wish list.

You add styles to improve the appearance of your application by applying a CSS style sheet in Lesson 8. Finally, you deploy the application on a remote web server using the hints given in lesson 9.

Code is provided for both users of the MySQL database and users of Oracle Database. The tutorial uses the mysqli and OCI8 APIs. For users interested in PDO, a PDO version of the complete tutorial is available for [download](https://netbeans.org/projects/www/downloads/download/php/wishlist-pdo.zip). Note that the PDO\_OCI API is experimental!

**To follow this tutorial, you need the following software and resources.**

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE](https://netbeans.org/downloads/index.html) | PHP download bundle |
| A PHP engine | Version 5 |
| A web server | [Apache HTTP Server 2.2](http://httpd.apache.org/download.cgi) is recommended. |
| A database server | MySQL, OracleXE, or Oracle11g |

For details on installing and configuring the required software, see the document that describes [preparing the environment for PHP development](https://netbeans.org/kb/trails/php.html#configuration) in your operating system.

## Next Step:

[Creating a MySQL Database](https://netbeans.org/kb/docs/php/wish-list-lesson1.html)

or

[Creating Oracle Database Tables](https://netbeans.org/kb/docs/php/wish-list-oracle-lesson1.html)

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20PHP%20Wish%20List%20CRUD%20Main)

To send comments and suggestions, get support, and keep informed on the latest developments on the NetBeans IDE PHP development features, [join the users@php.netbeans.org mailing list](https://netbeans.org/community/lists/top.html).

[Back to the PHP Learning Trail](https://netbeans.org/kb/trails/php.html)

# Getting Started with HTML5 Applications

This document demonstrates how to create an HTML5 project in the IDE and some of the features in the IDE that support the use of JavaScript and CSS in your project. In this document you will create a simple HTML5 application that uses a jQuery JavaScript library to modify a list in a web page.

This document also demonstrates how to install the NetBeans Connector extension for the Chrome browser from the Chrome Web Store.

For a tutorial on how to use jQuery in a NetBeans Java web application, see the tutorial [Using jQuery to Enhance the Appearance and Usability of a Web Page](https://netbeans.org/kb/docs/web/js-toolkits-jquery.html).

### Contents



* [Using the IDE with the Chrome Browser](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#installchrome)
  + [Installing the Extension from the Chrome Web Store](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#installchrome1)
  + [Offline Installation of the Extension](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#installchrome2)
* [Using the Embedded WebKit Browser](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#webkit)
* [Creating a NetBeans HTML5 Project](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#createproject)
* [Installing and Using Bower](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#usingbower)
* [Editing the HTML File](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#editingpage)
* [Saving the Project as a Site Template](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#saving)
* [Summary](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#summary)
* [See Also](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html#seealso)

#### To complete this tutorial, you will need the following resources.

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE, Java](https://netbeans.org/downloads/index.html) | 8.1 or above |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | version 7 or 8 |
| [Chrome Browser](http://www.google.com/chrome) | -- |
| [NetBeans Connector Extension for Chrome](https://chrome.google.com/webstore/detail/netbeans-connector/hafdlehgocfcodbgjnpecfajgkeejnaa?utm_source=chrome-ntp-icon) | 1.0.0 or greater |
| [HTML5 Demo project resources](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5Demo-projectresources.zip) | -- |
| [Site template of finished project](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5DemoSiteTemplate.zip) | -- |

**Notes:**

* The [project resources zip file](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5Demo-projectresources.zip) contains the JPG images and the CSS file that you need to add to the project in this tutorial.
* If you would like to compare your project with a working solution, you can download the [site template](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5DemoSiteTemplate.zip) of the finished project.
* This document assumes you have some basic knowledge of, or programming experience with HTML, CSS, and JavaScript.

## Using the IDE with the Chrome Browser

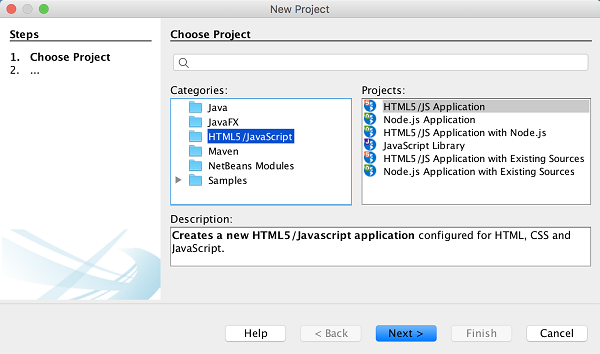
To take full advantage of some of the tools available in the IDE that support HTML5 application development it is recommended that you use the Chrome browser and install the NetBeans Connector extension from the [Chrome Web Store](https://chrome.google.com/webstore/). You only need to install the extension once.

When installed, the NetBeans Connector extension adds a NetBeans Actions menu that is accessible from the URL location bar when you run a NetBeans HTML5 project in the Chrome browser. The NetBeans Actions menu enables you to enable Inspect in NetBeans Mode and to quickly resize the browser window to the dimensions of common display devices. The JavaScript debugger is also automatically enabled when you run an HTML5 application.

### Installing the Extension from the Chrome Web Store

You will automatically be prompted to install the NetBeans Connector extension if you run a NetBeans HTML5 application project from the IDE and select Chrome with NetBeans Integration as the target browser. This exercise demonstrates how to install the extension by creating and running a dummy HTML5 project. You can skip this exercise and install the extension when you are prompted by the IDE or install the NetBeans Connector extension directly from the Chrome Web Store.

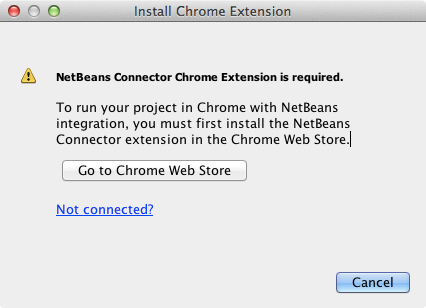
**Note.** The IDE will open the Chrome Web Store in the default browser. If Chrome is not set as the default browser for the IDE you need to open the Options window and select Chrome in the Web Browser dropdown list in the General category before you perform the following steps.

1. Choose File > New Project to open the New Project wizard.
2. Select **HTML5/JS Application** in the **HTML/JavaScript** category. Click Next.  
   
3. Specify a Name and Location for the project. Click Next.

For this exercise the name is not important.

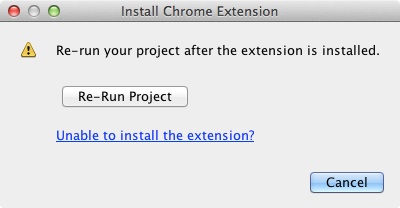
1. Select No Site Template. Click Finish.

When you click Finish the IDE creates a new HTML5 project and opens index.html in the editor.

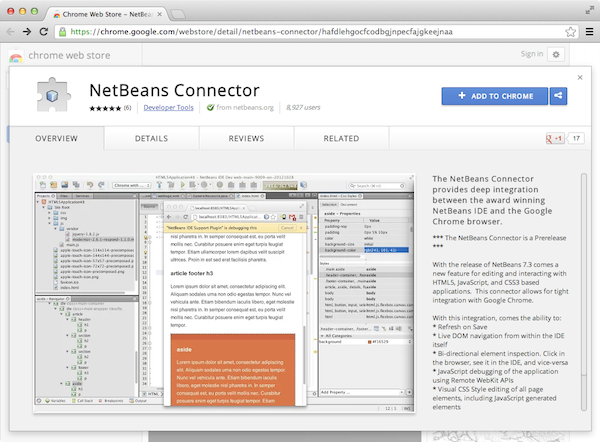
1. Confirm that Chrome with NetBeans Integration is selected in the dropdown list in the toolbar.
2. Click Run in the toolbar.
3. Click Go to Chrome Web Store in the Install Chrome Extension dialog box.  
   

When you click Go to Chrome Web Store the NetBeans Connector page in the Chrome Web Store opens in the Chrome browser.

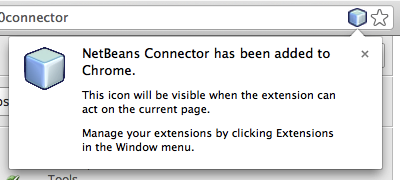
**Note.** The Install Chrome Extension displays a button that you will click after the extension is installed.



1. Go to the Chrome browser and click Add to Chrome in the NetBeans Connector page. Click Add when you are prompted to confirm that you want to add the extension.



When the extension is installed you will see a notification that the extension was added and that the NetBeans Connector icon will be visible in the URL location bar when you run a NetBeans HTML5 project in the Chrome browser.

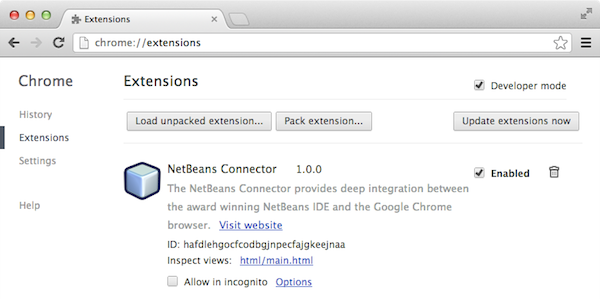


1. In the IDE, Click Re-Run Project in the Install Chrome Extension dialog box.

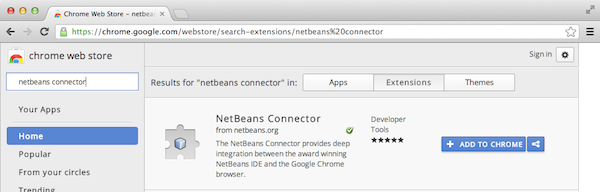
When you click Re-Run Project a new tab will open in the Chrome browser and you will see the index page of the HTML5 application.

After the extension is installed you will see the NetBeans Connector icon in the URL location bar when you run a NetBeans HTML5 project in the Chrome browser.

If you open the Chrome Extensions page (chrome://extensions/) for the Chrome browser you will see that the extension is now enabled.

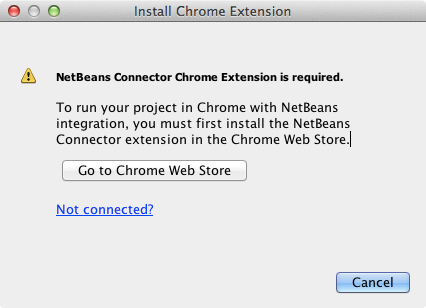
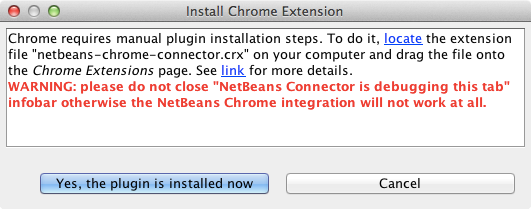
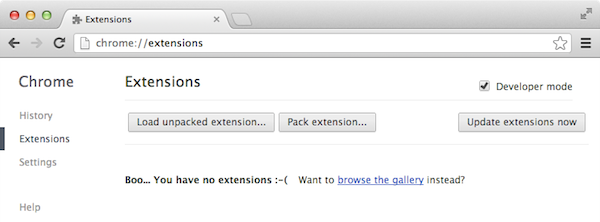


**Note.** Alternatively, you can install the NetBeans Connector extension directly from the Chrome Web Store by performing the following steps.

1. Start the Chrome browser and go to the [Chrome Web Store](https://chrome.google.com/webstore/).
2. Search the Chrome Web Store for the Netbeans Connector extension.
3. Click Add To Chrome in the search results page and click Add when you are prompted to add the extension.   
   

### Offline Installation of the Extension

If you are unable to connect to the Chrome Web Store you can install the NetBeans Connector extension that is bundled with the IDE. If you run a NetBeans HTML5 project and you are prompted to install the NetBeans Connector extension, you can perform the following steps to install the extension if you are not able to access the Chrome Web Store.

1. Click Not Connected in the Install Chrome Extension dialog box.  
   
2. Click **locate** in the dialog box to open the NetBeans IDE installation folder on your local system that contains the **netbeans-chrome-connector.crx** extension.  
   
3. Open the Chrome extensions page (chrome://extensions/) in your Chrome browser.  
   
4. Drag the netbeans-chrome-connector.crx extension into the Extensions page in the browser and click Add to confirm that you want to add the extension.

After the extension is added you will see that the NetBeans Connector extension is added to the list of installed extensions.

1. Click **Yes, the plugin is installed now** in the Install Chrome Extension dialog box to open the NetBeans HTML5 project in the Chrome browser. You will see the NetBeans Connector icon in the location bar of the browser tab.

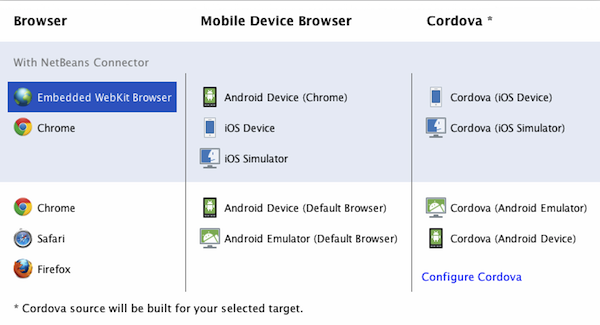
## Using the Embedded WebKit Browser

It is recommended that you run your HTML5 applications in the Chrome browser with the NetBeans Connector extension installed when you are developing the application. The **Chrome with NetBeans Integration** option is selected by default as the run target when you create an HTML5 application. However, it is also possible to run your HTML5 application in the Embedded WebKit Browser that is bundled with the IDE.

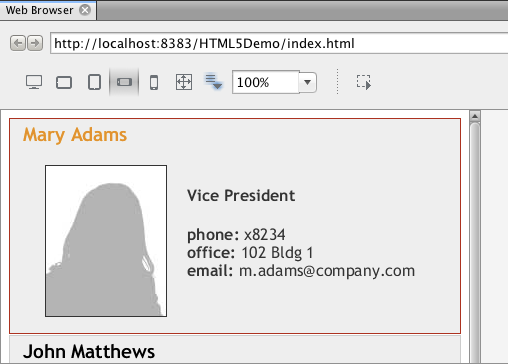
When you run the HTML5 application in the Embedded WebKit Browser the IDE opens the Web Browser window in the IDE. The Embedded WebKit Browser supports many of the features that are enabled in the Chrome browser when the NetBeans Connector extension is installed, including Inspect mode, various screen size options and JavaScript debugging.

**Note.** When you choose Window > Web > Web Browser in the main menu the IDE opens the browser that is specified as the Web Browser in the Options window.

Perform the following steps to run an HTML5 application in the Embedded WebKit Browser.

1. Select Embedded WebKit Browser in the dropdown list in the toolbar.  
   
2. Click Run in the toolbar or right-click the project node in the Projects window and choose Run.

When you run application the Web Browser window opens in the IDE.

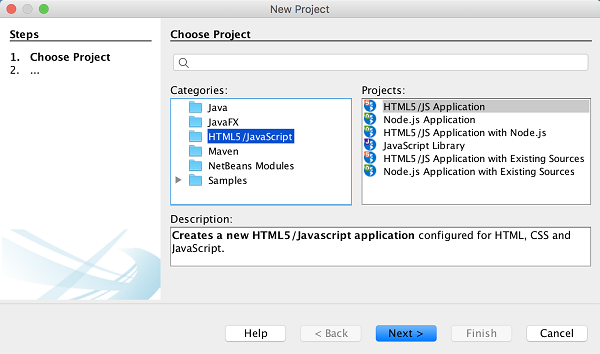


You can click the icons in the toolbar of the Web Browser tab to enable Inspect mode and to quickly switch between different display sizes.

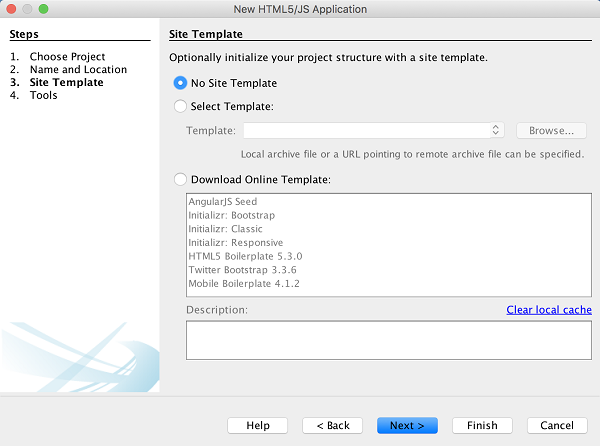
## Creating a NetBeans HTML5 Project

In this exercise you will use the New Project wizard in the IDE to create a new HTML5 project. For this tutorial you will create a very basic HTML5 project that only has an index.html file. In the wizard you will also select some jQuery JavaScript libraries that you want to use in the project.

1. Select File > New Project (Ctrl-Shift-N; ⌘-Shift-N on Mac) in the main menu to open the New Project wizard.
2. Select the **HTML5/JavaScript** category and then select **HTML5/JS Application**. Click Next.



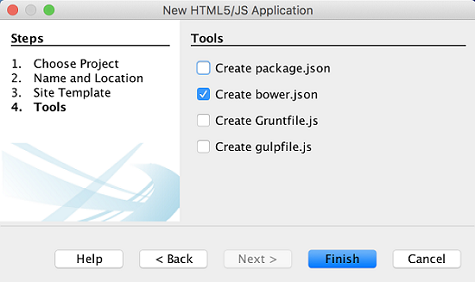
1. Type **HTML5Demo** for the Project Name and specify the directory on your computer where you want save the project. Click Next.
2. In Step 3. Site Template, select No Site Template. Click Next.



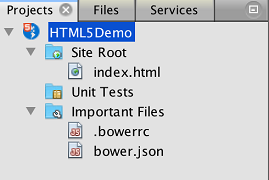
When you select the No Site Template option the wizard generates a basic empty NetBeans HTML5 project. If you click Finish now the project will only contain a Site Root folder and an index.html file in the Site Root folder.

The Site Template page of the wizard enables you to select from a list of popular online templates for HTML5 projects or to specify the location of a .zip archive of a site template. You can type the URL of the .zip archive or click Browse to specify a location on your local system. When you create a project based on a site template the files, libraries and the structure of the project are determined by the template.

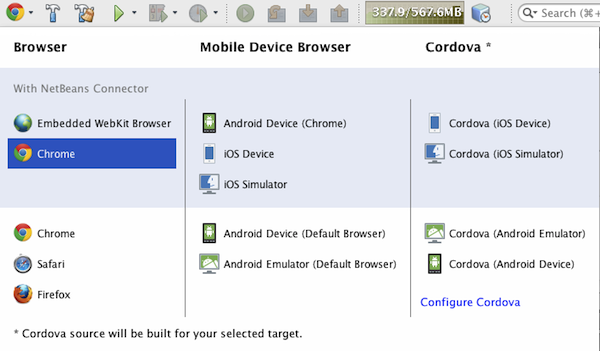
**Note:** You must be online to create a project that is based on one of the online templates in the list.

1. In Step 4. Tools, select only "Bower", which is the standard HTML5 dependency management tool that you can use via the IDE.  
   
2. Click **Finish** to complete the wizard.

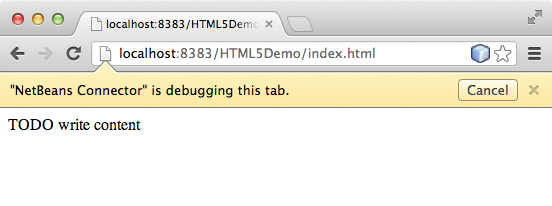
When you click Finish the IDE creates the project and displays a node for the project in the Projects window and opens the index.html file in the editor.



You can now test that your project displays correctly in the Chrome browser.

1. Confirm that Chrome with NetBeans Connector integration is selected in the browser dropdown table in the toolbar.   
   
2. Right-click the project node in the Projects window and choose Run.

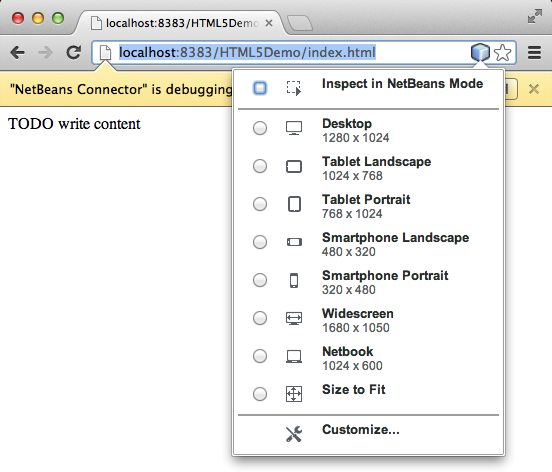
When you choose Run the IDE opens a tab in the Chrome browser and displays the default index.html page of the application. The Browser DOM window opens in the IDE and displays the DOM tree of the page that is open in the browser.



You will notice that there is a yellow bar in the browser tab that notifies you that the NetBeans Connector is debugging the tab. The IDE and the browser are connected and are able to communicate with each other when the yellow bar is visible. When you launch an HTML5 application from the IDE the JavaScript debugger is automatically enabled. When you save changes to a file or make changes to a CSS style sheet you do not need to reload the page because the browser window is automatically updated to display the changes.

If you close the yellow bar or click Cancel you will break the connection between the IDE and the browser. If you break the connection you will need to run the HTML5 application from the IDE again.

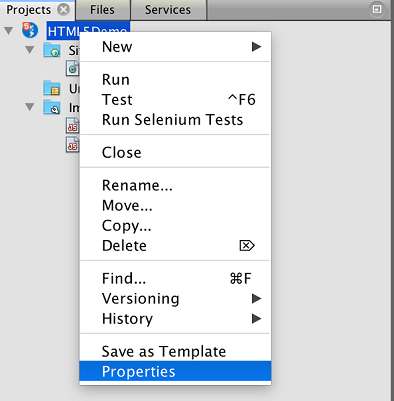
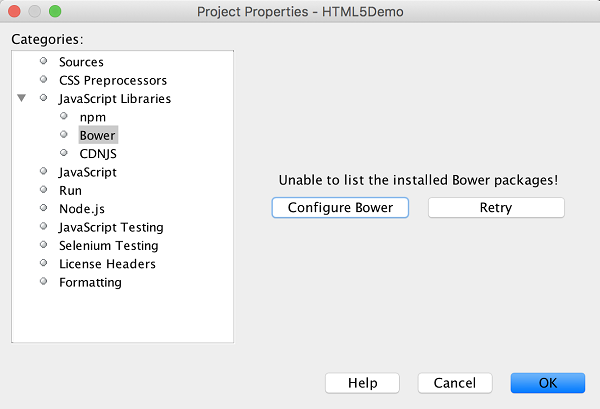
You will also notice that the NetBeans icon is visible in the URL location field of the browser. You can click the icon to open a menu that provides various options for changing the display size of the browser and for enabling the Inspect in NetBeans mode.

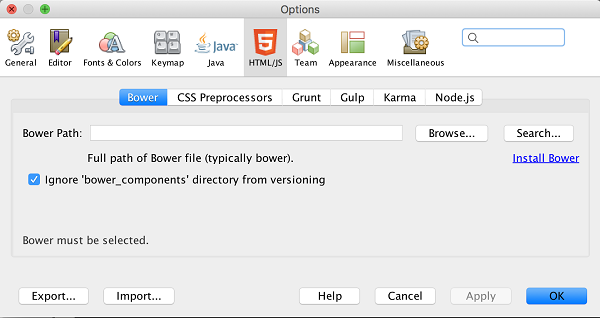


If you select one of the default devices in the menu the browser window will resize to the dimensions of the device. This enables you to see how the application will look on the selected device. HTML5 applications are usually designed to respond to the size of the screen of the device on which they are viewed. You can use JavaScript and CSS rules that respond to the screen size and modify how the application is displayed so that the layout is optimized for the device.

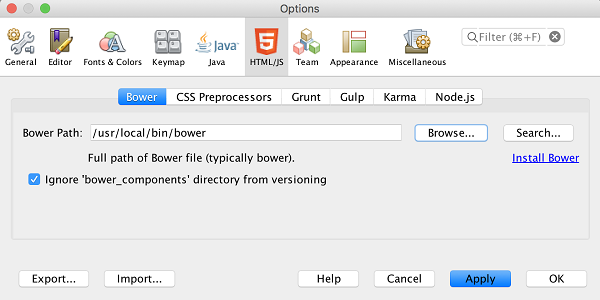
## Installing and Using Bower

In this exercise you will set up Bower ([bower.io](http://bower.io)) and use it to download the JavaScript libraries that you'll use in the sections that follow.

1. Right-click the project and choose Properties.  
   
2. If Bower has not been installed, you will see the tab below. Click Configure Bower. If Bower has been installed, skip to step 5 below.  
   
3. Click the Install Bower link and go through the steps required for setting up Bower. To use Bower, you will also need Node, NPM, and Git. All these are standard tools used in the JavaScript ecosystem. Familiarize yourself with them if needed before continuining.

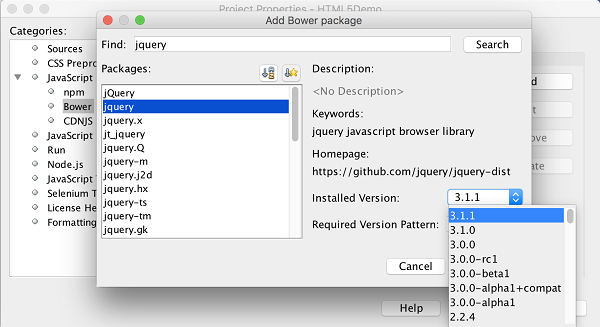


1. Specify the location of the Bower installation. Click Apply.



1. Back in the Bower tab, search for the jquery and jqueryui JavaScript libraries. By default the libraries are created in the public\_html/bower\_components folder of the project. Change the .bowerrc file in your project, if you want the libraries to be installed in a different folder.

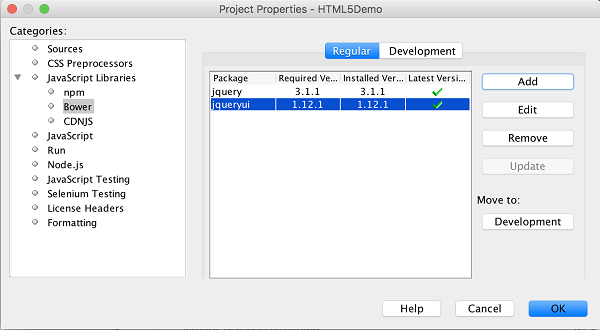
You can use the text field in the panel to filter the list of JavaScript libraries. For example, type **jq** in the field to help you find the jquery libraries. You can Ctrl-click the names of the libraries to select multiple libraries.

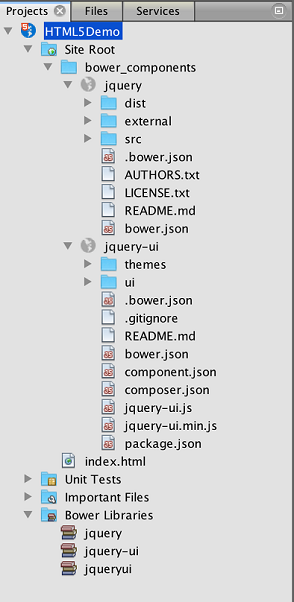


**Notes.**

* + You can click on the library version number in the Version column to open a popup window that enables you to select older versions of the library. By default the wizard displays the most recent version.
  + For this tutorial, choose the latest version of the libraries.

When you have completed this step, you should see the below.



1. Click OK and Bower will download the JavaScript libraries and, once the process is complete, you should see them in the Projects window.  
   

## Editing the HTML File

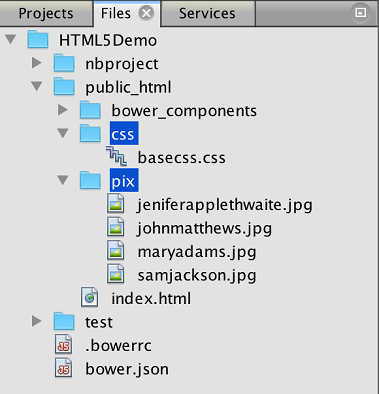
In this exercise you will add the project resources to the project and edit the index.html file to add links to the resources and add some CSS rules. You will see how a few simple CSS selectors when combined with JavaScript can significantly change how a page is displayed in a browser.

1. Download the [project resources](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5Demo-projectresources.zip) archive and extract the contents.

The ZIP archive contains two folders with files that you need to add to the project: pix and css.

1. Copy the pix and css folders into the Site Root folder.

**Note.** If you are looking at the directory structure of the project, you need to copy the folders into the public\_html folder.



1. Open index.html in the editor (if it is not already open).
2. In the editor, add references to the JavaScript libraries that you added when you created the project by adding the following code (in bold) between the opening and closing <head> tags.
3. <html>
4. <head>
5. <title></title>
6. <meta charset=UTF-8">
7. <meta name="viewport" content="width=device-width">
8. **<script type="text/javascript" src="bower\_components/jquery/dist/jquery.js"></script>**
9. **<script type="text/javascript" src="bower\_components/jquery-ui/jquery-ui.js"></script>**
10. </head>
11. <body>
12. TODO write content
13. </body>

</html>

You can use the code completion in the editor to help you.

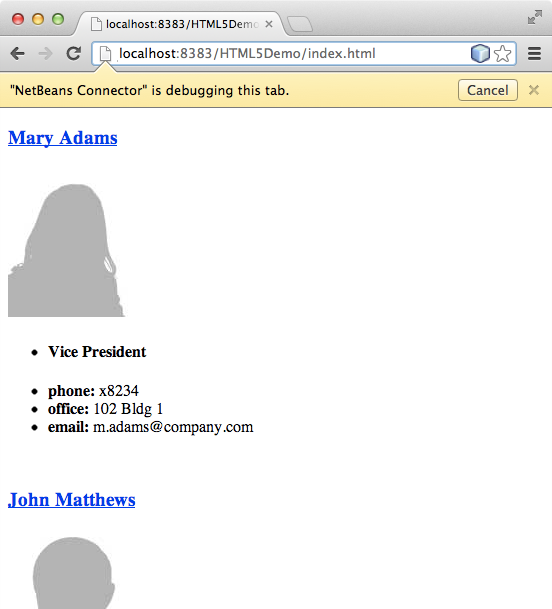


1. Remove the default 'TODO write content' comment and type the following code between the body tags.
2. <body>
3. <div>
4. <h3><a href="#">Mary Adams</a></h3>
5. <div>
6. <img src="pix/maryadams.jpg" alt="Mary Adams">
7. <ul>
8. <li><h4>Vice President</h4></li>
9. <li><b>phone:</b> x8234</li>
10. <li><b>office:</b> 102 Bldg 1</li>
11. <li><b>email:</b> m.adams@company.com</li>
12. </ul>
13. <br clear="all">
14. </div>
15. <h3><a href="#">John Matthews</a></h3>
16. <div>
17. <img src="pix/johnmatthews.jpg" alt="John Matthews">
18. <ul>
19. <li><h4>Middle Manager</h4></li>
20. <li><b>phone:</b> x3082</li>
21. <li><b>office:</b> 307 Bldg 1</li>
22. <li><b>email:</b> j.matthews@company.com</li>
23. </ul>
24. <br clear="all">
25. </div>
26. <h3><a href="#">Sam Jackson</a></h3>
27. <div>
28. <img src="pix/samjackson.jpg" alt="Sam Jackson">
29. <ul>
30. <li><h4>Deputy Assistant</h4></li>
31. <li><b>phone:</b> x3494</li>
32. <li><b>office:</b> 457 Bldg 2</li>
33. <li><b>email:</b> s.jackson@company.com</li>
34. </ul>
35. <br clear="all">
36. </div>
37. <h3><a href="#">Jennifer Brooks</a></h3>
38. <div>
39. <img src="pix/jeniferapplethwaite.jpg" alt="Jenifer Applethwaite">
40. <ul>
41. <li><h4>Senior Technician</h4></li>
42. <li><b>phone:</b> x9430</li>
43. <li><b>office:</b> 327 Bldg 2</li>
44. <li><b>email:</b> j.brooks@company.com</li>
45. </ul>
46. <br clear="all">
47. </div>
48. </div>

</body>

Save your changes.

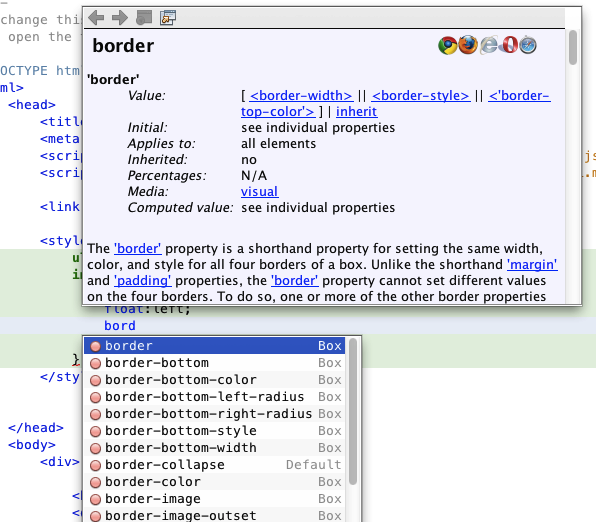
When you save your changes the page automatically reloads in the browser and the page should look similar to the following image.



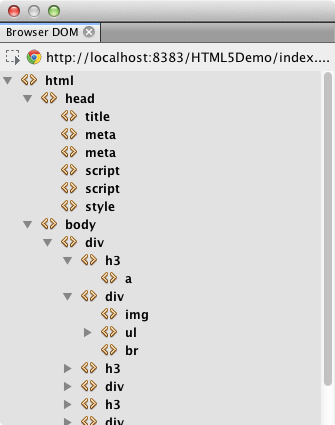
1. Type the following inline CSS rules between the <head> tags in the file.
2. <style type="text/css">
3. ul {list-style-type: none}
4. img {
5. margin-right: 20px;
6. float:left;
7. border: 1px solid;
8. }

</style>

Press Ctrl-Space to use the code completion in the editor when you add the CSS rules.



If you open the Browser DOM window you can see the current structure of the page.



1. Add the following link to the style sheet (in **bold**) between the <head> tags.
2. <head>
3. ...
4. <script type="text/javascript" src="bower\_components/jquery-ui/jquery-ui.js"></script>
5. **<link type="text/css" rel="stylesheet" href="css/basecss.css">**
6. ...

</head>

The basecss.css style sheet is based on some of the CSS rules that are defined in the custom CSS style sheet in the jQuery "UI lightness" theme.

You can open the basecss.css style sheet in the editor and modify the style sheet to add the CSS rules that you added in the previous step or create a new style sheet for the CSS rules.

Add the following code between the <head> tags to run a jQuery script when the elements in the page are loaded.

1. **<script type="text/javascript">**
2. **$(document).ready(function() {**
3. **});**
4. **</script>**

</head>

jQuery works by connecting dynamically-applied JavaScript attributes and behaviors to elements of the DOM (Document Object Model). The jQuery instructions that are used in this example must be executed only after all of the elements of the DOM have been loaded by the browser. This is important because jQuery behaviors connect to elements of the DOM, and these elements must be available to jQuery in order to get the results we expect. jQuery takes care of this for us through its built-in (document).ready function, which follows the jQuery object, represented by $.

You can also use the following abbreviated version of this function.

$(function(){

});

The instructions for jQuery take the form of a JavaScript method, with an optional object literal representing an array of parameters, and must be placed between the curly braces {} inside the (document).ready function in order to execute only at the proper time, which is after the DOM has completely loaded.

Add the following code (in bold) inside the (document).ready function, between the braces {}.

1. <script type="text/javascript">
2. $(document).ready(function() {
3. **$("#infolist").accordion({**
4. **autoHeight: false**
5. **});**
6. });
7. </script>

</head>

This code will invoke the [jQuery accordion widget](http://jqueryui.com/demos/accordion/) script that is included in the [jQuery UI library](http://jqueryui.com/). The accordion script will modify the elements within the DOM object that is identified as infolist. In this code, #infolist is a CSS selector connected to a unique DOM element that has an id attribute with the value infolist. It is connected using typical JavaScript dot notation ('.') to the jQuery instruction that uses the accordion() method to display this element.

In the next step you will identify an element in the page as infolist.

**Note.** You also specified 'autoHeight: false' in the above snippet. This prevents the accordion widget from setting the height of each panel based on the highest content part contained within the markup. For more information, consult the [accordion API documentation](http://docs.jquery.com/UI/Accordion).

The <head> section of the index.html file should look as follows.

<html>

<head>

<title></title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width">

<script type="text/javascript" src="bower\_components/jquery/dist/jquery.js"></script>

<script type="text/javascript" src="bower\_components/jquery-ui/jquery-ui.js"></script>

<link type="text/css" rel="stylesheet" href="css/basecss.css">

<style type="text/css">

ul {list-style-type: none}

img {

margin-right: 20px;

float:left;

border: 1px solid;

}

</style>

<script type="text/javascript">

$(document).ready(function() {

$("#infolist").accordion({

autoHeight: false

});

});

</script>

</head>

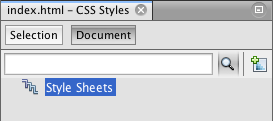
You can tidy up your code by right-clicking in the editor and choosing Format.

Modify the <div> element that encloses the page contents by adding the following id selector and value (in bold).

1. <body>
2. <div **id="infolist"**>

This <div> element encloses the contents of the page (the four sets of <h3> tags and <div> tags that you added earlier in the tutorial).

You can add the selector to the element in the Edit CSS Rules dialog box. To open the Edit CSS Rules dialog box, right-click in the <div> tag in the editor and choose Edit CSS Rules in the popup menu. Alternatively, if the insert cursor is in the <div> tag in the editor you can click the Edit CSS Rules button (screenshot of code completion in the editor) in the CSS Styles window (Window > Web > CSS Styles).



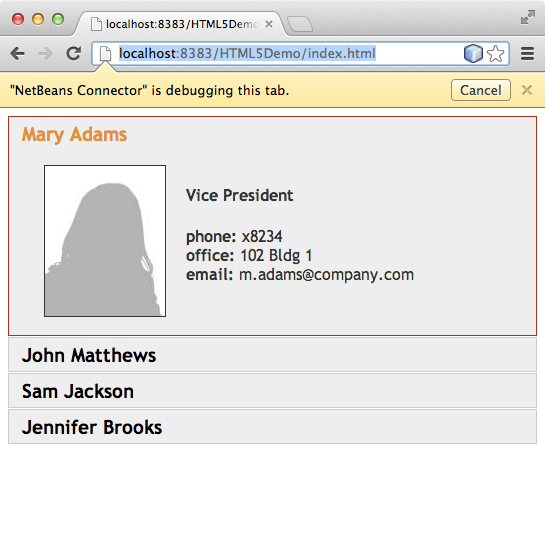
In the CSS Rules dialog box, select id as the Selector Type and type **infolist** as the Selector. Confirm that Apply Changes to the Element is selected.



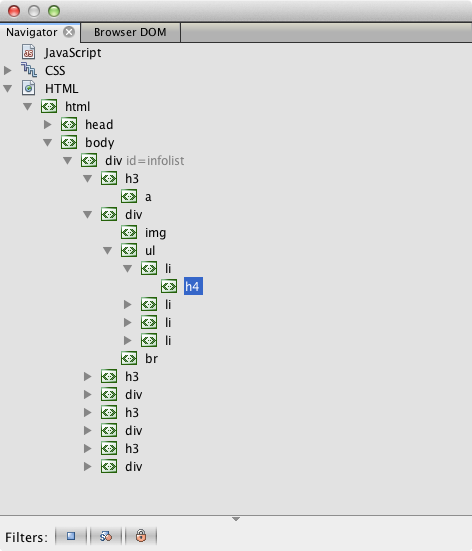
When you click OK in the dialog box a CSS rule for the infolist selector is automatically added to the basecss.css style sheet.

Save your changes to index.html (Ctrl-S; ⌘-S on Mac).

When you save your changes the page in the web browser reloads automatically. You can see that the layout of the page has changed and that the page now uses the CSS style rules that are defined in the basecss.css style sheet. One of the lists below the <h3> is open but the others are now collapsed. You can click an <h3> element to expand the list.

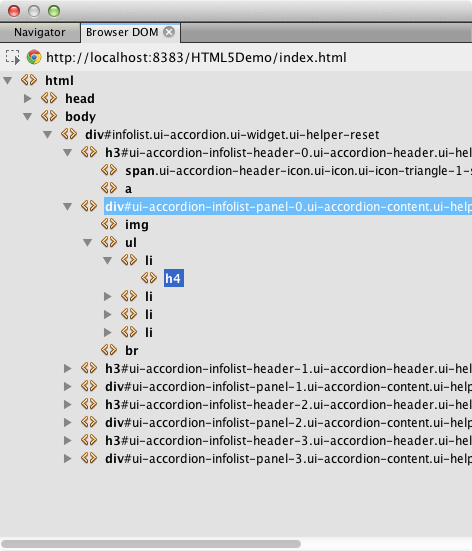


The jQuery accordion function now modifies all the page elements that are contained in the infolist DOM object. In the Navigator window you can see the structure of the HTML file and that the div element that is identified by id=infolist.



You can right-click on an element in the Navigator window and choose Go To Source to quickly navigate to the location of that element in the source file.

In the Browser DOM window you can see the DOM elements in the page that is rendered in the browser and the JQuery styles that are applied to the elements.



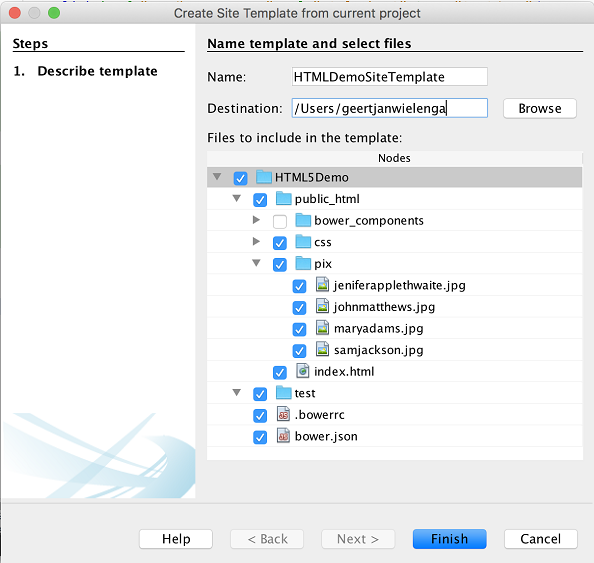
When Inspect in NetBeans Mode is enabled in the browser, when you select an element in the browser window the element is highlighted in the Browser DOM window.

## Saving the Project as a Site Template

You can save your project as a site template that you can use as a template to create other HTML5 sites that are based on the project. The site template can include JavaScript libraries, CSS files, images and templates for HTML files. The IDE provides a wizard to help you select the files that you want to include in the site template.

1. Right-click the project in the Projects window and choose Save as Template in the popup menu.
2. Type **HTML5DemoSiteTemplate** in the Name field and specify the location where you want to save the template.
3. Confirm that all the files are selected. Click Finish.

If you expand the nodes in the tree in the dialog box you can see the files that will be included in the site template.



You can see that the site template will include the index.html file, the CSS style sheet, the images used in the project, though not the JavaScript libraries since Bower can be used by anyone using the template to manage the libraries. The site template can also include any configuration files and tests.

When you click Finish the IDE will generate the site template as a .zip archive.

When you want to create a project that is based on the site template you specify the location of the .zip archive in the Site Template panel of the New Project wizard.

## Summary

In this tutorial you have learned how to create an empty HTML5 project that uses a couple jQuery JavaScript libraries. You also learned how to install the NetBeans Connector extension for the Chrome browser and run the HTML5 project in the browser. When you edited the index.html file you saw that the IDE provides some tools that can help you to edit HTML and CSS files.

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20Getting%20Started%20with%20HTML5%20Applications)

## See Also

For more information about support for HTML5 applications in the IDE on [netbeans.org](https://netbeans.org/), see the following resources:

* [Working with CSS Style Sheets in HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-editing-css.html). A document that continues with the application that you created in this tutorial that demonstrates how to use some of the CSS wizards and windows in the IDE and how to use the Inspect mode in the Chrome browser to visually locate elements in your project sources.
* [Debugging and Testing JavaScript in HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-js-support.html). A document that demonstrates how the IDE provides tools that can help you debug and test JavaScript files in the IDE.

For more information about jQuery, refer to the official documentation:

* Official Home Page: <http://jquery.com>
* UI Home Page: <http://jqueryui.com/>
* Tutorials: <http://docs.jquery.com/Tutorials>
* Documentation Main Page: <http://docs.jquery.com/Main_Page>
* UI Demos and Documentation: <http://jqueryui.com/demos/>

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Working with CSS Style Sheets in an HTML5 Application HTML5 applications typically combine HTML, CSS and JavaScript to create applications that are run in a browser and that are displayed on a variety of devices, including smartphones, tablets and laptops. This document demonstrates how the IDE provides tools that can help you work with CSS rules to modify the layout of an application. The IDE also provides support for creating and using Sass and LESS CSS preprocessors in your application.  The NetBeans Connector extension for the Chrome browser can help you view applications as they would appear on various devices. The extension enables the Chrome browser and the IDE to communicate with each other. The Inspect in NetBeans mode can help you locate elements in your source code by selecting them in the browser. After you select an element in the browser you can use the IDE to view the CSS rules and properties that are applied to that element.  For details on how to install the NetBeans Connector extension for the Chrome browser, see the tutorial [Getting Started with HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html).  To watch a screencast of this tutorial, see [Video of Working with CSS Style Sheets in HTML5 Applications](https://netbeans.org/kb/docs/web/html5-css-screencast.html). Contents Content on this page applies to NetBeans IDE 7.4 and 8.0   * [Creating the NetBeans HTML5 Application](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#create) * [Viewing the Application on Different Devices](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#viewing) * [Modifying the Layout](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#layout)   + [Create a Media Style Sheet](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#newstylesheet)   + [Add CSS Rules](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#addrules) * [Using CSS Preprocessors](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#preprocessors) * [Summary](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#summary) * [See Also](https://netbeans.org/kb/docs/webclient/html5-editing-css.html#seealso)  To complete this tutorial, you will need the following resources.  |  |  | | --- | --- | | **Software or Resource** | **Version Required** | | [NetBeans IDE, Java](https://netbeans.org/downloads/index.html) | 7.4, 8.0 | | [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | 7 or 8 | | [Chrome Browser](http://www.google.com/chrome) | -- | | [NetBeans Connector Extension for Chrome](https://chrome.google.com/webstore/detail/netbeans-connector/hafdlehgocfcodbgjnpecfajgkeejnaa?utm_source=chrome-ntp-icon) | 1.1 or newer | | [HTML5DemoSiteTemplate.zip](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5DemoSiteTemplate.zip) | -- | | CSS preprocessor | [Sass](http://sass-lang.com/install) or [LESS](http://lesscss.org/) |   **Notes:**   * The [HTML5DemoSiteTemplate.zip](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5DemoSiteTemplate.zip) is a site template that you will use when you create the project. * If you would like to compare your project with a working solution, you can download the [site template](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5DemoCssSiteTemplate.zip) of the finished project. * This document assumes you have some basic knowledge of, or programming experience with HTML, CSS, and JavaScript.  Creating the HTML5 Application In this tutorial you use a site template to create the HTML5 project. The site template that you use in this tutorial is the same template that was saved as a site template in the [Getting Started with HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html) tutorial. If you performed the steps in the Getting Started tutorial you can use the site template that you saved in the last section. Alternatively, you can download the HTML5DemoSiteTemplate.zip site template.  Perform the following steps to create the application from a site template.   1. Download the HTML5 site template ([HTML5DemoSiteTemplate](https://netbeans.org/projects/samples/downloads/download/Samples/Web%20Client/HTML5DemoSiteTemplate.zip)) 2. Choose File > New Project (Ctrl-Shift-N; ⌘-Shift-N on Mac) in the main menu to open the New Project wizard. 3. Select the **HTML5** category and then select **HTML5 Application**. Click Next. 4. Type **HTML5DemoCss** for the project name and specify a location. Click Next. 5. Select the **Select Template** option and click Browse to locate the site template (HTML5DemoSiteTemplate.zip). Click Finish.  screenshot of New HTML5 Application wizard   When you click Finish the IDE creates the project and opens the index.html file in the editor. In the Projects window you can see that the project contains index.html, a CSS style sheet, some JavaScript libraries and some images. The CSS Rules and Navigator windows also open in the IDE. Viewing the Application on Different Devices In this tutorial you will run the application in the Chrome browser with the NetBeans Connector extension installed. When the extension is installed you can use the NetBeans menu in the browser to easily resize the browser window to view the application as it would appear on some common devices.  **Notes.** For this tutorial it is recommended that you use the Chrome browser and install the NetBeans extension for Chrome. See the tutorial [Getting Started with HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html) for details on how to install the NetBeans Connector extension.  Perform the following steps to run the application in the Chrome browser.   1. Confirm that Chrome with NetBeans Integration is selected in the dropdown list in the Toolbar. 2. Click the Run button in the toolbar to run the project in the Chrome browser.   In the browser you can see a simple expandable menu.  You will notice that there is a yellow bar in the browser tab that notifies you that the NetBeans Connector is debugging the tab. The IDE and the browser are connected and are able to communicate with each other when the yellow bar is visible. When you launch an HTML5 application from the IDE the JavaScript debugger is automatically enabled. When you save changes to a file or make changes to a CSS style sheet you do not need to reload the page because the browser window is automatically updated to display the changes.  If you close the yellow bar or click Cancel you will break the connection between the IDE and the browser. If you break the connection you will need to launch the HTML5 application from the IDE.  You will also notice that the NetBeans icon is visible in the URL location field of the browser. You can click the icon to open a menu that provides various options for changing the display size of the browser and for enabling the Inspect in NetBeans mode.   1. Click the Open NetBeans Action icon in the URL field of the browser to open the NetBeans menu and select Tablet Portrait in the menu.   The window resizes to the size of a tablet browser in portrait mode. The menu stretches to fill the right side and the entire menu is visible.  screenshot of browser in Tablet Portrait view  If you select one of the default devices in the menu the browser window will resize to the dimensions of the device. This enables you to see how the application will look on the selected device. HTML5 applications are usually designed to respond to the size of the screen of the device on which they are viewed. You can use JavaScript and CSS rules that respond to the screen size and modify how the application is displayed so that the layout is optimized for the device.   1. Click the NetBeans icon again and select Smartphone Landscape in the NetBeans menu.  screenshot of NetBeans menu in browser   The window resizes to the size of a smartphone in landscape orientation and you can see that the bottom part of the menu is not visible without scrolling.  screenshot of browser window resized to Smartphone Landscape  In the next section you will modify the style sheet so that it is possible to view the entire menu without scrolling on a smartphone in landscape view. Modifying the Layout It should be possible to make some minor changes to the elements of the page so that no scrolling is required. These changes should be applied only when the size of the browser is the size of a smartphone or smaller. When oriented in landscape view, the browser window of a smartphone is 480 pixels wide and 320 pixels high. Create a Media Style Sheet In this exercise you will create a new style sheet and add a media rule for devices that have smartphone displays. You will then add some CSS Rules to the media rule.   1. Right-click the css folder node in the Projects window and choose New > Cascading Style Sheet in the popup menu. 2. Type **mycss** as the File Name. Click Finish.   When you click Finish the new style sheet opens in the editor.   1. Add the following media rule to the style sheet. 2. /\*My rule for smartphone\*/ 3. @media (max-width: 480px) {   }  Any CSS rules that you add between the brackets for this rule will only be applied when the dimension of the browser is 480 pixels wide or less.  Create code templates for code snippets that you might use frequently. You can create CSS code templates in the Code Templates tab in the Editor category of the Options window.   1. Save your changes. 2. Open index.html in the editor. 3. Add the following link to the style sheet between the <head> tags in index.html. Save your changes.   <link type="text/css" rel="stylesheet" href="css/mycss.css">  You can use code completion in the editor to help you add the link to the style sheet. Adding CSS Rules  1. In the Chrome browser, click the NetBeans icon and select Inspect in NetBeans Mode in the menu. 2. Click the image in the browser.   An element is highlighted when it is selected in Inspect mode. In this screenshot you can see that the image is highlighted in blue.  screenshot of image selected in the browser  In the IDE you can see that the CSS rules and properties that apply to img are listed in the CSS Styles window. The Selection tab of the CSS Styles window has three panes that provide details about the selected element.  screenshot of CSS Styles window when image is selected Upper Pane In the upper Properties pane of the window you can see that six property-value pairs are applied to the img element. Three of the pairs (border, float and margin) are applied via a CSS rule for the img element. The remaining three pairs are applied because the img element inherits the properties of class selectors that are applied to objects that contain the img element. You can clearly see the structure of the DOM in the Navigator window. The border property is currently selected in the Properties pane. of the CSS Styles window. Middle Pane In the middle Applied Styles pane you can see that the border property and value is specified in a CSS rule that defines the img element. The rule is located on line 12 in the basecss.css file. You can click the location in the pane to open the style sheet in the editor. Lower Pane The lower pane displays all the properties that are defined in the CSS rule for the rule that is selected in the middle pane. In this case you can see that the rule for img defines the border, float and margin-right properties.     1. Click the Document tab in the CSS Styles window. 2. Select the css/mycss.css node and click the Edit CSS Rules button (Edit CSS Rules button) window to open the Edit CSS Rules dialog box. screenshot of Edit CSS Rules dialog box 3. Select Element as the Selector Type and type **img** as the Selector. 4. Select css/mycss.css as the Style Sheet and **(max-width:480px)** as the At-Rule. Click OK. screenshot of Edit CSS Rules dialog box   When you click OK the IDE creates a CSS rule for img in the css/mycss.css style sheet between the brackets of the media rule. The new rule is now listed in the Applied Styles pane.   1. Click the Selection tab in the CSS Styles window.   You can see that there are two CSS rules for img. One of the rules is located in mycss.css and one is located in basecss.css.   1. Select the new img rule (defined in mycss.css) in the Applied Styles pane of the CSS Styles window.  screenshot of styles for the selected element in the CSS Styles window   In the lower pane of the window you can see that the rule has no properties.   1. Click Add Property in the left column in the lower pane of the CSS Styles window and type **width**. 2. Type **90px** in the right column next to the width property and hit Return on your keyboard.  screenshot of Image Properties pane of the CSS Styles window   When you start typing in the value column you can see that a drop-down list displays common values for the width property.  When you hit the Return key the image in the browser is automatically resized to be 90 pixels wide. The IDE added the property to the CSS rule in the mycss.css style sheet. In the editor the style sheet should now contain the following rule.  /\*My rule for smartphone\*/  @media (max-width: 480px) {  img {  width: 90px;  }  }  Some additional changes need to be made to the style sheet because the menu still does not fit within the window.   1. Select the unordered list (<ul>) element in the browser window. screenshot of list element selected in browser   When you select the element you can see that <ul> is selected in the Browser DOM window and you can see the styles that are applied to that element in the CSS Styles window.  screenshot of Browser DOM window  If you select font-family in the CSS Styles window you can see that the font-family property and value are defined in the .ui-widget class selector.   1. Click in the index.html file in the editor and then click the Document tab in the CSS Styles window. 2. Expand the css/mycss.css node in the CSS Styles window. screenshot of Document tab of CSS Styles window 3. Click the Edit CSS Rules button (Edit CSS Rules button) in the CSS Styles window to open the Edit CSS Rules dialog box. 4. Select Class as the Selector Type and type **ui-widget** as the Selector. 5. Select **css/mycss.css** as the Style Sheet and **(max-width:480px)** as the At-Rule. Click OK.   When you click OK the IDE adds the new rule to the mycss.css style sheet and opens the file in the editor. If the file does not open in the editor you can double-click the ui-widget rule under the css/mycss.css node in the CSS Styles window to open the style sheet. The cursor is placed in the line containing the rule in the style sheet.   1. Add the following property and value (in bold) to the rule for ui-widget. 2. .ui-widget { 3. **font-size: 0.9em;**   }  When you change the value in the style sheet the page automatically updates in the browser window.  You can type the property and value in the editor and use the code completion to help you. Alternatively, you can select the .ui-widget rule in the upper pane and click the Add Property button in the lower pane to open the Add Property dialog box.  screenshot of CSS code completion in the editor  After you add the rule you can see that the menu now fits in the page.  screenshot of page in browser after CSS rules are applied   1. Click the NetBeans icon in the browser and select Tablet Portrait in the menu.   When the browser window resizes you can see that the changes to the style sheet do not affect the display when the screen size is larger than 480 pixels wide. Using CSS Preprocessors In addition to tools for editing standard CSS files, the IDE also supports using Sass and LESS CSS preprocessors to generate stylesheets for your applications. The IDE provides wizards for generating CSS preprocessor files and for specifying watched directories. If you specify a watched directory the CSS files will be generated automatically each time you modify the CSS preprocessor files in the directory.  **Note.** To use a CSS preprocessor you need to install the preprocessor software and specify the location of the executable. You can specify the location of the executable in the Options window.   1. Install the CSS preprocessor software on your local system.   The IDE supports the [Sass](http://sass-lang.com/) and [LESS](http://lesscss.org/) preprocessors. This tutorial demonstrates how to use Sass to generate the CSS files, but the configuration for LESS is similar.  **Note.** If you are installing LESS on OS X you might need to confirm that Node.js can be found in the usr/bin directory. For details, see the following [note](http://stackoverflow.com/questions/8638808/problems-with-less-sublime-text-build-system).   1. Expand the HTML5Demo project in the Files window. 2. Right-click the public\_html folder in the Files window and choose New > Folder in the popup menu.   If Folder is not an option in the popup menu, choose Other and then select the Folder file type in the Other category of the New File wizard.   1. Type **scss** for the File Name. Click Finish.   When you click Finish the IDE generates the new folder in the public\_html folder.   1. Right-click the scss folder node in the Projects window and choose New > Sass File in the popup menu. 2. Type **mysass** as the File Name. 3. Click Configure to open the CSS Preprocessor tab in the Options window. 4. Type the path to the Sass executable or click Browse to locate the executable on your local system. Click OK to close the Options window.   screenshot of page in browser after CSS rules are applied   1. Select Compile Sass Files on Save in the New File wizard. Click OK. screenshot of page in browser after CSS rules are applied   When you click OK the new Sass file mysass.scss opens in the editor.   1. Add the following to mysass.scss and save your changes. 2. img { 3. margin-right: 20px; 4. float:left; 5. border: 1px solid; 6. @media (max-width: 480px) { 7. width: 90px; 8. } 9. } 10. .ui-widget { 11. @media (max-width: 480px) { 12. font-size: 0.9em; 13. li { 14. color: red; 15. } 16. }   }  When you save the file the Sass preprocessor generates a new CSS file mysass.css in the css folder. If you open mysass.css in the editor you can see the rules that are generated from the scss file. By default, CSS debug information is generated in mysass.css. When you no longer want the debug information generated you can disable generation in the CSS Preprocessors tab in the Options window.  **Notes.**   * + When you want to modify the CSS rules you should edit the Sass preprocessor file mysass.scss file and not the mysass.css style sheet because the style sheet is regenerated each time the preprocessor file is modified and saved.   + For documentation about Sass syntax and other Sass features, see the [Sass reference](http://sass-lang.com/documentation/file.SASS_REFERENCE.html).  1. Open index.html and make the following changes to change the link to the style sheet from mycss.css to mysass.css. Save your changes.   <link type="text/css" rel="stylesheet" href="css/**mysass.css**">  When you save the file the page in the browser is automatically updated. You can see that the list item elements are now red.   See Also For more information about support for HTML5 applications in the IDE on [netbeans.org](https://netbeans.org/), see the following resources:   * [Getting Started with HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html). A document that demonstrates how to install the NetBeans Connector extension for Chrome and creating and running a simple HTML5 application. * [Debugging and Testing JavaScript in HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-js-support.html). A document that demonstrates how the IDE provides tools that can help you debug and test JavaScript files in the IDE. * [Developing HTML5 Applications](http://docs.oracle.com/cd/E50453_01/doc.80/e50452/dev_html_apps.htm) chapter in the [Developing Applications with NetBeans IDE User's Guide](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG)   For more information about jQuery, refer to the official documentation:   * Official Home Page: <http://jquery.com> * UI Home Page: <http://jqueryui.com/> * Tutorials: <http://docs.jquery.com/Tutorials> * Documentation Main Page: <http://docs.jquery.com/Main_Page> * UI Demos and Documentation: <http://jqueryui.com/demos/> |

# Debugging and Testing JavaScript in an HTML5 Application

HTML5 applications typically combine HTML, CSS and JavaScript to create applications that are run in a browser and that are displayed on a variety of devices, including smartphones, tablets and laptops. This document demonstrates how the IDE provides tools that can help you debug and test JavaScript files in the IDE.

When you want to debug the JavaScript files in your HTML5 application it is recommended that you install the NetBeans Connector extension for the Chrome browser. Debugging is enabled automatically when you run the application in the browser and the extension is installed.

The IDE also enables you to easily configure and run unit tests on JavaScript files using the Jasmine testing framework and the Karma or the JS Test Driver test runners. You can configure the test runner to run unit tests against a variety of browsers and you can quickly specify the JavaScript libraries, scripts and tests that you want the IDE to load when running the tests. When a test fails you can use the debugger to help you locate the problematic code.

For details on how to install the NetBeans Connector extension for the Chrome browser, see the tutorial [Getting Started with HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html).

For more information on the JavaScript editing features in the IDE, see [Creating JavaScript Files](http://docs.oracle.com/cd/E50453_01/doc.80/e50452/dev_html_apps.htm#BACFIFIG) in the [Developing Applications with NetBeans IDE User's Guide](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG).

To watch a screencast of this tutorial, see [Video of Testing and Debugging JavaScript in HTML5 Applications](https://netbeans.org/kb/docs/web/html5-javascript-screencast.html).

### Contents

* [Creating the NetBeans HTML5 Application](https://netbeans.org/kb/docs/webclient/html5-js-support.html#createproject)
* [Using the JavaScript Debugger](https://netbeans.org/kb/docs/webclient/html5-js-support.html#debugger)
* [Running JS Unit Tests](https://netbeans.org/kb/docs/webclient/html5-js-support.html#unittests)
  + [How to Run Tests with Karma](https://netbeans.org/kb/docs/webclient/html5-js-support.html#karmatests)
  + [How to Run Tests with JS Test Driver](https://netbeans.org/kb/docs/webclient/html5-js-support.html#jstestdriver)
* [Debugging a JS Unit Test with JS Test Driver](https://netbeans.org/kb/docs/webclient/html5-js-support.html#debugtest)
* [Summary](https://netbeans.org/kb/docs/webclient/html5-js-support.html#summary)
* [See Also](https://netbeans.org/kb/docs/webclient/html5-js-support.html#seealso)

#### To complete this tutorial, you will need the following resources.

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE, Java](https://netbeans.org/downloads/index.html) | 8.0 |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | 7 or 8 |
| [Chrome Browser](http://www.google.com/chrome) | -- |
| [NetBeans Connector Extension for Chrome](https://chrome.google.com/webstore/detail/netbeans-connector/hafdlehgocfcodbgjnpecfajgkeejnaa?utm_source=chrome-ntp-icon) | 1.x |
| [JS Test Driver JAR](http://code.google.com/p/js-test-driver/) | -- |

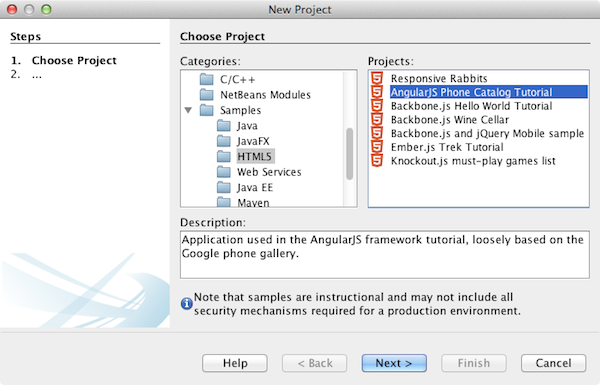
**Notes:**

* This document uses JS Test Driver server to run the JavaScript unit tests. You might want to familiarize yourself with the properties of the server at the [JS Test Driver project home](http://code.google.com/p/js-test-driver/).
* This document assumes you have some basic knowledge of, or programming experience with HTML, CSS, and JavaScript.

## Creating the HTML5 Sample Application

Perform the following steps to create the HTML5 sample application from a site template.

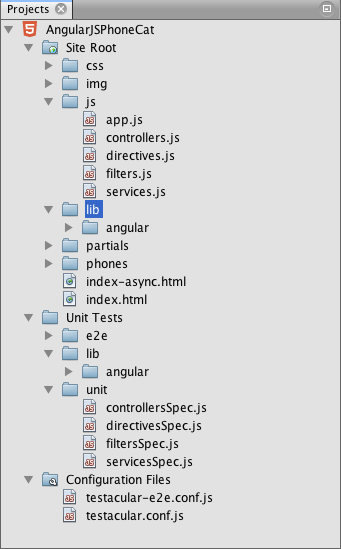
1. Choose File > New Project (Ctrl-Shift-N; ⌘-Shift-N on Mac) in the main menu to open the New Project wizard.
2. Expand the **Samples** node in the New Project wizard and select the **HTML5** category.
3. Select the **AngularJS Phone Catalog Tutorial** project. Click Next



1. Specify a location for the project. Click Finish.

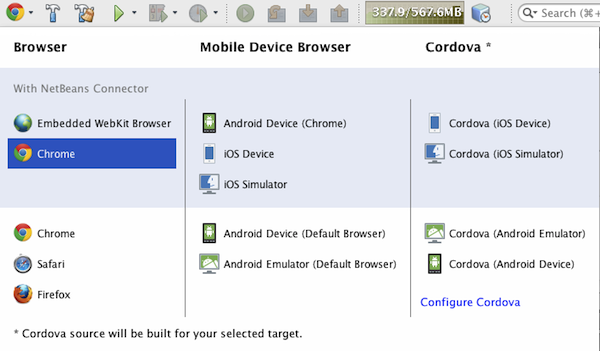
**Note.** Note that the URL of the template is github. The IDE needs to be able to access the network to retrieve the template archive. Check your proxy settings in the Options window if you encounter problems downloading the archive.

When you click Finish the IDE creates the project and opens the index.html file in the editor. In the Projects window you can see that the project contains index.html and various CSS style sheets and JavaScript files and libraries.



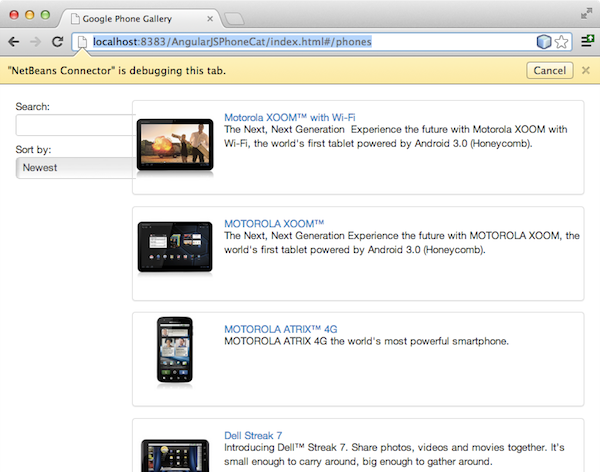
The project also includes several JavaScript unit test and configuration files that were generated by default.

1. Confirm that Chrome with NetBeans Connector is selected in the dropdown list in the toolbar.



1. Click the Run button in the toolbar (F6) or right-click the project node in the Projects window and choose Run.

When you Run the project the front page of the HTML5 application opens in the Chrome browser and you can see a list of mobile phones. When you click the name of a mobile phone the page displays the phone details.



You will notice that there is a yellow bar in the browser tab that notifies you that the NetBeans Connector is debugging the tab. The IDE and the browser are connected and are able to communicate with each other when the yellow bar is visible. When you launch an HTML5 application from the IDE the JavaScript debugger is automatically enabled. When you save changes to a file or make changes to a CSS style sheet you do not need to reload the page because the browser window is automatically updated to display the changes.

If you close the yellow bar or click Cancel you will break the connection between the IDE and the browser. If you break the connection you will need to run the HTML5 application from the IDE again to use JavaScript debugger.

You will also notice that the NetBeans icon is visible in the URL location field of the browser. You can click the icon to open a menu that provides various options for changing the display size of the browser and for enabling the Inspect in NetBeans mode.

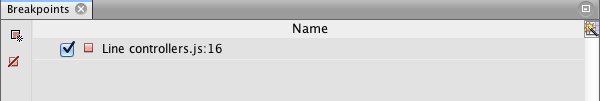
## Using the JavaScript Debugger

In this exercise you will place a breakpoint in a JavaScript file and run the application. You can use the tooltip in the editor to quickly see the values of variables.

1. Expand the js node in the Projects window and double-click controllers.js to open the file in the editor.
2. Place a line breakpoint on line 16 in controllers.js by clicking in the left margin.



You can view the breakpoints that are set in the project by choosing Window > Debugging > Breakpoints to open the Breakpoints window.

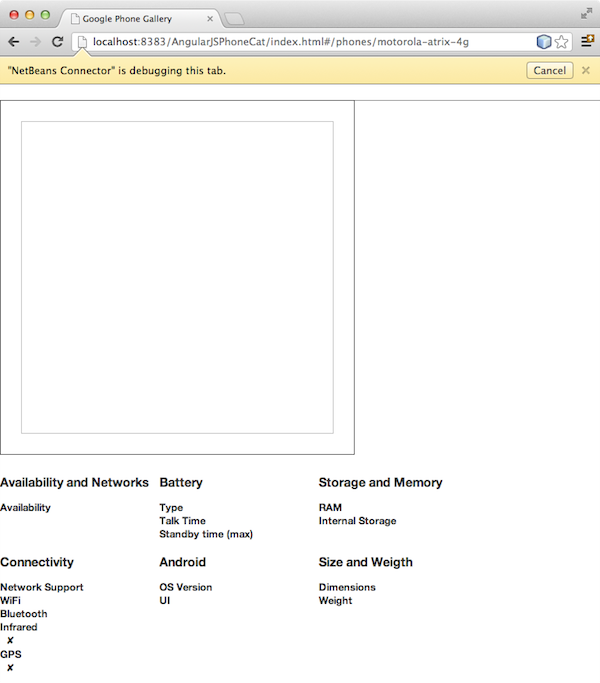


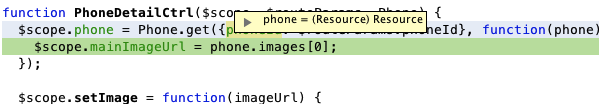
1. Click the Run button in the toolbar to run the project again.

When you run the project you will see the same page because the breakpoint that you set was not hit.

1. In the browser, click one of the entries in the page, for example, Motorola Atrix4G.

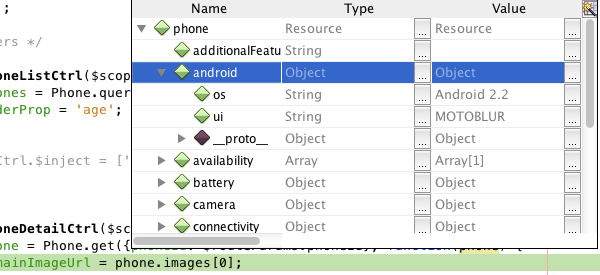
You will see that the page is partially loaded but that the data for the phone is missing because the data has not been passed to the JavaScript and rendered.



1. In the editor in the IDE you can see that the breakpoint was hit and that the Program Counter is currently in line 16 of controllers.js.
2. Hover your cursor over the phone variable to view a tooltip with information about the variable.  
   

In the tooltip you can see the following information: phone = (Resource) Resource.

1. Click the tooltip to expand the tooltip and view a list of the variables and values.



For example, when you expand the android node you can see the values of the strings os and ui.

You can also choose Window > Debugging > Variables to view the list in the Variables window.

1. Use the step buttons in the toolbar to step through the JavaScript functions in the angular.js library or click the Continue button (F5) to resume the application.

## Running JS Unit Tests

You can easily configure the IDE to use the Karma or JS Test Driver test runners to run your unit tests. Karma and JS Test Driver are test runners that provide a URL that is the target for running JavaScript unit tests.

In this tutorial you will use Karma to run the JavaScript unit tests that are included with the sample project. The sample project already includes a Karma configuration file. When you run your tests the test runner server starts and waits to run the tests. Your browser opens and displays a status message in the window that confirms that the server is running and waiting.

### How to Run Tests with Karma

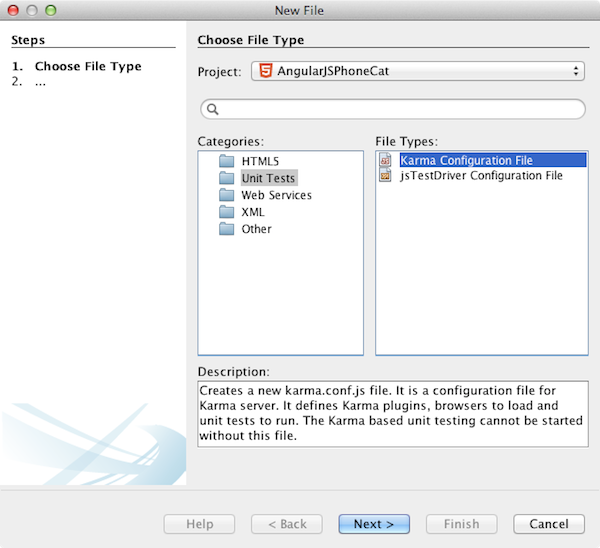
To run tests with Karma you first need to download Karma to you local file system. After you install Karma you need to create a Karma configuration file and then specify the location of the installation and configuration file in the Project Properties window.

1. Install Karma.

You can choose how and where you want to install Karma. You will specify the installation later when you configure the project to use Karma. You can find information about the options for installing Karma on the [Karma website](http://karma-runner.github.io).

1. Create a Karma configuration file.

In this tutorial this step is optional because the sample application already includes a Karma configuration file. You can create a skeleton Karma configuration file by selecting Karma Configuration File in the Unit Tests category of the New File wizard.



Alternatively, you can run the Karma init command on the command line. See the Karma documentation for more details on using the Karma init command.

1. Expand the Configuration Files node in the Projects window and double-click karma.conf.js to open the file in the editor. Note that the sample includes two Karma configuration files.

In the Karma configuration file you can see the files that will be included and excluded when you run the tests. You can also see the Karma plugins that are required to run the tests with this configuration.

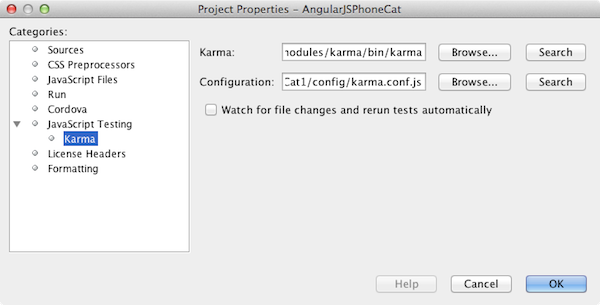


1. Right-click the project node in the Projects window and choose Properties in the popup menu.
2. Select JavaScript Testing category in the Categories pane of the Project Properties window.
3. Select Karma in the Testing Provider drop-down list. Click OK.
4. Open the Project Properties window again and select Karma under the JavaScript Testing category in the Categories pane.
5. Specify the location of your Karma installation.

If you installed Karma in your project directory you can click Search and the IDE will find the installation. You can also click Browse to manually locate your local Karma installation.

1. Specify the location of your Karma configuration file. Click OK.

In this tutorial you can click Search and the IDE will find the default Karma configuration file. You can also click Browse to manually locate a configuration file.



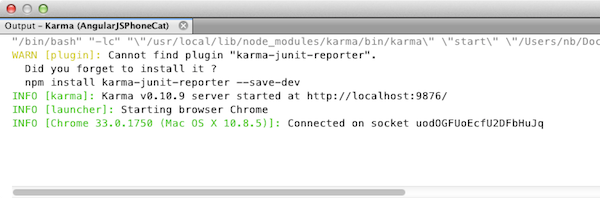
When you click OK you can see that a Karma node appears under the project node in the Projects window. You right-click the Karma node and start and stop the Karma server and set the configuration file in the popup menu.

1. Right-click the Karma node in the Projects window and choose Start in the popup menu.

When you click Start the Karma server starts and a browser window opens that displays the status of the server.

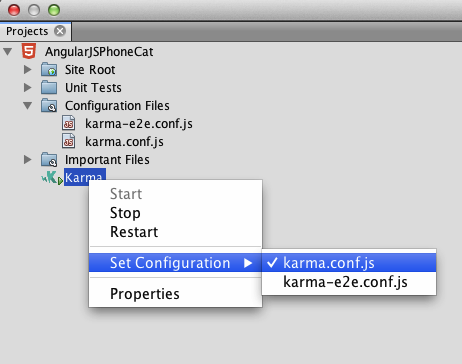


In the Output window you can see the status of the server. You are also prompted to install any missing plugins.



**Note.** The browser window must be open and the Karma server must be running to run the unit tests.

1. Right-click the Karma node and choose Set Configuration > karma.conf.js to confirm that the correct configuration file is selected.

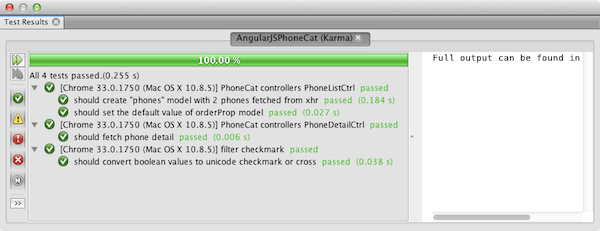


1. Disable any breakpoints that you set in the project.

You can disable the breakpoints by deselecting the checkbox for the breakpoints in the Breakpoints window.

1. Right-click the project node in the Projects window and choose Test.

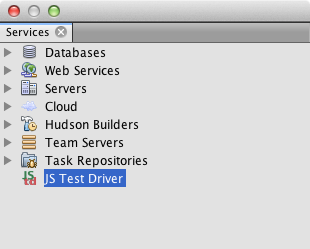
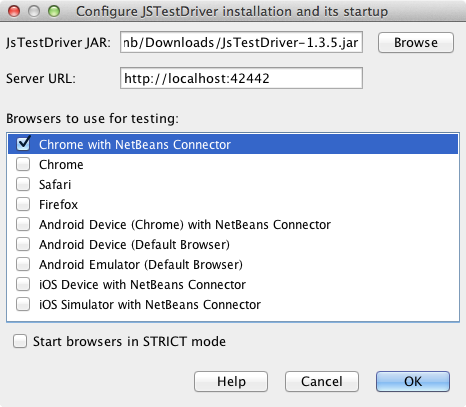
When you choose Test the test runner runs the unit tests on the files. The IDE opens the Test Results window and displays the results of the test.



### How to Run Test with JS Test Driver

If you want to use JS Test Driver, the IDE provides a configuration dialog for JS Test Driver that you can open from the JS Test Driver node in the Services. The configuration dialog enables you to easily specify the location of the JS Test Driver server JAR and the browsers that you want to run tests against. The JS Test Driver node enables you to quickly see if the server is running and to start and stop the server.

For more details on configuring the JS Test Driver server, see the [Getting Started with JsTestDriver](http://code.google.com/p/js-test-driver/wiki/GettingStarted) documentation.

1. Download the [JS Test Driver JAR](http://code.google.com/p/js-test-driver/) and save the JAR to your local system.
2. In the Services window, right-click the JS Test Driver node and choose Configure.   
   
3. In the Configure dialog box, click Browse and locate the JS Test Driver JAR that you downloaded.
4. Select the Chrome with NetBeans Connector (in NetBeans IDE 7.3, select Chrome with NetBeans JS Debugger) for the browser. Click OK.  
   

**Notes.** You only need to specify the location of the JS Test Driver JAR the first time that you configure the JS Test Driver.

The list of browsers that can be captured and used for testing is based on the browsers that are installed on your system. You can select multiple browsers as slave browsers, but to run the tests a window that can be a slave for the server must be open for each browser. The selected browsers will be captured automatically when you start the server from the IDE.

When you select Chrome with NetBeans Connector you can debug your tests that are run with JS Test Driver.

1. Right-click the project node in the Projects window and choose New > Other.
2. Select the **jsTestDriver Configuration File** in the Unit Tests category. Click Next.
3. Confirm that **jsTestDriver** is the File Name.
4. In the Created File field, confirm that the location for the file is the config folder of the project (AngularJSPhoneCat/config/jsTestDriver.conf).

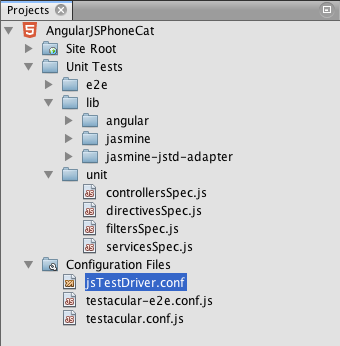
**Note.** The jsTestDriver.conf configuration file must be in the config folder of the project. If the location for the created file is not the config folder, click Browse and select AngularJSPhoneCat - Configuration Files folder in the dialog box.

1. Confirm that the checkbox for downloading the Jasmine libraries is selected. Click Finish.



**Note.**You need to download the Jasmine libraries to run jsTestDriver. If you are notified that the IDE is unable to download the Jasmine libraries, check the proxy settings of the IDE in the Options window.

When you click Finish the IDE generates a skeleton jsTestDriver.conf configuration file and opens the file in the editor. In the Projects window you can see that the configuration file is created under the Configuration Files node. If you expand the lib folder under the Unit Tests node you can see that the Jasmine libraries were added to the project.



In the editor you can see the following contents of the configuration file that are generated by default:

server: http://localhost:42442

load:

- test/lib/jasmine/jasmine.js

- test/lib/jasmine-jstd-adapter/JasmineAdapter.js

- test/unit/\*.js

exclude:

The configuration file specifies the default location of the local server that is used to run the tests. The file also lists the files that must be loaded. By default the list includes the Jasmine libraries and any JavaScript files that are in the unit folder. Tests are usually located in the unit folder but you can modify the list to specify the locations of other files that need to be loaded to run the tests. To run the unit tests you also need to add the location of the JavaScript files that you want to test and the Angular JavaScript libraries to the list of files that are loaded.

For this tutorial, if you want to run the tests using JS Test Driver you will want to add the following files (in bold) to the list of files that are loaded.

load:

- test/lib/jasmine/jasmine.js

- test/lib/jasmine-jstd-adapter/JasmineAdapter.js

**- app/lib/angular/angular.js**

**- app/lib/angular/angular-mocks.js**

**- app/lib/angular/angular-route.js**

**- app/lib/angular/angular-animate.js**

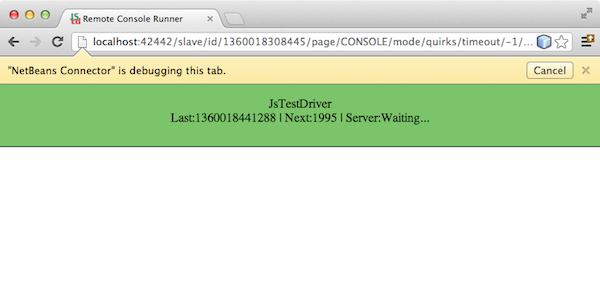
**- app/lib/angular/angular-resource.js**

**- app/js/\*.js**

- test/unit/\*.js

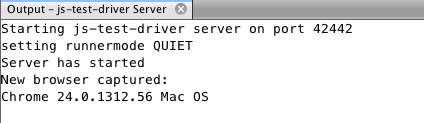
1. After you update the configuration file you can right-click the project node in the Projects window and choose Test.

When you click Test the IDE automatically opens the JS Test runner in the Chrome browser and two tabs in the Output window.



The Chrome browser window displays a message when the jsTestDriver server is running. You can see that the server is running on localhost:42442. In the js-test-driver Server tab in the Output window you can see the status of the server.

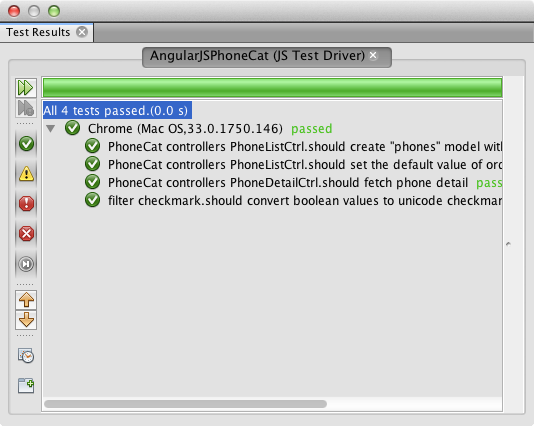
Note that the JsTestDriver is running in a browser tab and that the NetBeans Connector is debugging the tab. You can debug your unit tests if you run tests with JS Test Driver and select Chrome with NetBeans Connector as one of the target browsers.



**Note.** The browser window must be open and the jsTestDriver server must be running to run the unit tests. You can start the server and open the window by right-clicking the JS Test Driver node in the Services window and choosing Start.



1. Choose Window > Output > Test Results in the main menu to open the Test Results window and see the results of the tests.



You can click the green check icon in the left margin of the window to view the the expanded list of the tests that passed.

## Debugging a JS Unit Test with JS Test Driver

This exercise demonstrates how you can use the IDE and JS Test Driver to debug your unit tests.

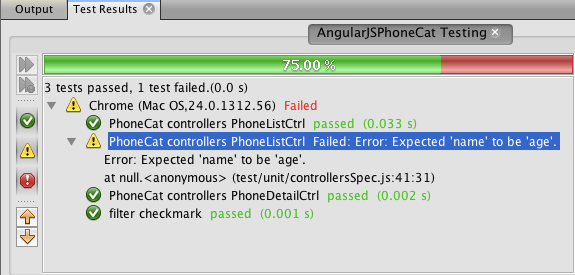
**Note.** NetBeans IDE 8.0 does not support debugging tests that are run with the Karma test runner.

1. Expand the js folder in the Projects window and double-click controllers.js to open the file in the editor.
2. Modify line 7 in the file to make the following changes (in **bold**). Save your changes.
3. function PhoneListCtrl($scope, Phone) {
4. $scope.phones = Phone.query();
5. $scope.orderProp = '**name**';

}

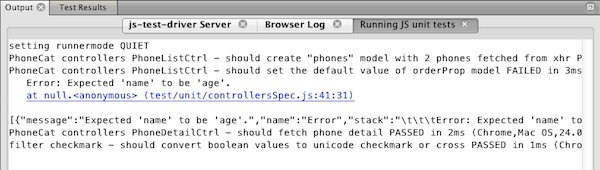
When you save your changes the page automatically reloads in the browser. You can see that the order of the phones in the list changed.

1. Confirm that the JS Test Driver server is running and that the status message is visible in the Chrome browser window.
2. Right-click the project node in the Projects window and choose Test.



When you run the test you can see that one of the tests failed with the message that the value "name" was encountered instead of the expected value "age".

1. Open the Running JS unit tests tab in the Output window.



You can see in the message that the orderProp is expected to be age on line 41.

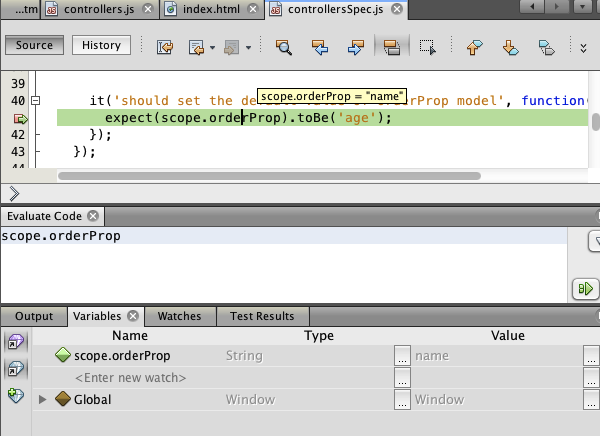
1. Click the link in the Running JS unit tests tab to navigate to the line in the test that failed. The test file controllersSpec.js opens in the editor at line 41 (in **bold**)
2. it('should set the default value of orderProp model', function() {
3. **expect(scope.orderProp).toBe('age');**

});

You can see that the test expected "age" as the value of scopeOrder.prop.

1. Set a breakpoint at the line where the test failed (line 41).
2. Right-click the project node in the Projects window and choose Test.

When you run the test again the program counter hits the breakpoint. If you hover your cursor over scopeOrder.prop you can see in the tooltip that the value of the variable is "name" when the breakpoint is hit.



Alternatively, you can choose Debug > Evaluate Expression in the main menu to open the Evaluate Code window. If you type the expression scopeOrder.prop in the window and click the Evaluate Code Fragment button (Evaluate Expression button)(Ctrl-Enter) the debugger displays the value of the expression in the Variables window.

1. Click Continue in the toolbar to finish running the test.

## Summary

In this tutorial you learned how the IDE provides tools that can help you debug and run unit tests on JavaScript files. Debugging is automatically enabled for HTML5 applications when you run the application in the Chrome browser and the NetBeans Connector extension is enabled. The IDE also enables you to easily configure and run unit tests on JavaScript files using the Jasmine testing framework and the JS Test Driver server.

[Send Feedback on This Tutorial](https://netbeans.org/about/contact_form.html?to=3&subject=Feedback:%20Debugging%20and%20Testing%20JavaScript%20in%20HTML5%20Applications)

## See Also

For more information about support for HTML5 applications in the IDE on [netbeans.org](https://netbeans.org/), see the following resources:

* [Getting Started with HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html). A document that demonstrates how to install the NetBeans Connector extension for Chrome and creating and running a simple HTML5 application.
* [Working with CSS Style Sheets in HTML5 Applications](https://netbeans.org/kb/docs/webclient/html5-editing-css.html). A document that demonstrates how to use some of the CSS wizards and windows in the IDE and how to use the Inspect mode in the Chrome browser to visually locate elements in your project sources.
* [Developing HTML5 Applications](http://docs.oracle.com/cd/E50453_01/doc.80/e50452/dev_html_apps.htm) chapter in the [Developing Applications with NetBeans IDE User's Guide](http://www.oracle.com/pls/topic/lookup?ctx=nb8000&id=NBDAG)

For more information about running unit tests using JS Test Driver, refer to the following documentation:

* JS Test Driver Project Page: <http://code.google.com/p/js-test-driver/>
* Jasmine Home Page: <http://pivotal.github.com/jasmine/>
* [Intro to JsTestDriver](http://transitioning.to/2012/07/magnum-ci-the-jenkins-chronicles-1-intro-to-jstestdriver/). An introduction to using JsTestDriver with a continuous integration server.