



The most powerful way
to develop
Web applications!



CodeCharge Studio 4
**Quick Start
Tutorials**



8275 S. Eastern Ave. Suite 200 Las Vegas, NV 89123 USA
Telephone: + 1 (888) 241-7338 Fax: + 1 (866) 312-8049
info@yessoftware.com
<http://www.yessoftware.com>

Copyright © 2000-2008 YesSoftware, Inc.

All rights reserved.

This manual, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. The content of this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by YesSoftware, Inc. YesSoftware, Inc assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation. Except as permitted by such license, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of YesSoftware, Inc.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to YesSoftware Inc., 6330 S. Eastern Ave. #5, Las Vegas, NV 89119 USA. YesSoftware, the YesSoftware logo, CodeCharge and the CodeCharge Logo are either registered trademarks or trademarks of YesSoftware, Inc in the United States and/or other countries. ASP, IIS, Microsoft, Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. JavaScript and JSP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other trademarks are the property of their respective owners.

Quick Start Tutorials	6
Creating a Simple Hello World Application	6
Creating a New CodeCharge Studio Project	6
Specifying Project Properties	6
Modifying the Page Properties	7
Adding a Label to the Page	8
Configuring Label Properties	8
Adding an Action to the Label	9
Previewing the Code	10
Publishing the Project	11
Testing the Application	12
Creating an Employee Directory	13
Step 1	13
Creating a New Project	13
Create a Blank Project	13
Save the Newly Created Project	14
Step 2	14
Open Project Settings	14
Specify the General Project Settings	15
Enter the Publishing Settings	16
Create Database Connection(s) for the Project	17
Setup Security Settings for the Project	18
Configure Security Groups for the Project	19
Step 3	20
Build the Design Connection	20
Specify the Data Provider (JET, ODBC, etc.)	21
Specify Connection Parameters (Database Filename)	22
Test the Database Connection	23
Complete the Build Process of the Design Connection	24
Setup the Server Connection	25
Save Project Settings	26
Step 4	27
Launch the Grid Builder	27
Launch the Visual Query Builder	27
Specify Database Fields in the Visual Query Builder	28
Select Database Fields for the Grid Data Source	29
Setup the Search Form to be used with the Grid	29
Define Grid Sorting and Navigation	30
Select a Style for the Grid	31
Preview the Grid	32
Save the Project	33
Step 5	33
Launch the Authentication Builder	33
Run the Authentication Builder	34
Specify Login Form Options	34
Select a Style for the Login Form	35
Specify the Login Page for the Project	35
Restrict Page Access	36
Step 6	37
Rename the Page	37
Change the Size of the Search Fields	37
Create a ListBox Field	38

Configure the ListBox Field.....	38
Change Field Captions	39
Setup Search Parameters	40
Publish the Page	41
Preview and Test the Project.....	42
Conclusion	42
Creating a Task Management System with the Application Builder.....	43
Step 1.....	43
Create a New Project.....	43
Launch the Application Builder	44
Specify Project Properties	45
Select Database Connection	46
Configure the Application Builder	47
Setup Site Security and Authentication.....	48
Select Database Tables.....	48
Configure Site Pages.....	49
Specify Site Layout and Menu	50
Select Site Style	50
Review Pages and Create the Site	51
Step 2.....	51
Open the Task List Page	51
Test the Page	52
Implicit Relationships	53
Delete Unneeded Columns.....	54
Change Field Caption.....	55
Synchronize HTML and Programming Code	55
View and Test the Live Page	56
Step 3.....	56
Add ListBox Search for Project Names.....	57
Add ListBox Search - Insert a ListBox Control.....	57
Add ListBox Search - Set ListBox Properties.....	57
Add ListBox Search - Move Table Row	58
Filter Grid Records - Select "Where" Property.....	59
Filter Grid Records - Add Search Parameter	60
Filter Grid Records - Set AND Operator	61
View the Working Page	62
Login to the System.....	62
Access Record Maintenance Page.....	63
Step 4.....	63
Changing Field Labels	63
Create Label Fields	64
Rearrange Label Fields	65
Preview the Task Maintenance Page	66
Enhancing Application Functionality with Programming Events.....	67
ASP and VBScript	67
Step 1.....	67
Step 2.....	70
Step 3.....	73
Step 4.....	76
Step 5.....	80
PHP	82
Step 1.....	82

Step 2.....	85
Step 3.....	89
Step 4.....	91
Step 5.....	95
Perl.....	97
Step 1.....	97
Step 2.....	100
Step 3.....	103
Step 4.....	106
Step 5.....	110
CFML.....	112
Step 1.....	112
Step 2.....	115
Step 3.....	118
Step 4.....	121
Step 5.....	124
JSP.....	126
Step 1.....	126
Step 2.....	129
Step 3.....	133
Step 4.....	136
Step 5.....	139
C# and VB.Net.....	142
Step 1.....	142
Step 2.....	145
Step 3.....	149
Step 4.....	152
Step 5.....	158
Conclusion.....	161

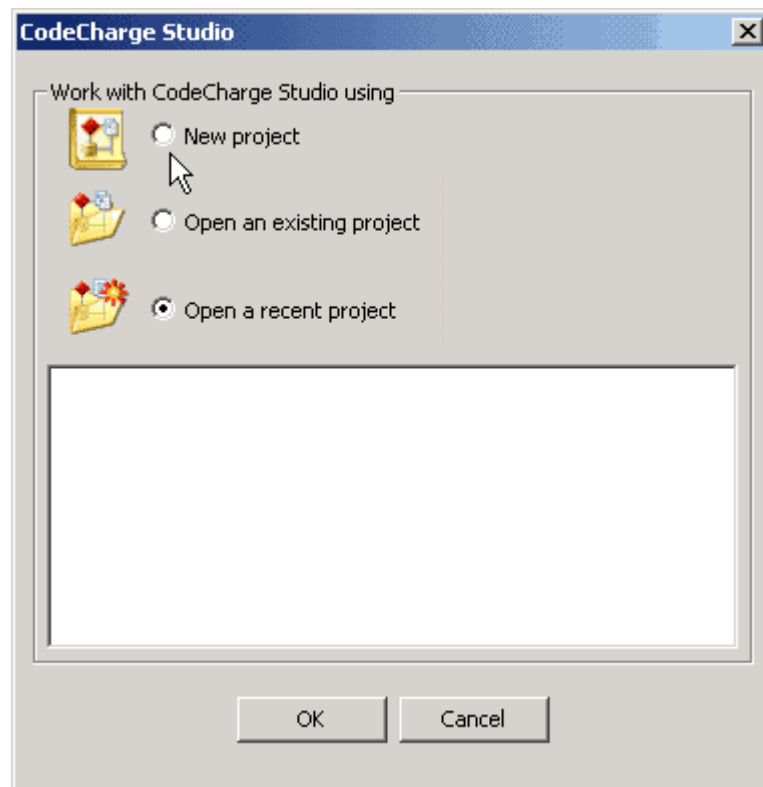
Quick Start Tutorials

Creating a Simple Hello World Application

Creating a New CodeCharge Studio Project

The creation of any web application using CodeCharge Studio inevitably requires that we first create a new project.

1. Start CodeCharge Studio
2. In the dialog window that appears, select the **New Project** option. If you are already within CodeCharge Studio, you can use the **File => New... => Project...** menu option.



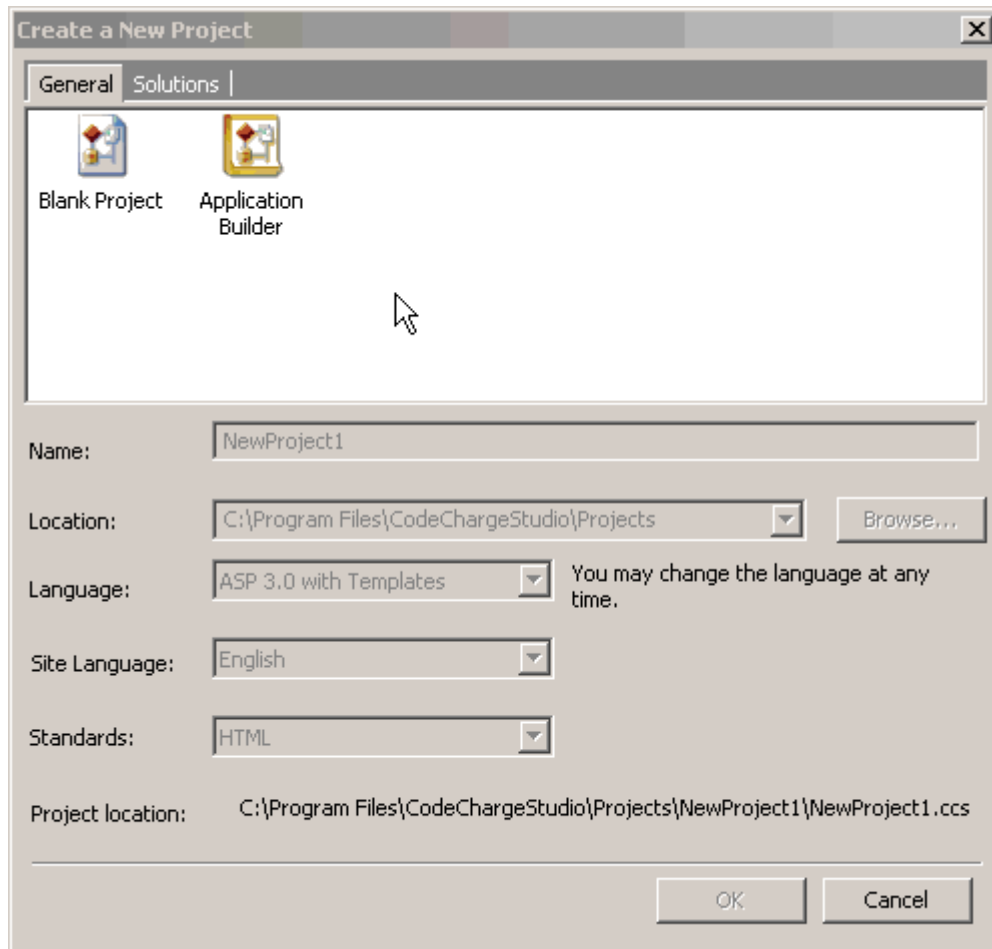
Next: [Specifying Project Properties](#)

Specifying Project Properties

When the **Add New Project** dialog appears, you should then proceed to specify properties for the new project.

1. Under the **General** tab, make sure that the **Blank Project** option is selected.
2. Enter *HelloWorldProject* in the **Name** field.
3. The **Location** field contains the file system path where the project files will be stored and you can leave this value as-is unless you want to save the files to a different location.

4. For the **Language** field, select the programming language in which you want to publish the project.
5. Specify the human language that the application should use by default in the **Site Language** field.
6. After specifying all the properties, click on the **OK** button so that CodeCharge Studio can create the project framework.



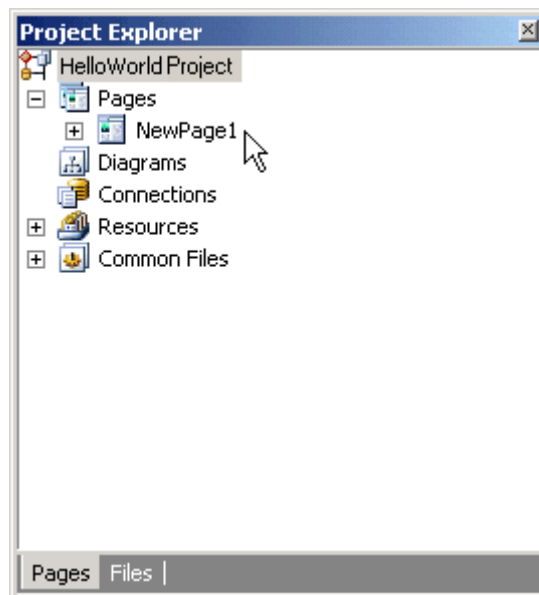
Next: [Modifying the Page Properties](#)

Modifying the Page Properties

CodeCharge Studio creates a project which has a single empty page called *NewPage1*. Our first task within the project will be to rename the page and assign it a more meaningful name.

To do so:

1. Right-click on the page within the **Project Explorer** window and select the **Rename** option or press F2.
2. Type in *HelloWorld* as the new name for the page.
3. After changing the page name, a dialog may appear asking whether to "Update all links to this page?". Simply click **Yes** although in this case we don't have other pages that might have links to the page that was renamed.



Next: [Adding a Label to the Page](#)

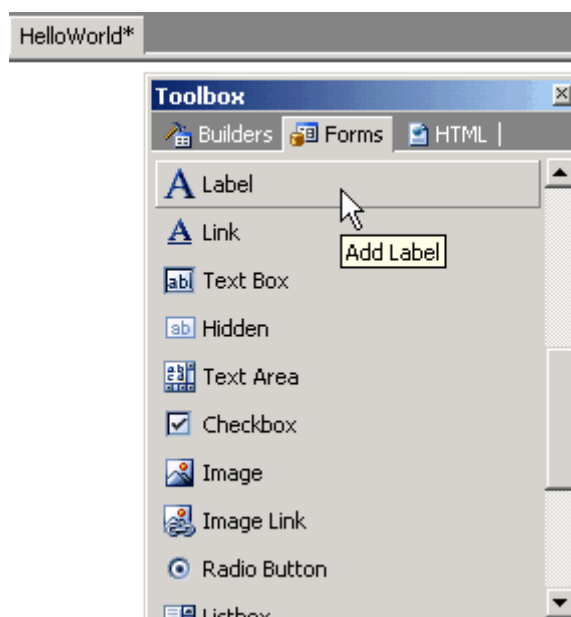
Adding a Label to the Page

A Label is usually used to dynamically display a text content in pre-designated place on the page. The main difference between the plain text and a label is that the label can be controlled programmatically, thus it can display the text determined by the program or retrieved from the database. A label is also the simplest of Controls.

We shall now proceed to add the Label control to the page. In doing this, we shall mainly work in **Design** mode although it is also possible to perform the same tasks from within **HTML** mode.

To add the Label control to the page:

1. Click on the **Label** button in the **Forms** tab of the **Toolbox**.



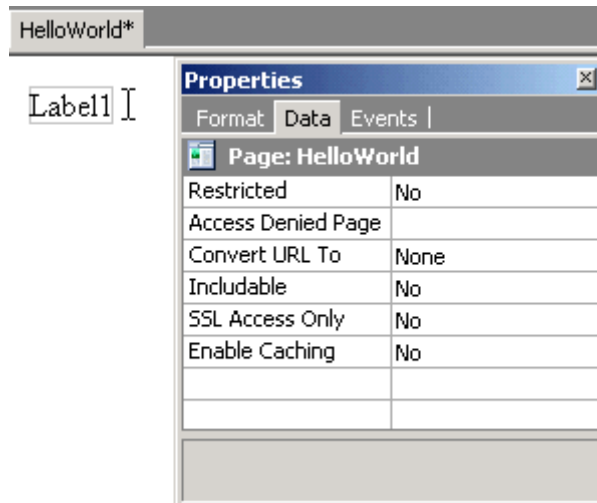
Next: [Configuring Label Properties](#)

Configuring Label Properties

By default the name of the Label added on the previous step is *Label1*, though occasionally you may want to rename your controls to reflect its purpose, refer to a database field name, etc.

To modify the name of control:

1. While remaining in the **Design** mode right-click on *{Label1}* and select the Properties option from the popup menu that appears. The Properties window switches to the Data tab and displays the properties for the Label.
2. Set the **Name** property to *MyLabel*.



Next: [Adding an Action to the Label](#)

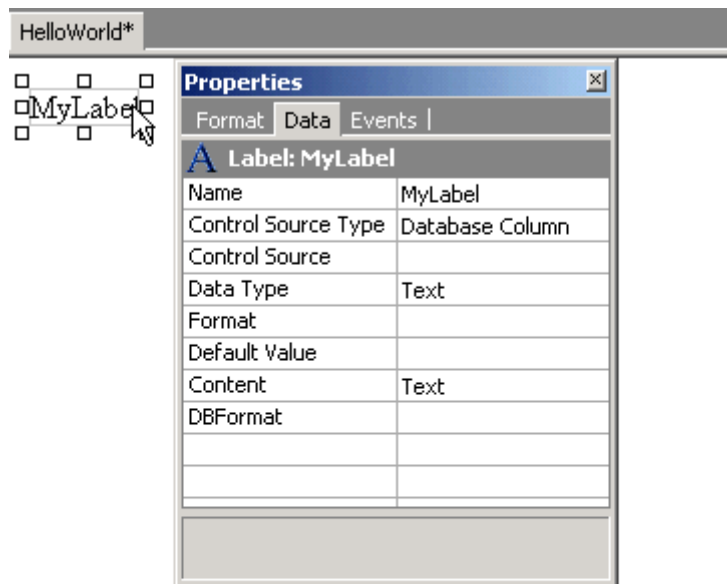
Adding an Action to the Label

Now that you created the label control, it is time to assign a value that will be shown by this label. To do so, we will need to use some programming code that assigns specific text to the label's value.

Luckily in CodeCharge Studio you can utilize "actions" to automatically generate several lines of code without programming. Actions are usually used for small tasks, such as sending an email, validating data entry, reading or saving a cookie, etc. In this example we will use the "Retrieve Value for Control" action to set the label's value to the plain text "Hello World!".

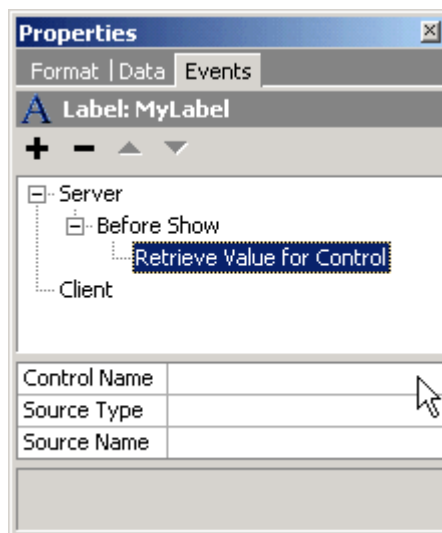
To assign *Hello World!* to the Label use the **Before Show** event using the **Retrieve Value for Control** action:

1. Switch to **Events** tab of the **Properties**.
2. Right-click on the **Before Show** and select the **Add Action...** option.
3. Select **Retrieve Value for Control** from the list.
4. Click **Ok**.



To configure the action properties:

1. Select **Retrieve Value for Control** action in **Events** tab.
2. Set *MyLabel* in the **Control Name** property.
3. Select *Expression* in the **Source Type** property.
4. Set *"Hello World!"* in the **Source Name** property.



Next: [Previewing the Code](#)

Previewing the Code

Since the actions automatically generate snippets of the programming code, you can instantly see the result by viewing the generated code event.

To see the result of **Retrieve Value for Control** action:

1. Right-click on the **Retrieve Value for Control** in the **Events** tab of the **Properties** and select the **Show Code...** option.



In the opened **Code** mode of the editor you can see the code that is used to assign the *Hello World!* value to the *MyLabel* control.

Note: You can also modify the code generated by the action, or add the code manually.

To add the code manually without using the action - right-click on the **Before Show** event and select **Add Code...** option from the menu.

Next: [Publishing the Project](#)

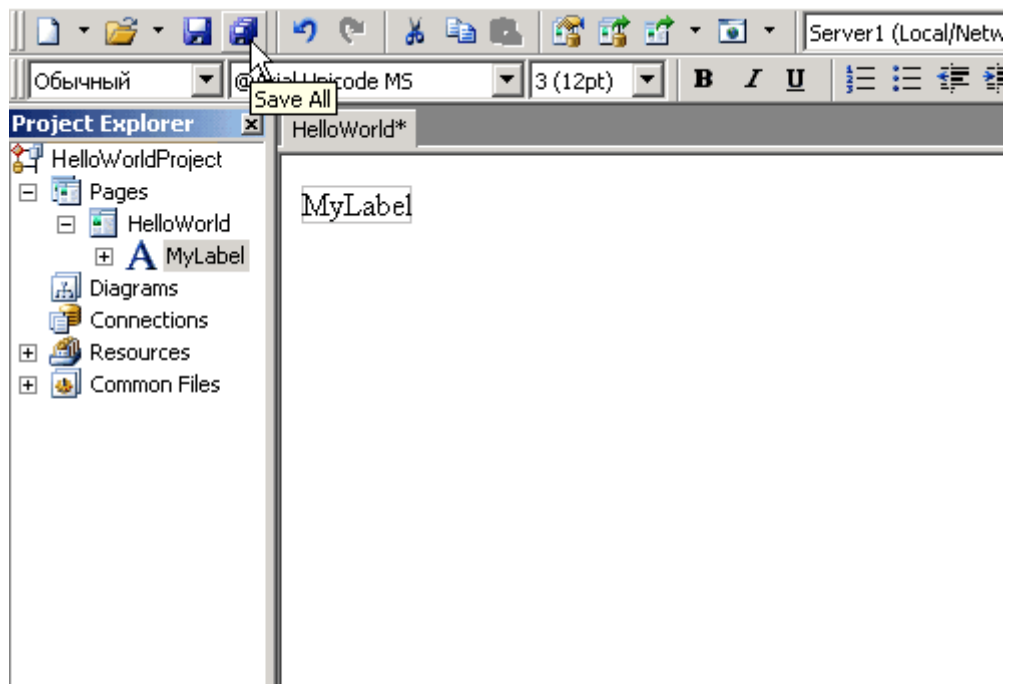
Publishing the Project

All the components of our simple Hello World web application are now in place.

1. Click on the **Save All** button to save the project
2. Press the **F9** key to begin the publishing process.
3. When the **Project Settings** dialog appears, confirm the entries made in the **Active Server**, **Server Path** and **Server URL** fields.

The **Server Path** should be a location in the web server where the pages will be published and the **Server URL** should be a URL which maps to the **Server Path**.

4. Once you are satisfied with the settings, click **Ok** to publish the project.

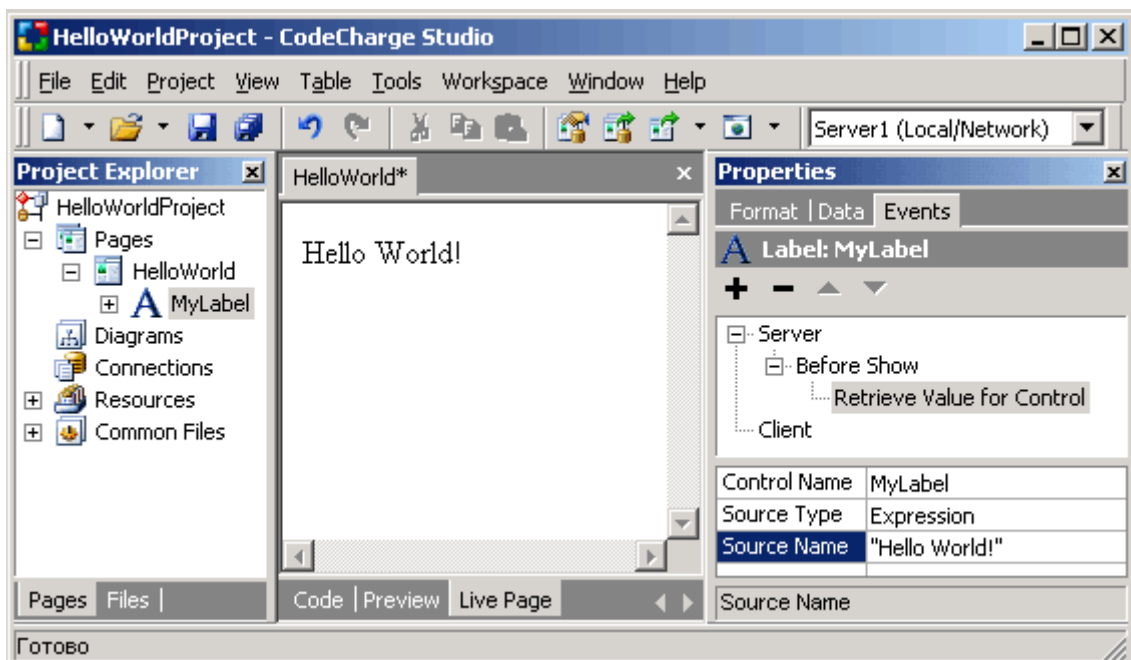


Next: [Testing the Application](#)

Testing the Application

1. After the publishing process is complete, click on the **Live Page** tab of the document window to view the generated page.
2. You should see *Hello World!* text.

That brings us to the end of this exercise. The Hello World Project application is indeed very simple but hopefully it has set the stage for more complex projects.



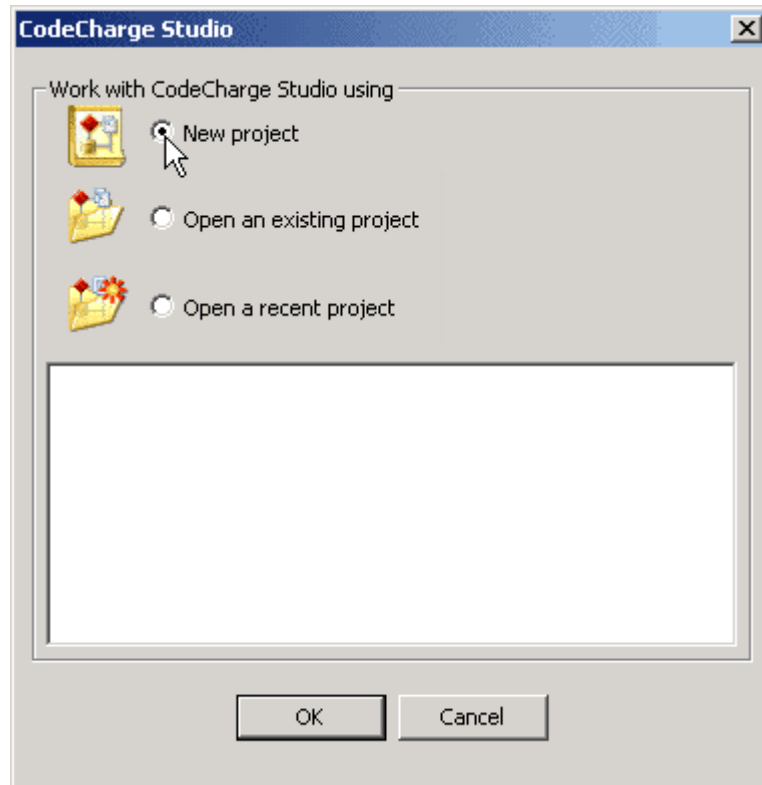
Back to: [Quick Start Tutorials](#)

Creating an Employee Directory

Step 1

Creating a New Project

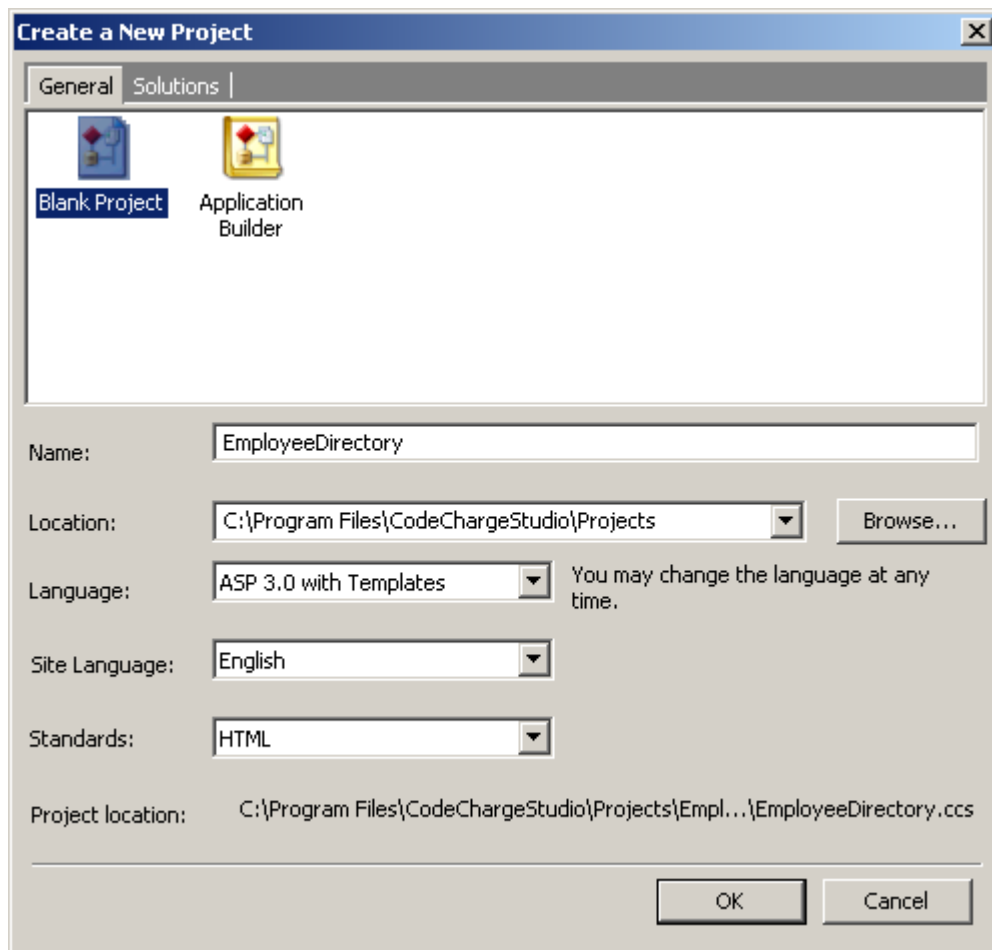
Start CodeCharge Studio and select **New project** on the initial screen.



Next: [Create a Blank Project](#)

Create a Blank Project

1. In the **Add New Project** window, set the **Name** property to *EmployeeDirectory*.
2. Also enter the **Location** where the project should be saved on the disk
3. Click the **OK** button to confirm and create a new project.

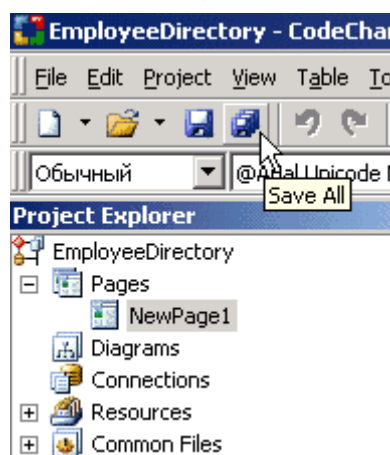


Next: [Save the Newly Created Project](#)

Save the Newly Created Project

At any time,

- you can click on the **Save All** icon on the toolbar to save your project,
- or press **CTL+S** to save the current page.

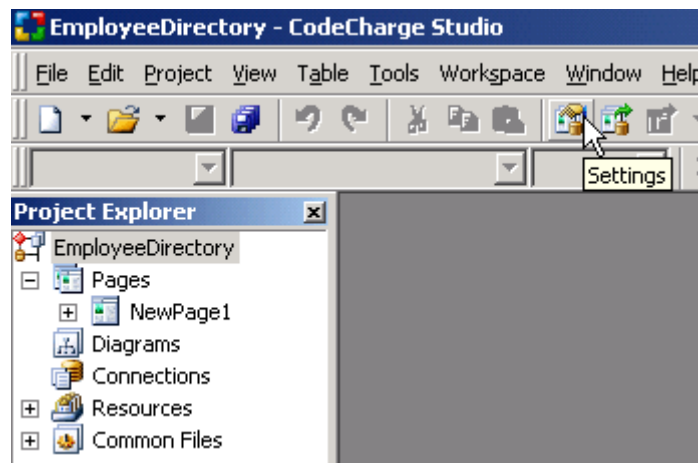


Next: [Specifying Project Settings](#)

Step 2

Open Project Settings

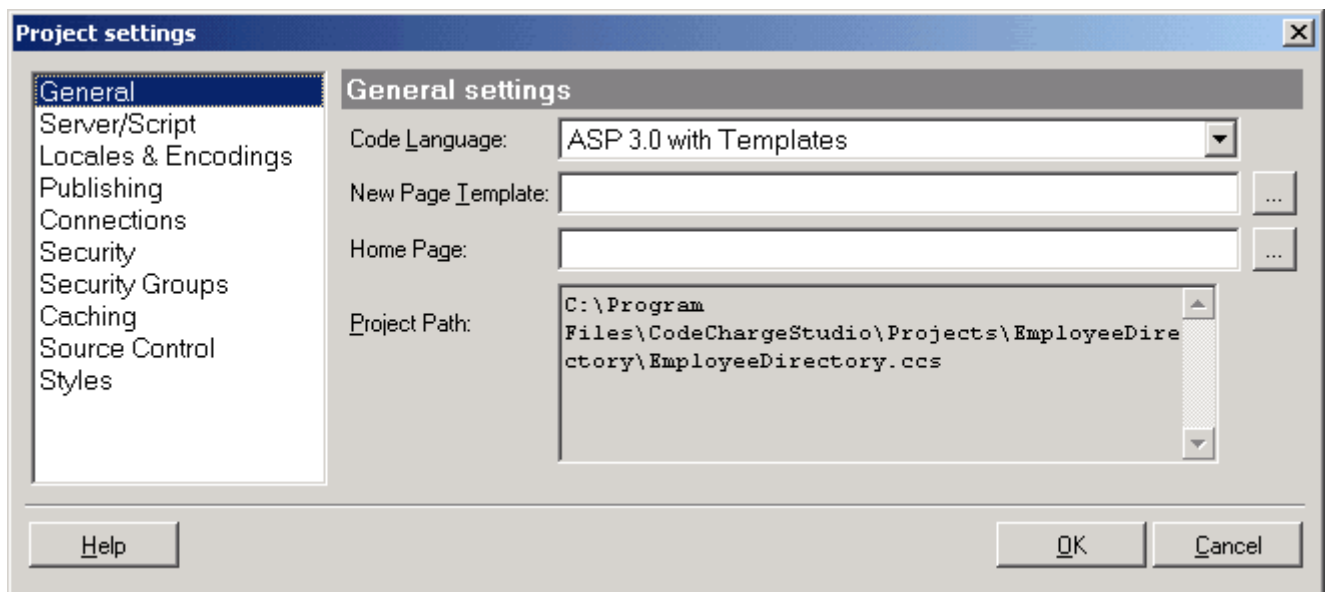
- Click on the **Settings** icon in the Toolbar.
- Or select **Project => Settings...** on the main menu bar.
- You can also right-click on the Project Name (*EmployeeDirectory*) in the **Project Explorer** window and select the **Settings...** option.



Next: [Specify the General Project Settings](#)

Specify the General Project Settings

Specify the general project properties, such as Programming Language and Date Display Format.



- **Code Language**

The *Code Language* specifies the type of the programming code that you will generate from your project to create the web application. The currently available programming languages are:

- *ASP 3.0 with Templates*: Generates ASP 3.0 (VBScript) programs that use separate .html files as templates during run-time.
- *ASP.Net C#*: Generates .aspx files with C# code.
- *ASP.Net VB*: Generates .aspx files with VB code.
- *CFML 4.0.1/MX*: Generates ColdFusion 4.0.1 code.
- *CFML 4.0.1/MX with Templates*: Generates ColdFusion 4.0.1 code (.cfm) and separate .html template files.

- *JSP 1.1 JDK 1.3*: Generates JSP 1.1 code.
- *PERL 5.0 with Templates*: Generates PERL 5.0 code and separate .html template files.
- *PHP4/PHP5 with Templates*: Generates PHP code (.php) and separate .html template files.
- *Servlets 2.2 JDK 1.3 with Templates*: Generates Java Servlet code that utilizes .html templates.

- **Home Page**

The *Home Page* specifies the main page of your web application, which you can later launch with the **F7** key or from the menu with *Project | Home Page*. The specified home page will then be generated, published, and opened within CodeCharge Studio in *Live Page* mode.

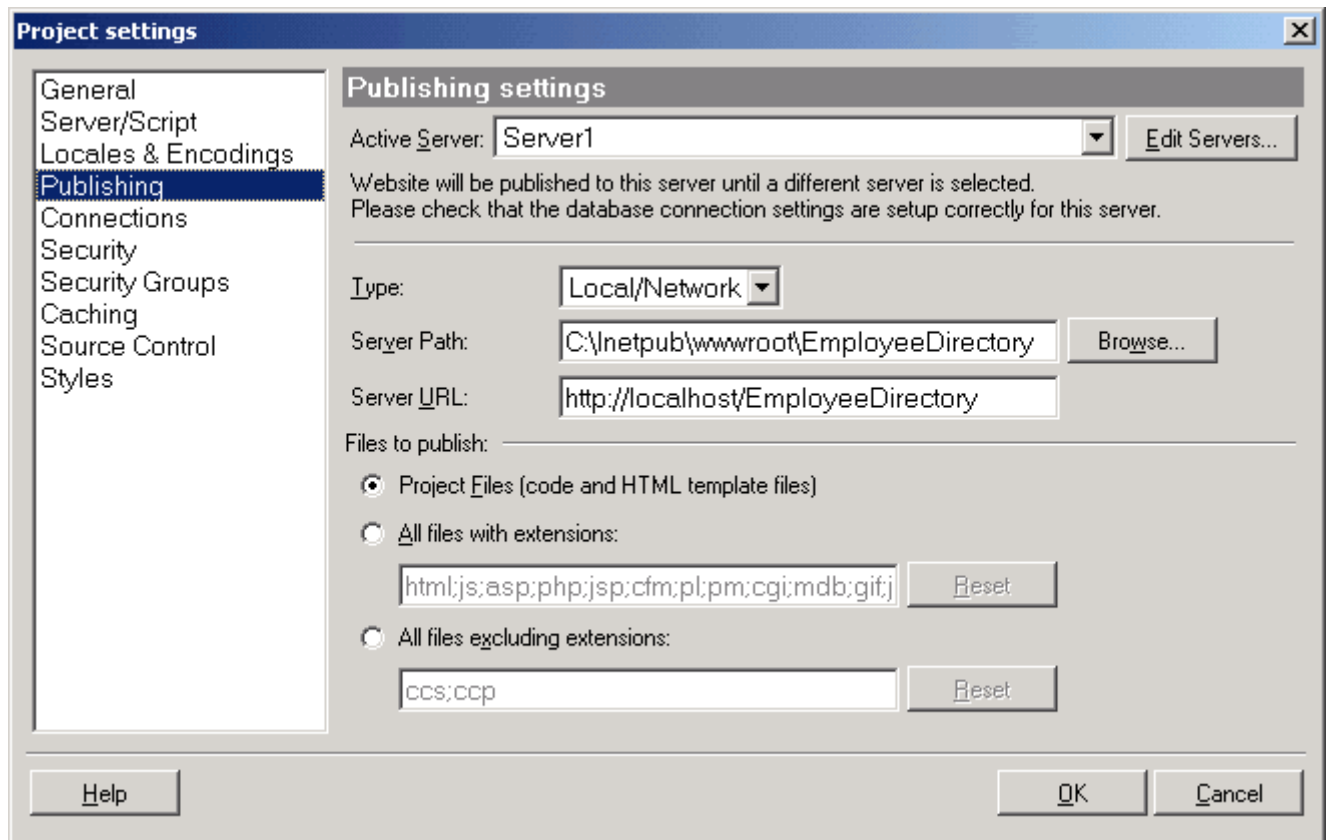
- **Project Path**

The *Project Path* specifies the current location of the project and its file name. This is for informational purposes and cannot be edited from this dialog.

Next: [Enter the Publishing Settings](#)

Enter the Publishing Settings

In the **Publishing** section, specify the server and folder where CodeCharge Studio should output generated files during the publishing process.



- **Active Server:**

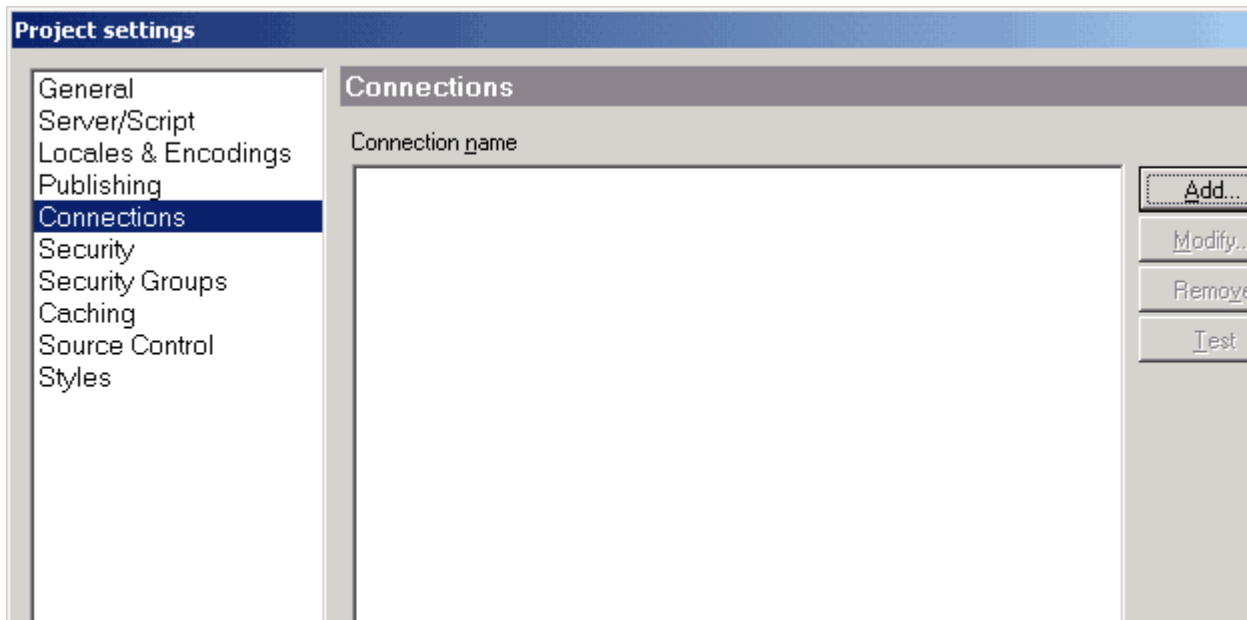
Specify the server the code should be copied to after the project is generated.

- **Type:**
The location can be either a local or network drive, or an Ftp address on an external server.
- **Server Path:**
The full path where generated files should be published.
- **Server URL:**
The web address corresponding to the **Server Path**. This URL will be used to view the pages in **Live Page** mode.
- **Files to publish:**
Leave the default option unless you want to publish specific files or exclude others.

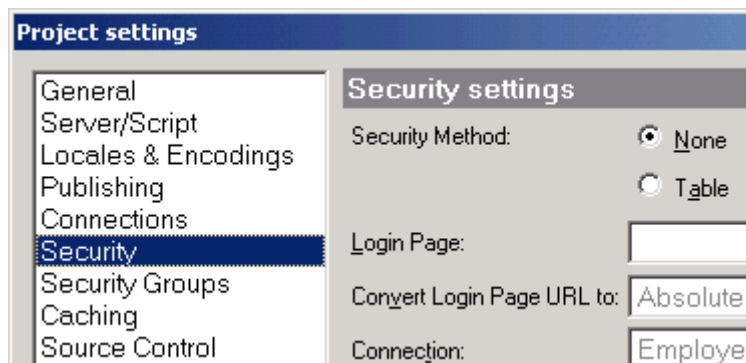
Next: [Create Database Connection\(s\) for the Project](#)

Create Database Connection(s) for the Project

1. Click on the **Connections** section to setup a new database connection.
2. Click **Add...** and follow the steps described in the [Create Database Connection](#) section to complete creating a database connection(s).



3. Once completed building the connection, click on the **Security** tab.



Next: [Setup Security Settings for the Project](#)

Setup Security Settings for the Project

Security settings allow you to protect specific pages from unauthorized access by directing unauthorized users to a Login page.

- If you are just starting with CodeCharge Studio, skip this step. Click the **OK** button to complete configuring the Project Settings.
- If you are ready to configure your site security, enter the appropriate information as shown.
 - **Security Method:**
Specify whether the project will have Table based security or no security at all. If you select Table based security, then you also have to specify the table and fields that will be used.
 - **Login Page:**
The page to which users will be redirected if they are not logged in or their access permissions are insufficient to access a page within your site. This page must be created before you can start using the authentication features.
 - **Connection:**
Database Connection that contains user login information.
 - **User Table:**
The table that contains user and login information.
 - **User ID Field:**
The key field in the user table, which will be used as the user's unique id.
 - **Login Field:**
The field in the user table that contains the user's login name.
 - **Password Field:**
The field in the user table that contains the user's password.
 - **Level/Group Field:**
The field containing user's security level or group, which will be used to verify access authorization.

The screenshot shows the 'Project settings' dialog box with the 'Security settings' tab selected. On the left is a navigation pane with options: General, Server/Script, Locales & Encodings, Publishing, Connections, Security (highlighted), Security Groups, Caching, Source Control, and Styles. The main area contains the following settings:

Security Method:	<input type="radio"/> None	
	<input checked="" type="radio"/> Table	
Login Page:	<input type="text"/>	...
Convert Login Page URL to:	None	▼
Connection:	Employees	▼ Refresh
User Table:	employees	▼
User ID Field:	emp_id	▼
Login Field:	emp_login	▼
Password Field:	emp_password	▼
Level/Group Field:	group_id	▼

At the bottom left of the main area is an 'Advanced...' button.

In addition, levels or groups should be configured under the **Security Groups** tab.

Note: Please enable the [Cookies](#) in your web browser to make the Security work properly.

Next: [Configure Security Groups for the Project](#)

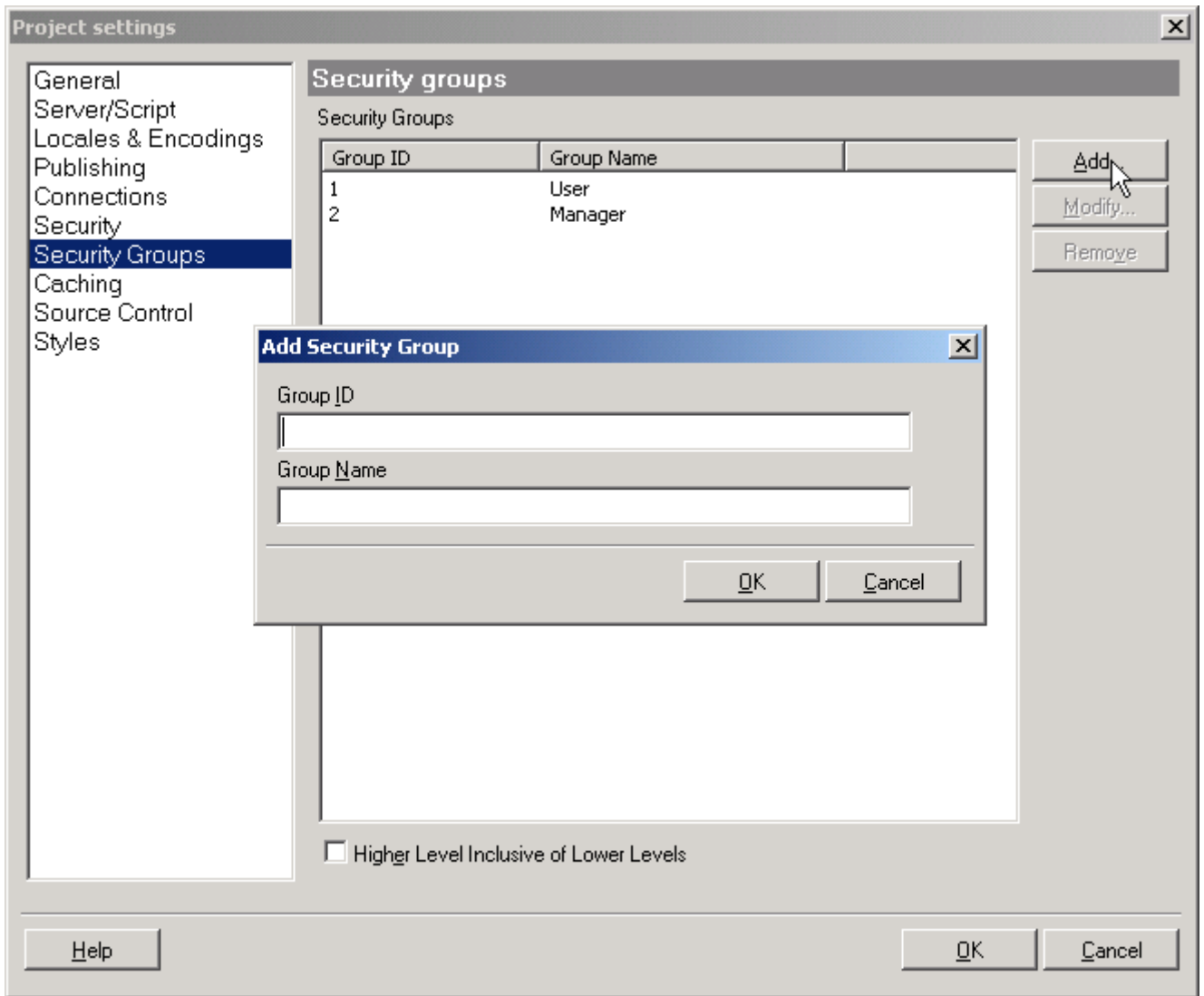
Configure Security Groups for the Project

1. Go to the **Security Groups** section in the **Project Settings** dialog to configure the security groups.
2. Click the **Add...** button to create security levels or groups that will be used for page authentication.

The security groups specified here usually should match levels or groups in the table specified under the [Security tab](#). However, you can also configure additional groups that will be available in the future, or you can configure groups that exist in other tables or are programmatically assigned.

When later restricting page access, CodeCharge Studio will allow you to select any of the groups configured in this screen.

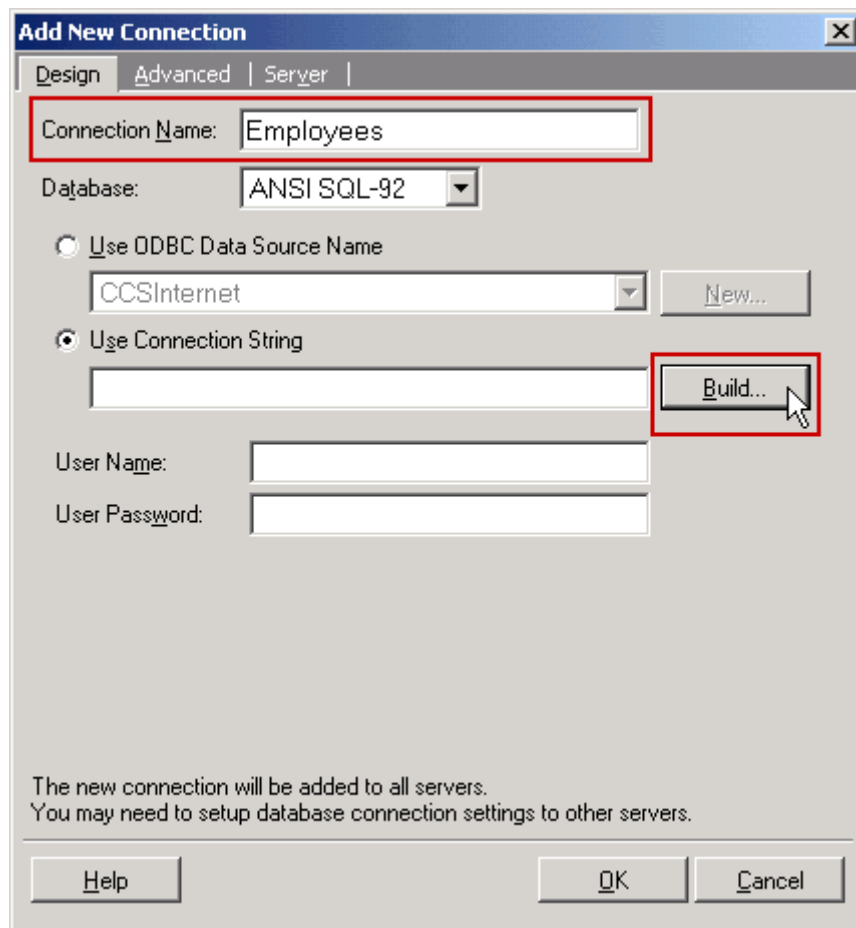
3. If you use numbers as your Group IDs, you can select the option **Higher Level Inclusive of Lower Levels**, which will cause the generated programs to assume that users with a higher security level can access pages with lower security levels. For example, users with level 4 will be able to access pages with level 3 but not 5.



Next: [Creating a Grid using the Grid Builder](#)

Step 3

Build the Design Connection



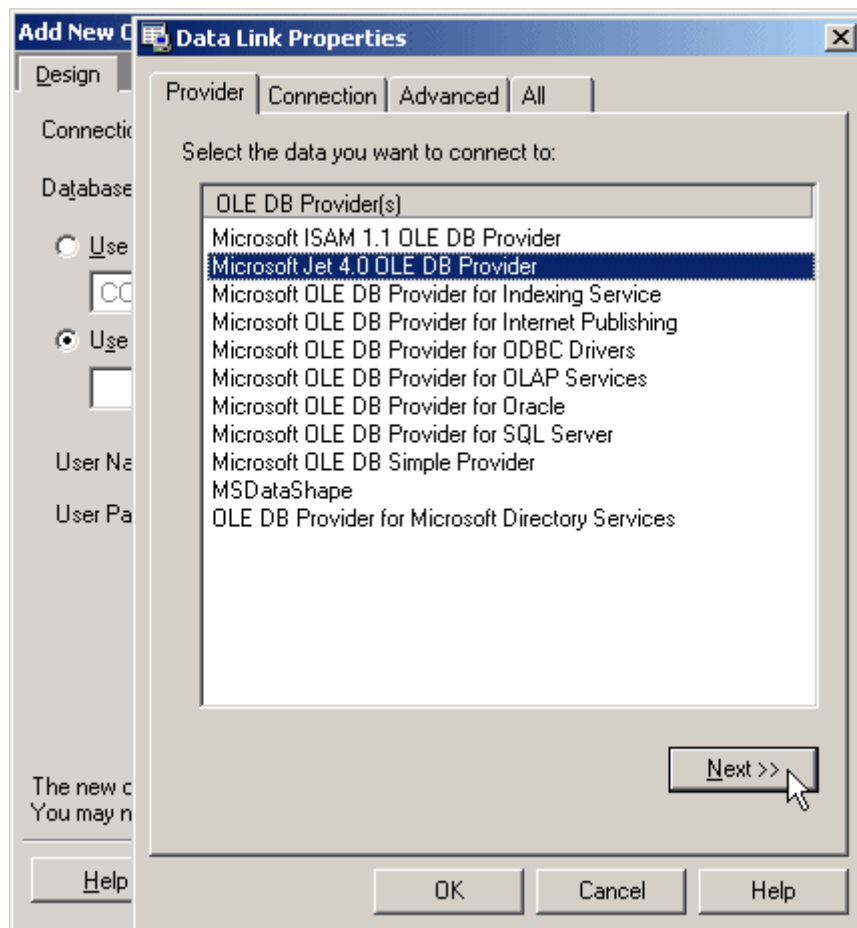
1. In the **Connection Name** field, enter a name for your connection (*Employees*)
2. Click **Build...** to specify connection parameters.

The **Design** Connection is the database connection utilized by CodeCharge Studio and will allow you to select database tables and fields in various places during the project building process. This connection can be different from the **Server** Connection, which is used by the generated programs.

Next: [Specify the Data Provider \(JET, ODBC, etc.\)](#)

Specify the Data Provider (JET, ODBC, etc.)

1. In the **Data Link Properties** window, select *Microsoft Jet 4.0 OLE DB Provider*.
This will allow you to connect directly to a database file, such as MS Access .mdb.
2. You can also select ODBC or other specialized drivers to connect to a variety of other databases. The list of options varies depending on the selection of drivers installed on your computer.



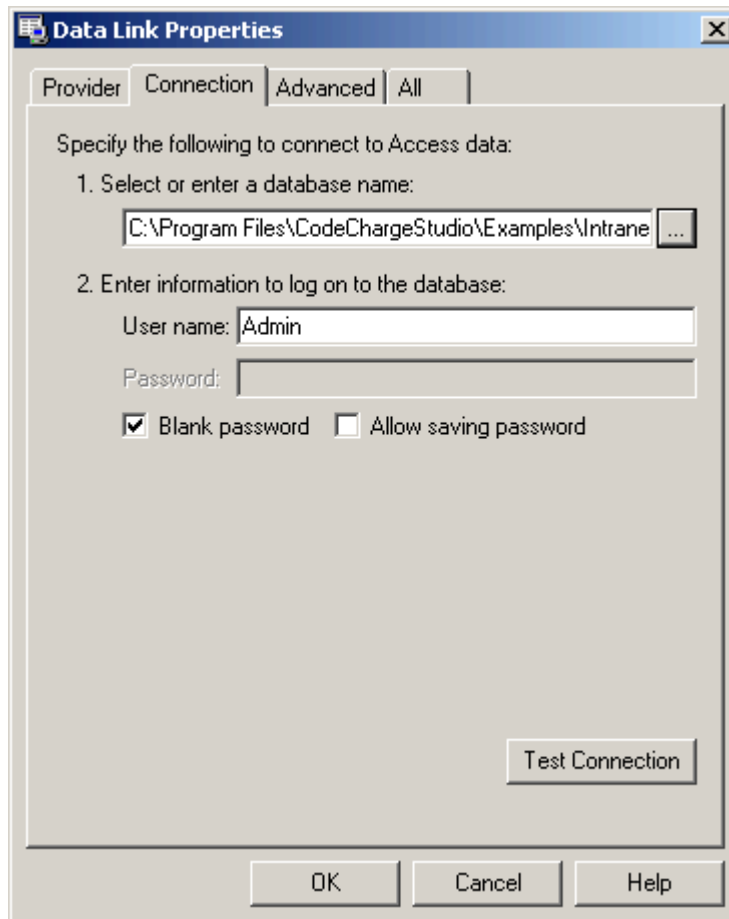
Next: [Specify Connection Parameters \(Database Filename\)](#)

Specify Connection Parameters (Database Filename)

1. If using the JET provider, select the database file to be used for this connection.

In this case, we will use the *intranet.mdb* database located in the *examples/intranet* folder of your CodeCharge Studio installation.

2. Copy this file to your project folder then connect to the copied file.
3. If using ODBC or other provider, select the DSN (Data Source Name) or other parameters needed to establish the database connection.

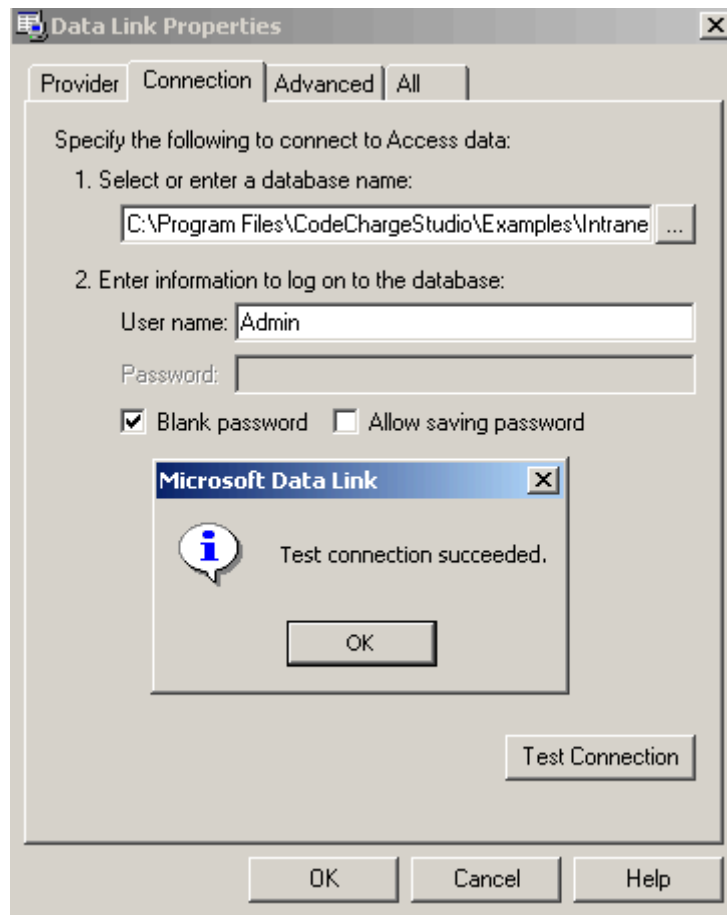


Next: [Test the Database Connection](#)

Test the Database Connection

Once you have selected the database,

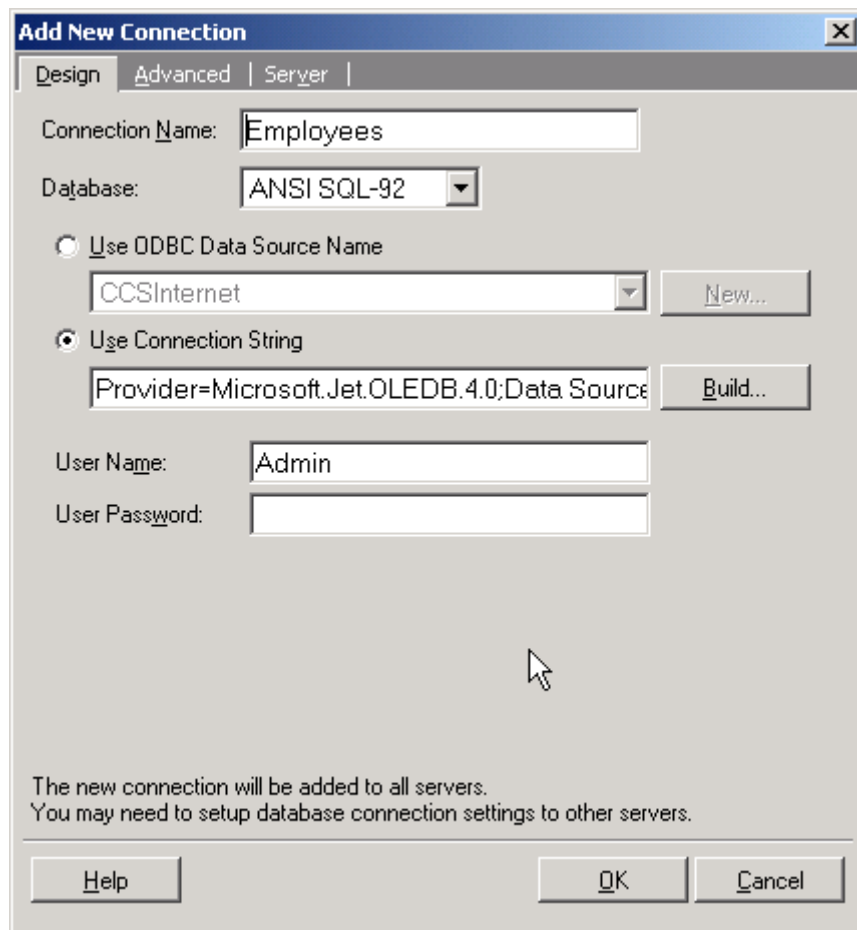
1. Click **Test Connection** to check if the connection to the database is working correctly.



Next: [Complete the Build Process of the Design Connection](#)

Complete the Build Process of the Design Connection

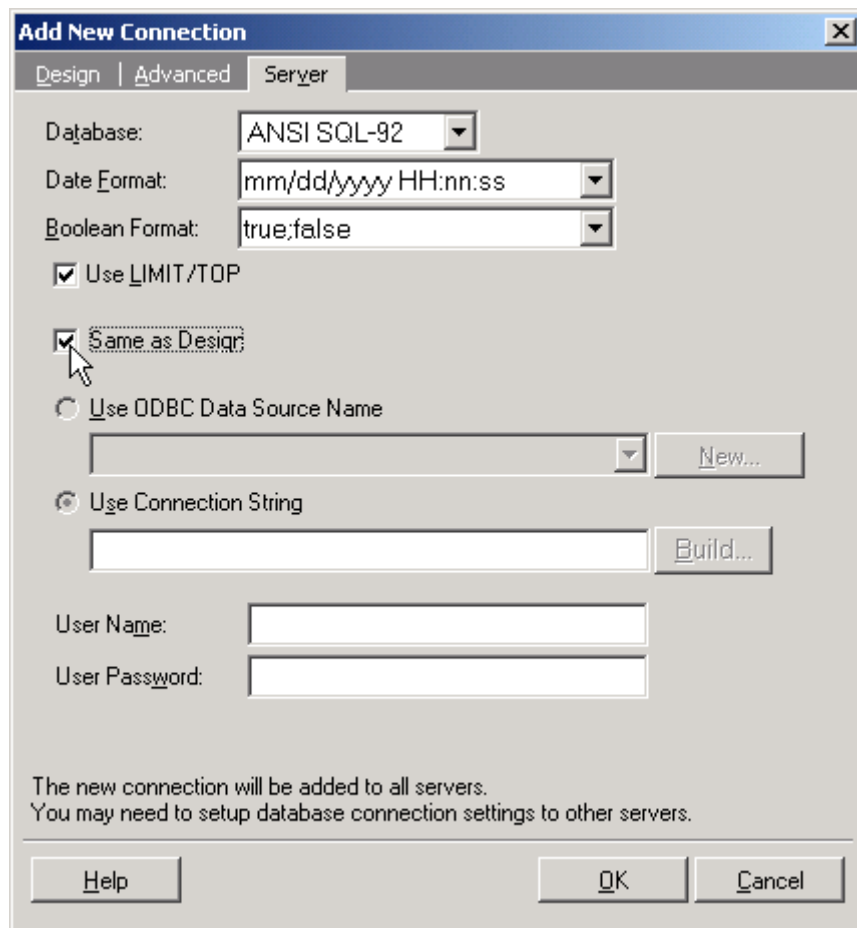
1. Confirm that the connection string was created in the **Use Connection String** field.
2. Click on the **Server** tab to create the server connection.



Next: [Setup the Server Connection](#)

Setup the Server Connection

Specify that the Server connection is the **Same as Design** connection. The Server Connection is the database connection utilized by the generated pages to retrieve and update the data. This connection can be different from the Design Connection that is used by the CodeCharge Studio GUI.



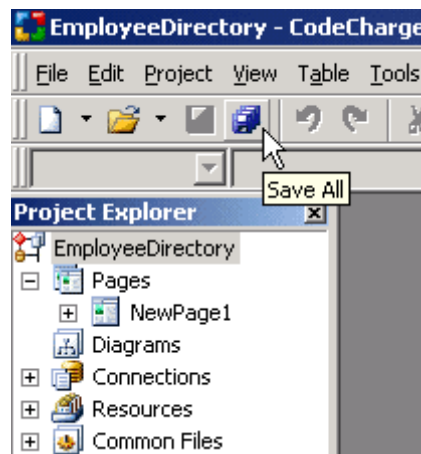
Note:

- Instead of selecting **Same as Design**, you can build a separate connection string if you are publishing the project to an external server, or if you want to use a separate database for website testing on your local machine.
- This screen may look different depending on the programming language you use.

Next: [Save Project Settings](#)

Save Project Settings

Click the **Save All** icon on the toolbar to save your project.

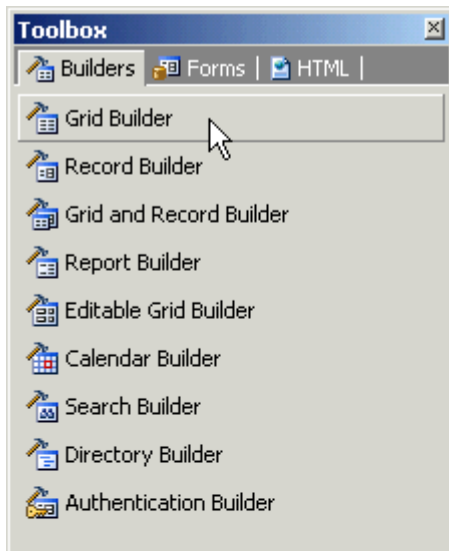


Back to: [Create Database Connection\(s\) for the Project](#)

Step 4

Launch the Grid Builder

1. Click the **Grid Builder** icon under the **Builders** tab of the **Toolbox** to start the Grid Builder.
2. In the first step of the Grid Builder dialog, make sure that *Employees* is selected as the database connection to be used.
3. Click **Next** to proceed.

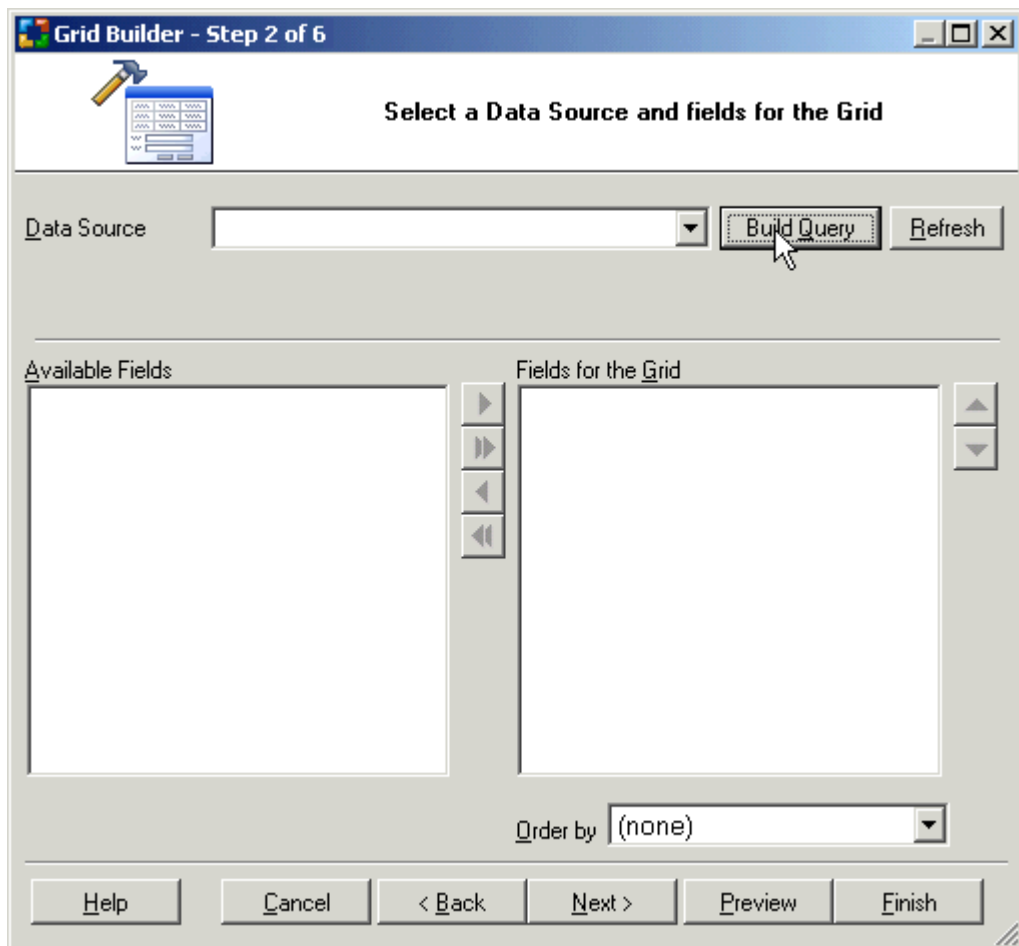


Next: [Launch the Visual Query Builder](#)

Launch the Visual Query Builder

To aid you in the process of selecting database tables and fields to be used in the grid, CodeCharge Studio has a Visual Query Builder.

1. Click on the **Build Query** button to access it.
2. Select the following tables to be used as the data source for the grid:
 - o *departments*
 - o *employees*
3. Click **Add** to place the tables in the Query Builder.

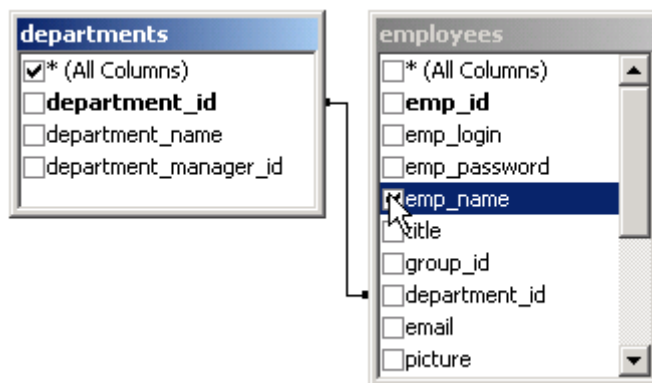


Next: [Specify Database Fields in the Visual Query Builder](#)

Specify Database Fields in the Visual Query Builder

Set the checkboxes next to all database fields that you would like to include in your grid. For this tutorial, select the following fields:

- *emp_name*
- *title*
- *phone_home*
- *phone_work*
- *email*
- *department_name* (in *departments* table)



Next: [Select Database Fields for the Grid Data Source](#)

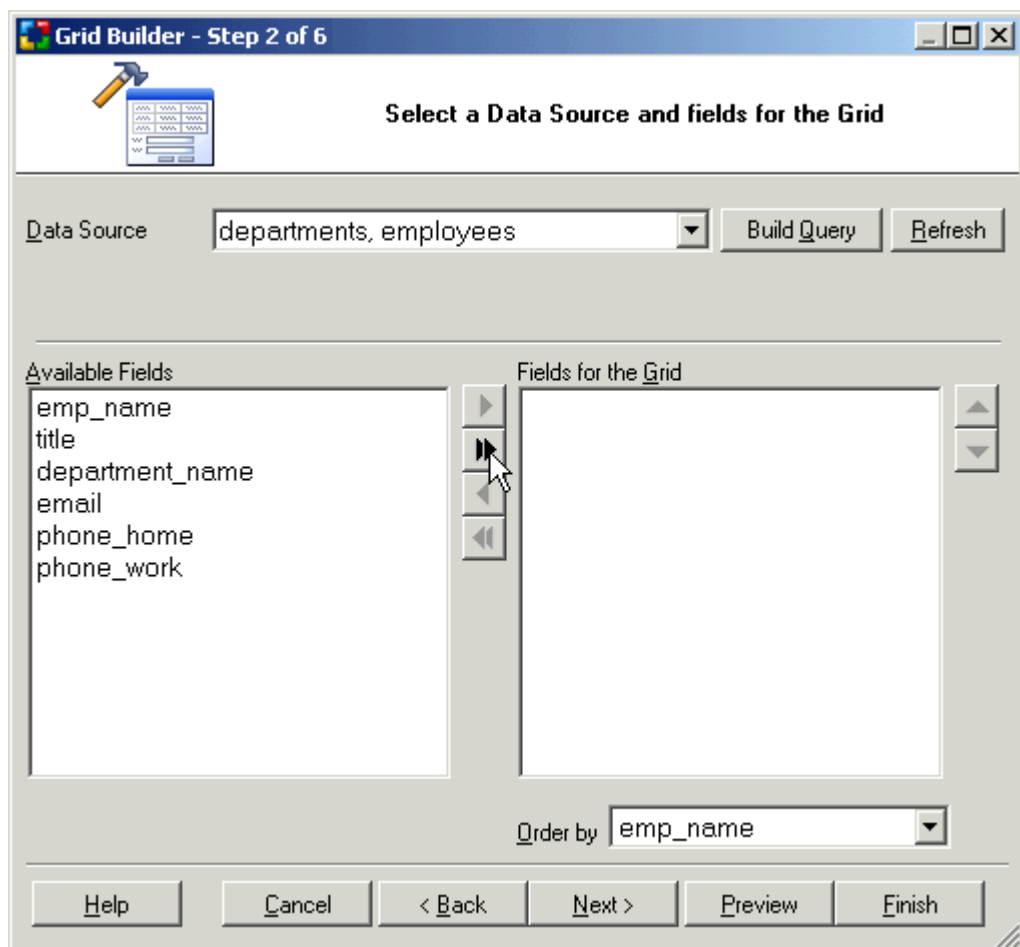
Select Database Fields for the Grid Data Source

Once finished using the Visual Query Builder, the Grid Builder will display all database fields available for inclusion in the grid.

1. Click on the double right arrow (>>) to include all fields into the grid.
2. Click on the up and down arrows to move fields.
3. Specify the order in which they will appear in the grid.
4. Click on the **Order by** drop-down menu and select the field that will be used as the basis of the sort order for the grid.

For example, if you select the field *emp_name*, by default the grid records will be sorted by employee name.

5. Click **Next** when finished.

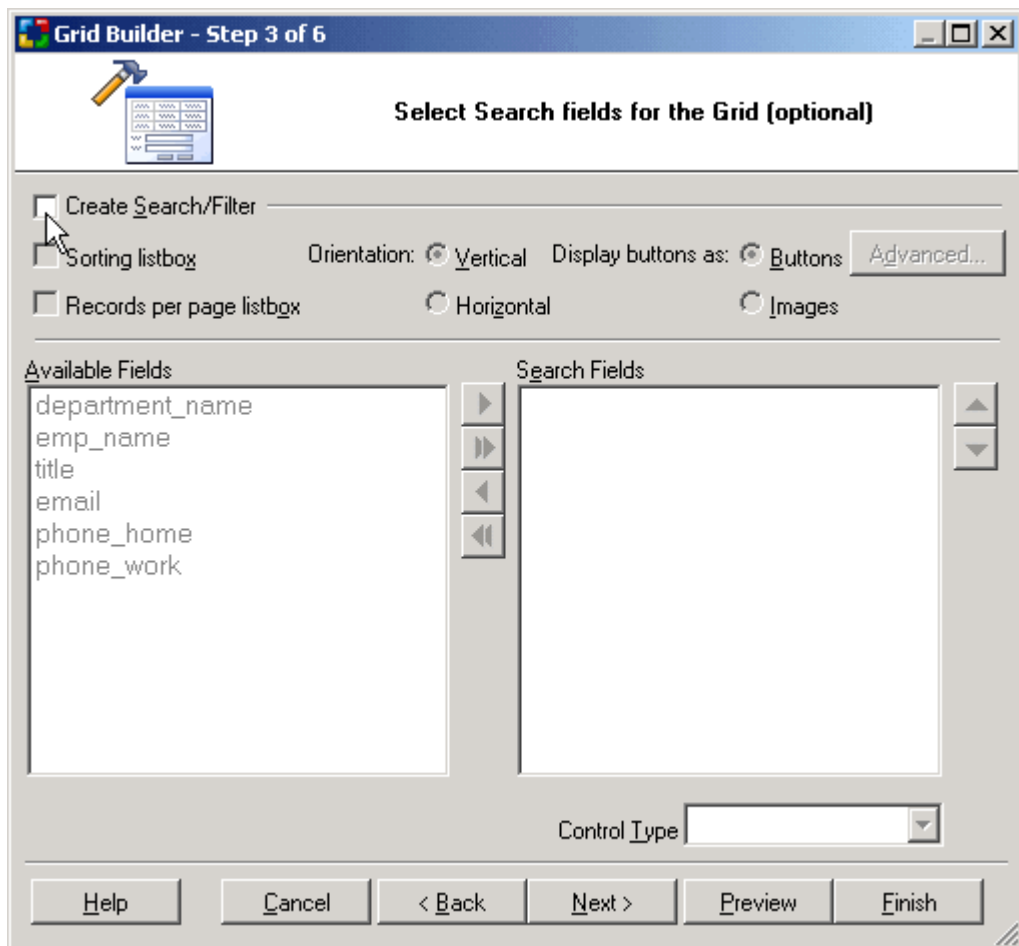


Next: [Setup the Search Form to be used with the Grid](#)

Setup the Search Form to be used with the Grid

To make the grid searchable we will now add a Search form to the page.

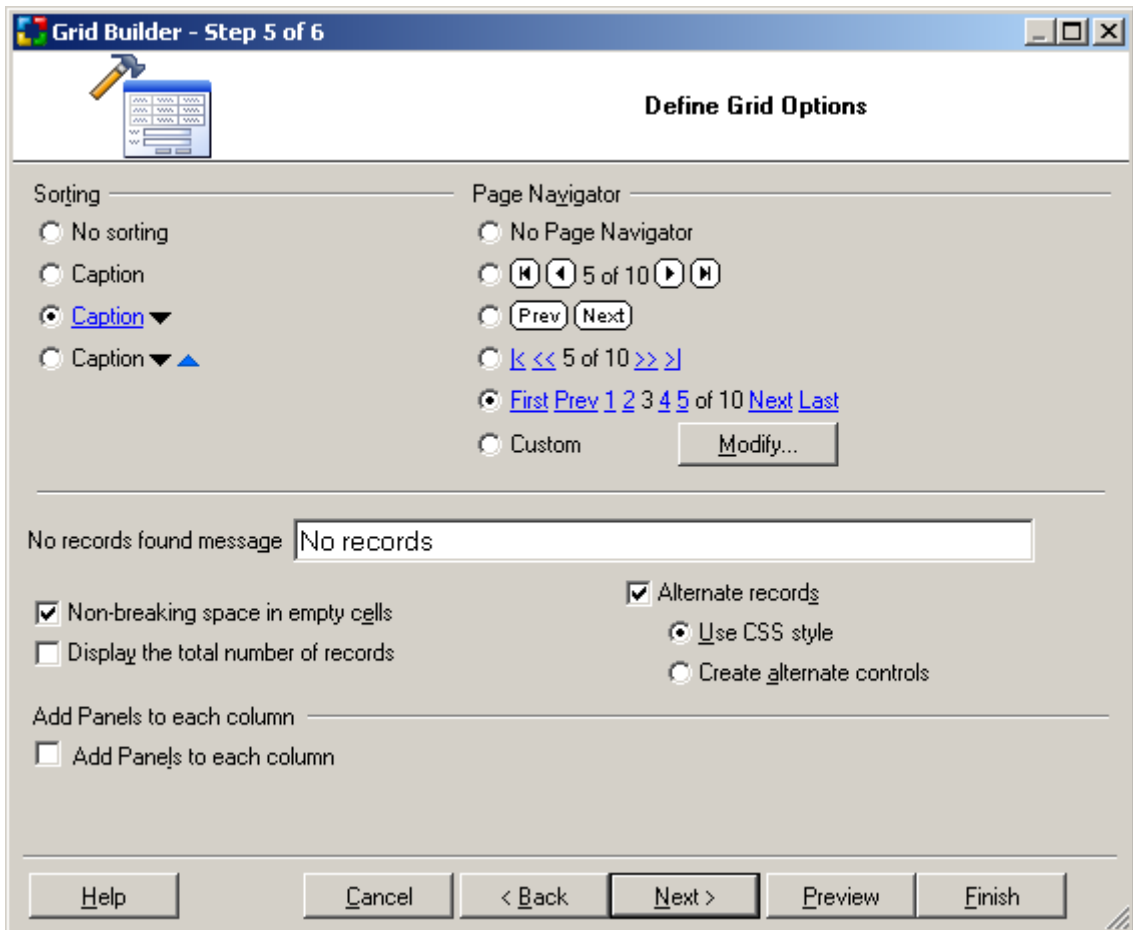
1. Activate the **Create Search/Filter** checkbox
2. Specify fields to be included in it.
3. Click **Next** when finished configuring the screen as shown above.



Next: [Define Grid Sorting and Navigation](#)

Define Grid Sorting and Navigation

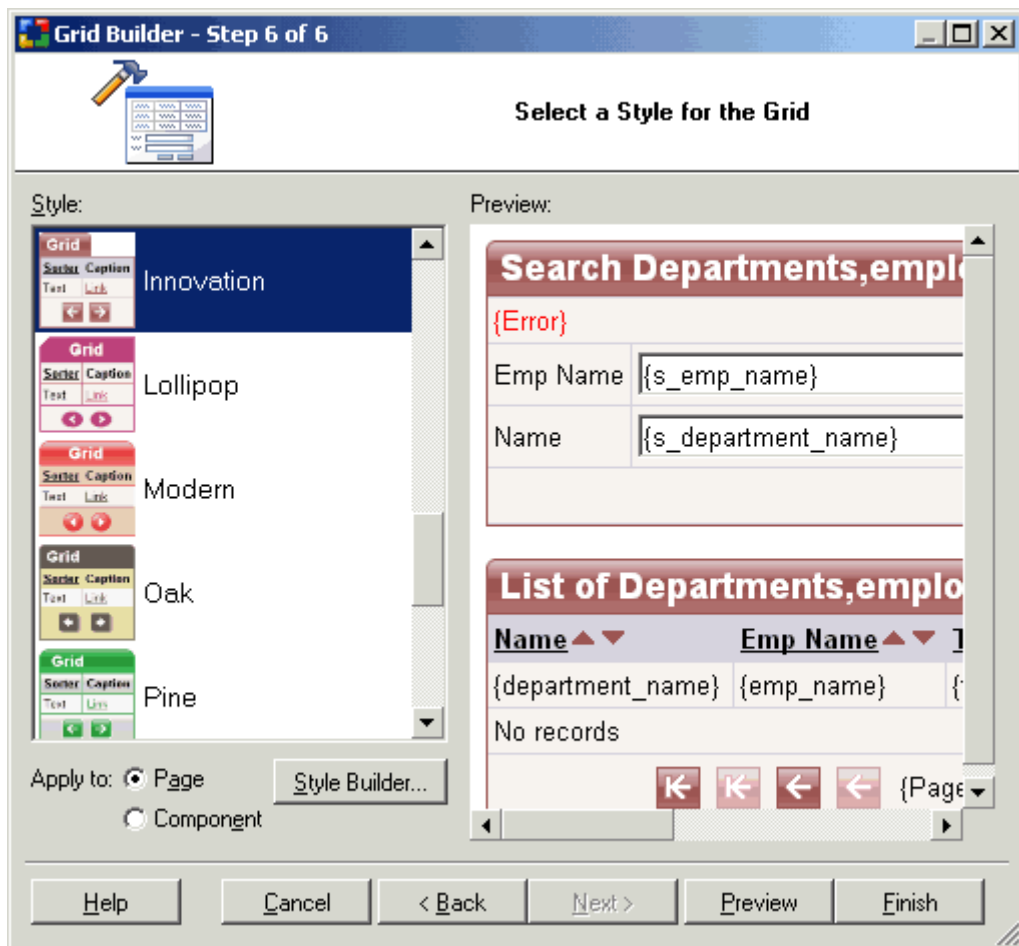
1. Specify if users can sort the Grid by clicking on the column headings and if users can navigate the grid by clicking on page numbers or *First/Last/Previous/Next* page links.
2. You can also specify if sorting and navigation should be represented by graphical icons or plain text.
3. Specify the number of grid rows to be shown on a page, as well as the message to be displayed when no records are found.
4. Additionally, specify if the builder should add a panel to each column so that it can be hidden programmatically.



Next: [Select a Style for the Grid](#)

Select a Style for the Grid

Select one of the available styles for the grid.



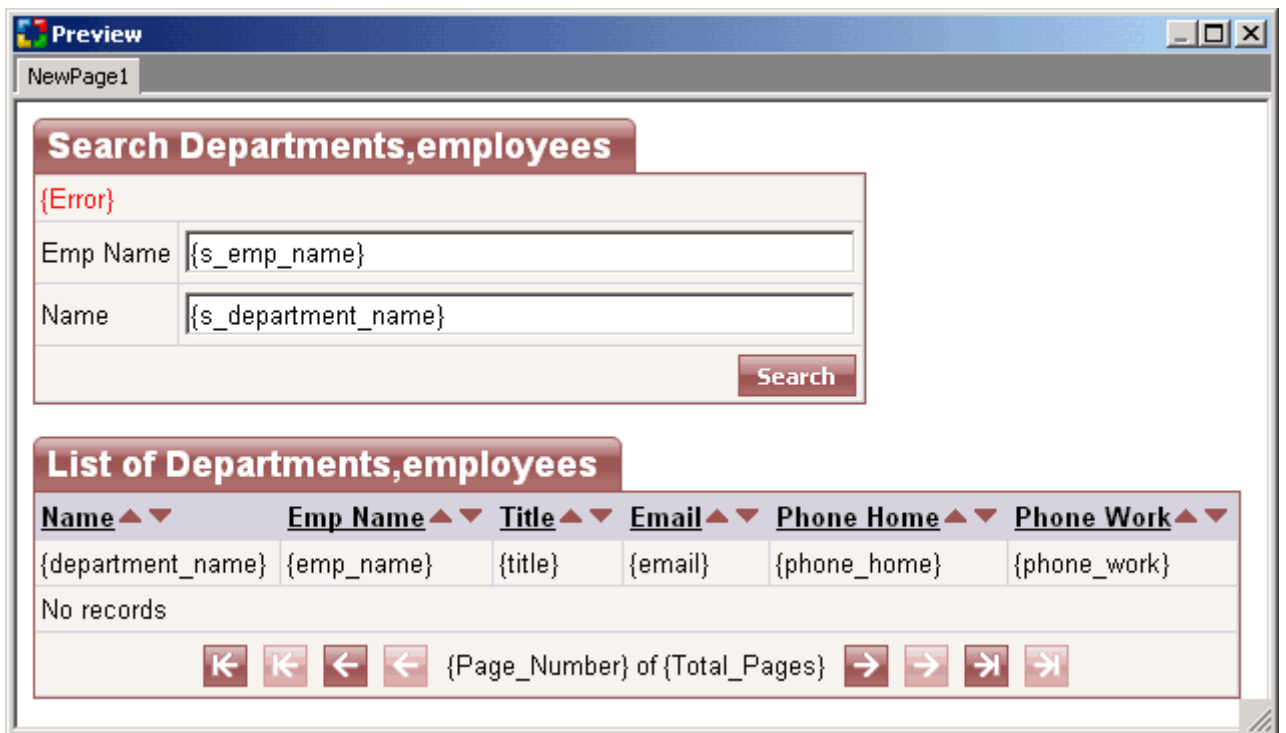
Next: [Next: Preview the Grid](#)

Preview the Grid

1. Click the **Preview** button to see the draft HTML generated by the Grid Builder.
2. Close the **Preview** dialog.
3. Click on the **Finish** button.

The next window will display summary information about the grid that will be built.

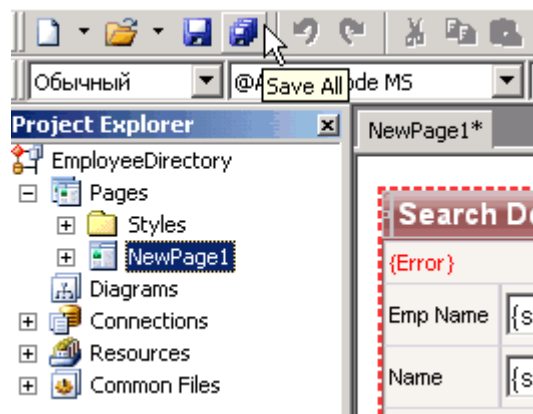
4. Click **OK** to generate the forms to the page.



Next: [Save the Project](#)

Save the Project

Click the **Save All** icon on the toolbar to save your project.

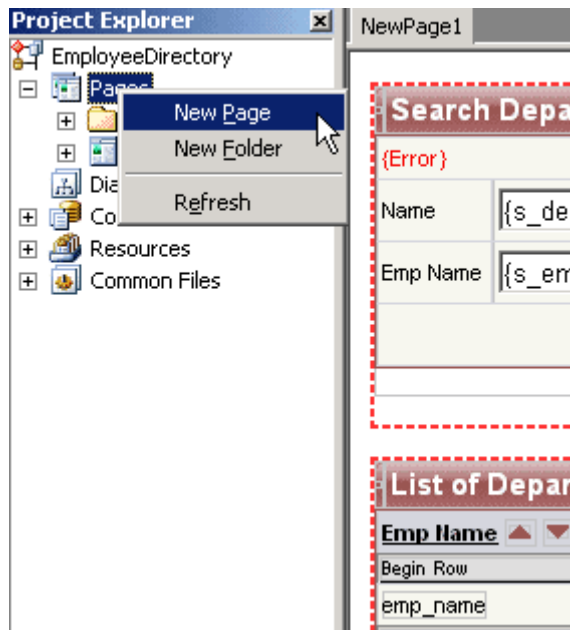


Next: [Protecting Web Pages from Unauthorized Access](#)

Step 5

Launch the Authentication Builder

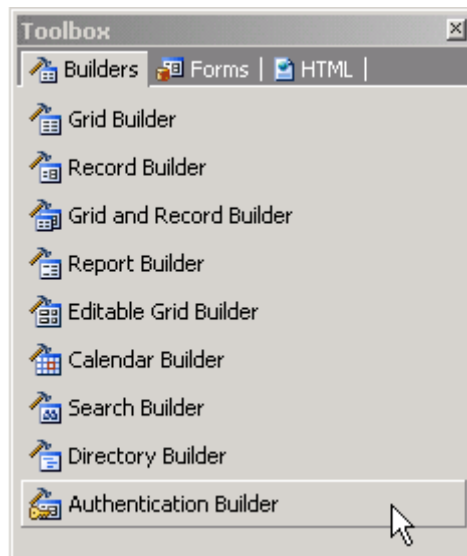
1. Create a new page.
2. Right-click on its name and select the **Rename** option.
3. Change the name of the page to *Login*.



Next: [Run the Authentication Builder](#)

Run the Authentication Builder

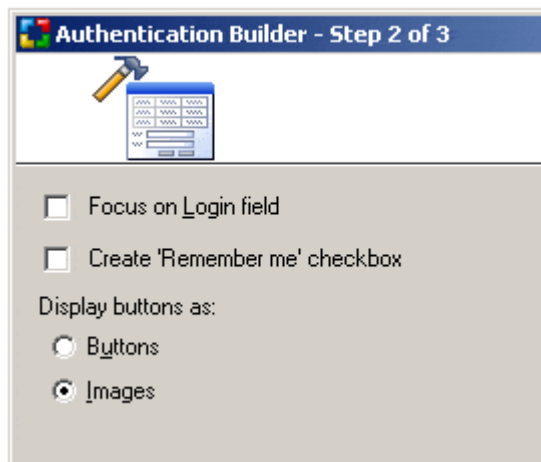
1. Click on the **Authentication Builder** icon in the **Toolbox**.
2. Once the Builder window opens, select **Login Form**.
3. Click **Next**.



Next: [Specify Login Form Options](#)

Specify Login Form Options

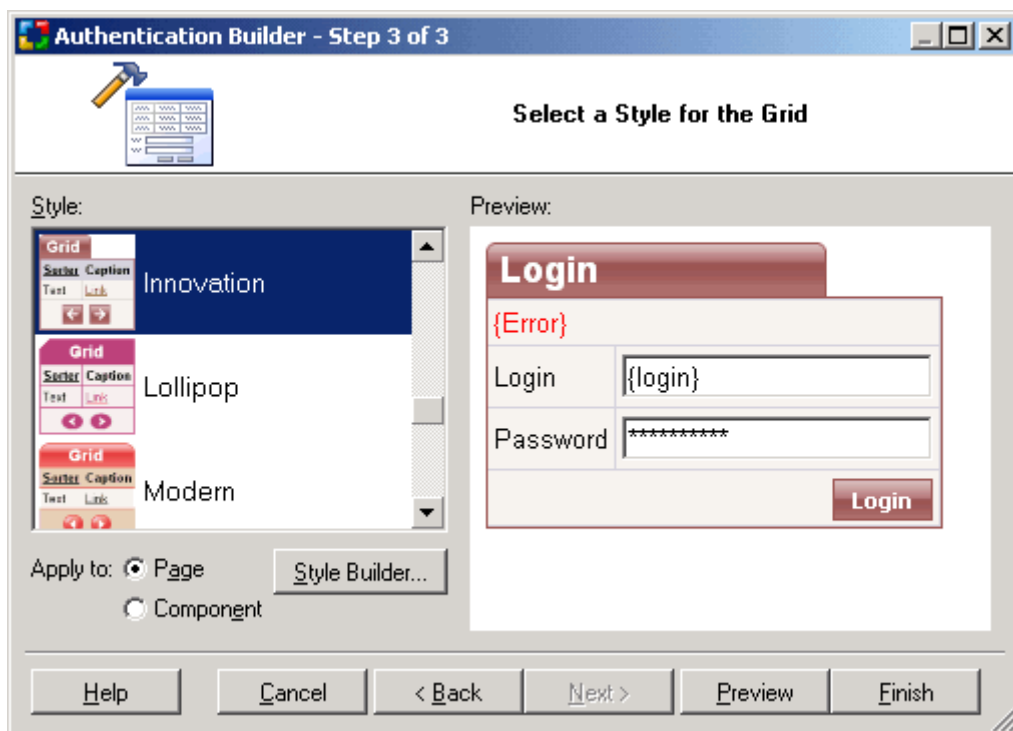
1. Select **Focus on Login field** option if you would like to generate JavaScript that will cause the Login page to always open in the browser with the Login field in focus. This will allow users to type their login information as soon as the page is shown, without the need to click on the Login field first.
2. Click **Next** to proceed.



Next: [Select a Style for the Login Form](#)

Select a Style for the Login Form

1. Select one of the available color and graphics schemes that you would like to use for the login form.
2. After selecting a style, click on the **Finish** button to generate the Login form.
3. Click on the **Save All** button to save your changes.

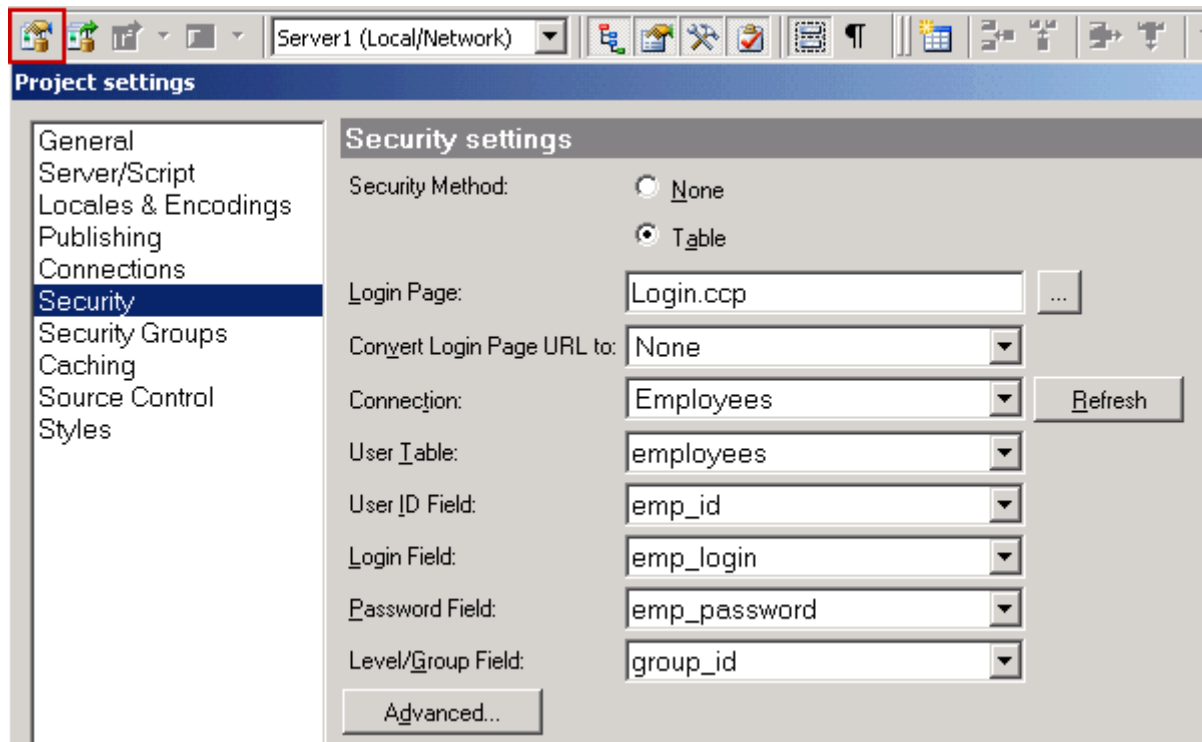


Next: [Specify the Login Page for the Project](#)

Specify the Login Page for the Project

Once back in the main CodeCharge Studio screen,

1. Use the **Project => Settings...** menu to open the **Project Settings** dialog.
2. Click on the **Security** tab.
3. Click the **[...]** button for the **Login page** property.
4. Select the *Login.ccp* page.



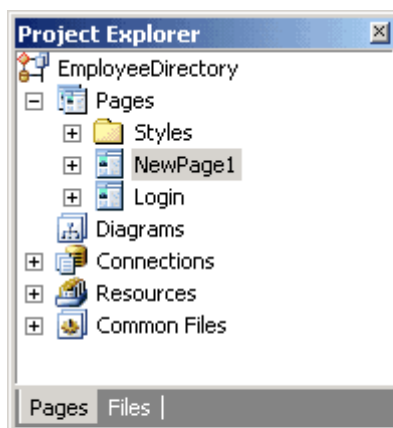
Next: [Restrict Page Access](#)

Restrict Page Access

Now you can use the authentication feature by configuring restricted access to your pages.

1. Select the *NewPage1* page in the **Project Explorer**.
2. Click on the [...] button next to the **Restricted** property.
3. In the **Page Security Groups** dialog.
4. Specify all user levels that should be able to access this page.

If you specify all user levels, all users will be able to see the page, but first will need to login to the system using the Login page.



Next: [Finalizing the Search and Grid Forms created with the Builder](#)

Step 6

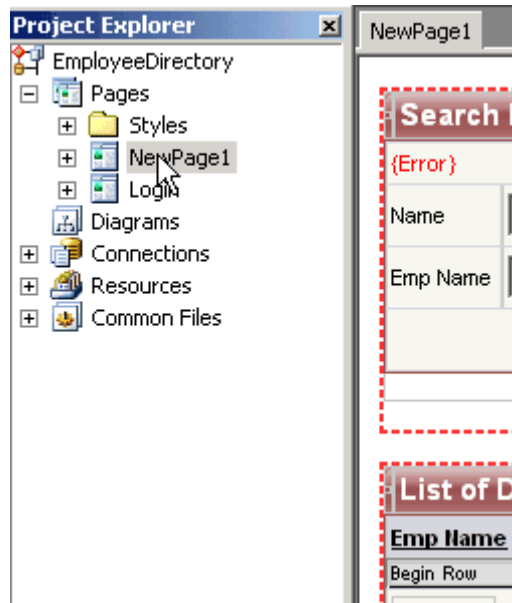
Rename the Page

First let's change the name of the page to a more appropriate name.

1. Right click on the current page name in the **Project Explorer** window.
2. Select the **Rename** option.

You can also rename a page by clicking on its name in the **Project Explorer** and pressing **F2**.

3. Type the new name for the page (*Default*).

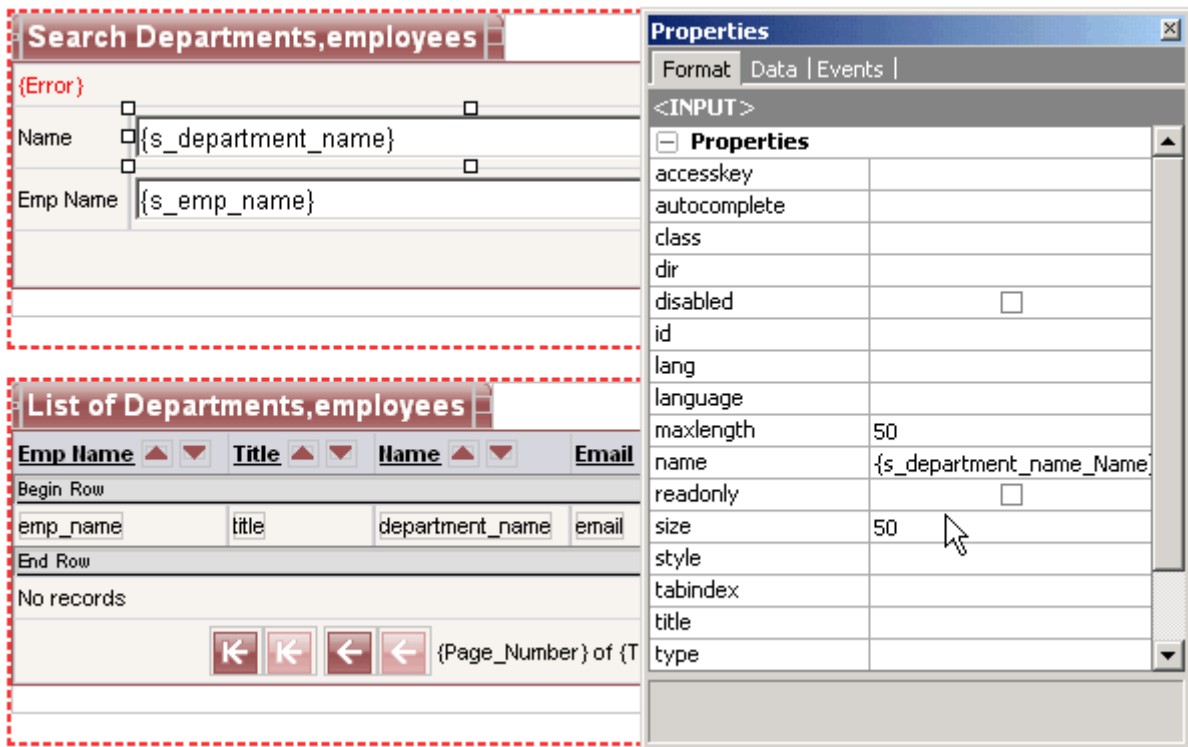


Next: [Change the Size of the Search Fields](#)

Change the Size of the Search Fields

Since some of the fields may be unnecessarily long,

1. Click on the field.
2. Adjust its **Format** Properties, for example by changing the size from 50 to 30.

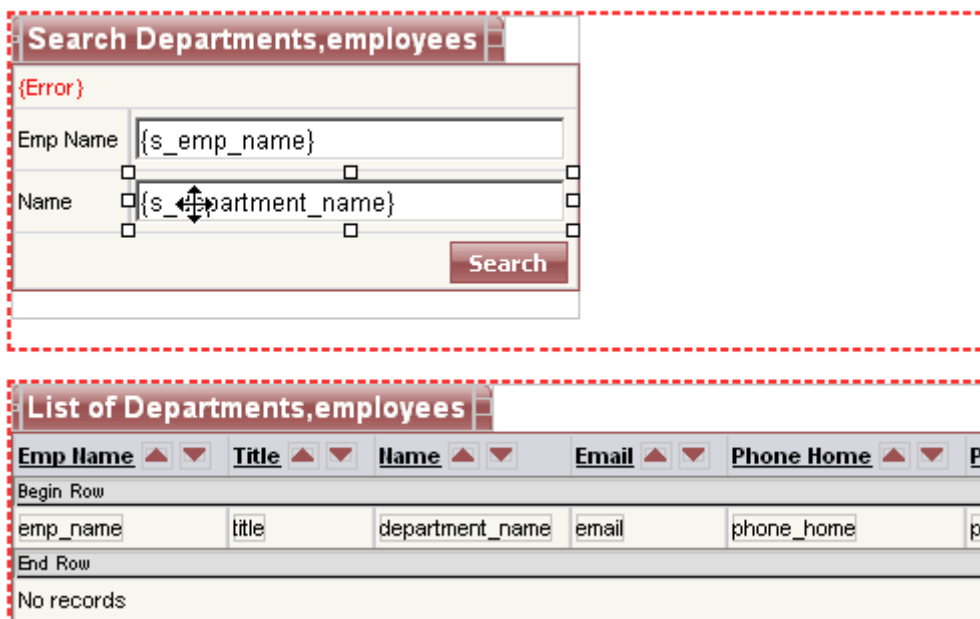


Next: [Create a ListBox Field](#)

Create a ListBox Field

ListBox fields are drop-down menus that display values from the databases. Since in the Grid Builder you specified the *department_name* field as a textbox field without configuring it as a ListBox, you now need to add a ListBox to your Search component and configure its Data Properties.

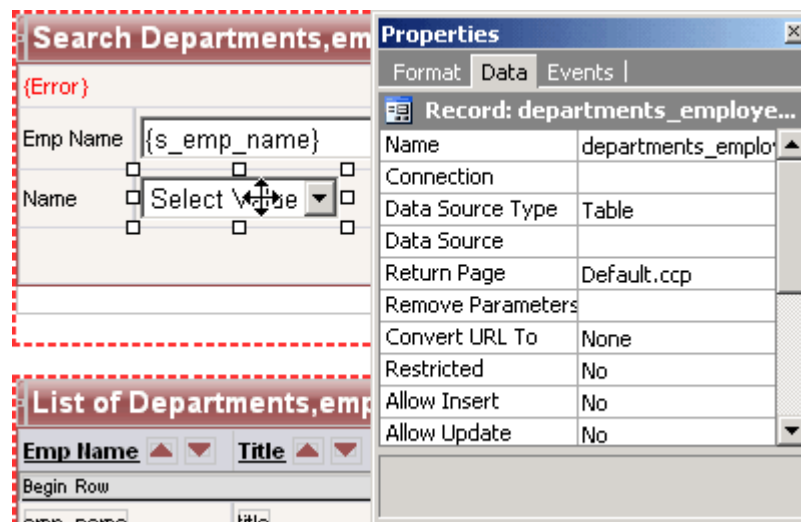
1. Right-click on the *department_name* Textbox.
2. Select **Change to** => **List Box**. This action will change your Textbox to a ListBox.



Next: [Configure the ListBox Field](#)

Configure the ListBox Field

1. Specify the Connection, Data Source, Bound Column and Text Column for the ListBox.
 - **Connection:** The database connection that contains the data for the ListBox. (*Employees*)
 - **List Data Source:** The table, view or SQL query to be used to retrieve database records for the ListBox. (*departments*)
 - **Bound Column:** The database field whose values will be submitted by the Listbox. (*department_id*)
 - **Text Column:** The database field whose values will be displayed in the Listbox. (*department_name*)
 - **Data Type:** The data type of the value that will be submitted by the Listbox. (*Integer*)



Next: [Change Field Captions](#)

Change Field Captions

Adjust the field captions created by the Grid Builder by

- working within the HTML Design window
- and typing the new titles and captions as needed.

The image shows two parts of a web application interface. The top part is a search form titled "Search" with a red header. It contains a red error message "{Error}", a text input field for "Keyword" containing "{s_emp_name}", and a dropdown menu for "Department" with "Select Value" selected. A red "Search" button is at the bottom right. The bottom part is an "Employees" grid with a red header. It has columns for Name, Title, Department, Email, Home Phone, and Work Phone. Below the columns is a "Begin Row" with input fields for emp_name, title, department_name, email, phone_home, and phone_work. Below that is an "End Row" and a "No records" message. At the bottom are navigation buttons (back, forward) and a pagination display "{Page_Number} of {Total_Pages}".

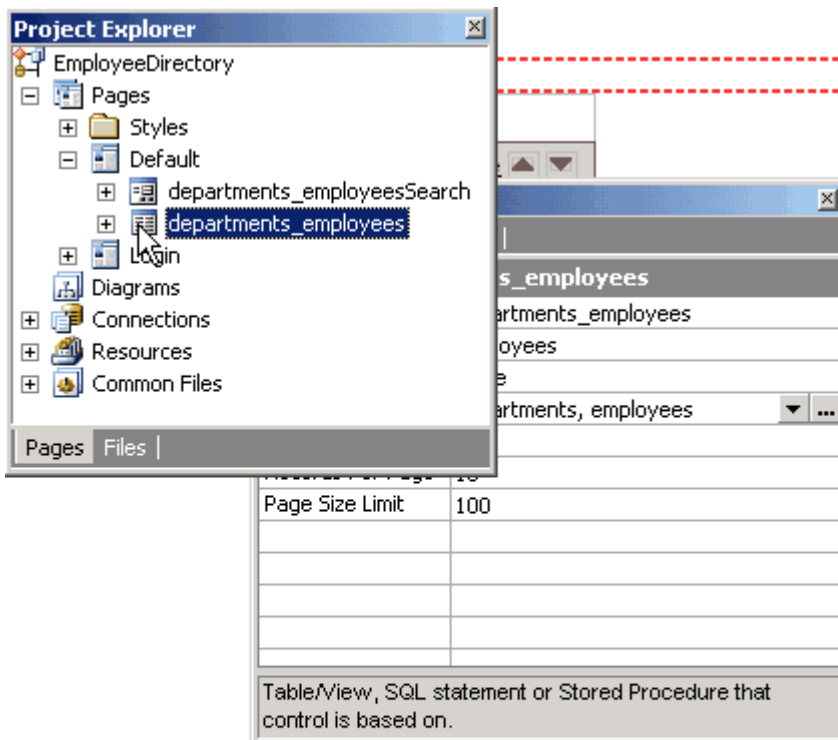
Next: [Setup Search Parameters](#)

Setup Search Parameters

1. Switch to page **Design** mode.
2. Select the Grid form by either selecting it in the **Project Explorer** or by positioning the cursor anywhere within the Grid's caption on the page.
3. Click on [...] button in the **Data Source** property of the Grid.

Once the Data Source window opens up you will see several parameters created by the Query Builder in the **Where** section. The two parameters there indicate that the Grid should be filtered by two keywords: *s_emp_name* and *s_department_name*. Both of those keywords come from the Search form and will be matched against the *emp_name* and *department_name* fields from the database respectively. However, we would also like to search by employees' titles, and since the *department_name* was changed to a ListBox it will need adjustment as well.
4. Click the '+' sign to add a new **Where** parameter that will be used in the Grid.
5. Select the field *title*.
6. Specify that it should be matched against the search parameter *s_emp_name*, the same as used to search *emp_name*.
7. For the **Condition** specify *contains (like '%...%')* so that all employee titles that contain the keyword *s_emp_name* will be retrieved.
8. Modify the search parameter associated with the *department_name*.
 - a. Double click on it.
 - b. In the parameter setup window select *departments.department_id* as the field that will be searched with the *s_department_name* parameter.
 - c. Select *equals (=)* as the **Condition** since users will select the exact department using the ListBox and set the **Type** to *Integer*.

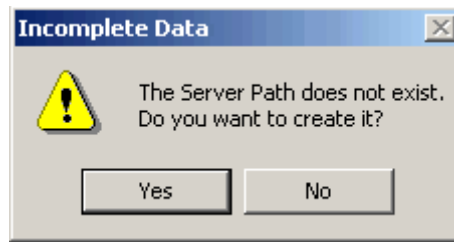
9. Finally, we need to specify that we want to see the results if either the Employee Name or Title matches the specified Keyword.
Click on the title parameter and move it upwards (^)
 10. Select both the *emp_name* and *title* parameters by
 - o holding the **Ctrl** key
 - o and clicking on each one.
 1. Click the parentheses button [()] to make the search of these two parameters independent from searching the department.
 2. Double-click on the *emp_name* parameter and set the **And/Or** field to *Or*.
- Your final Where/Search parameters screen should look like the one shown here.



Next: [Publish the Page](#)

Publish the Page

1. Click on the **Live Page** tab in the document window to test the page in the same way as it would be accessed by users via a browser.
2. If this is the first time you are publishing this project and the publishing folder doesn't exist, CodeCharge Studio displays a window asking to approve the creation of a new folder.
3. Click **Yes** to confirm and continue.



Next: [Preview and Test the Project](#)

Preview and Test the Project

As the final step,

1. Click on the **Live Page** tab.
2. Test the page by interacting with it and testing its functionality.

Search

Keyword	<input type="text" value="developer"/>
Department	<input type="text" value="Web Development"/>
<input type="button" value="Search"/>	

Employees

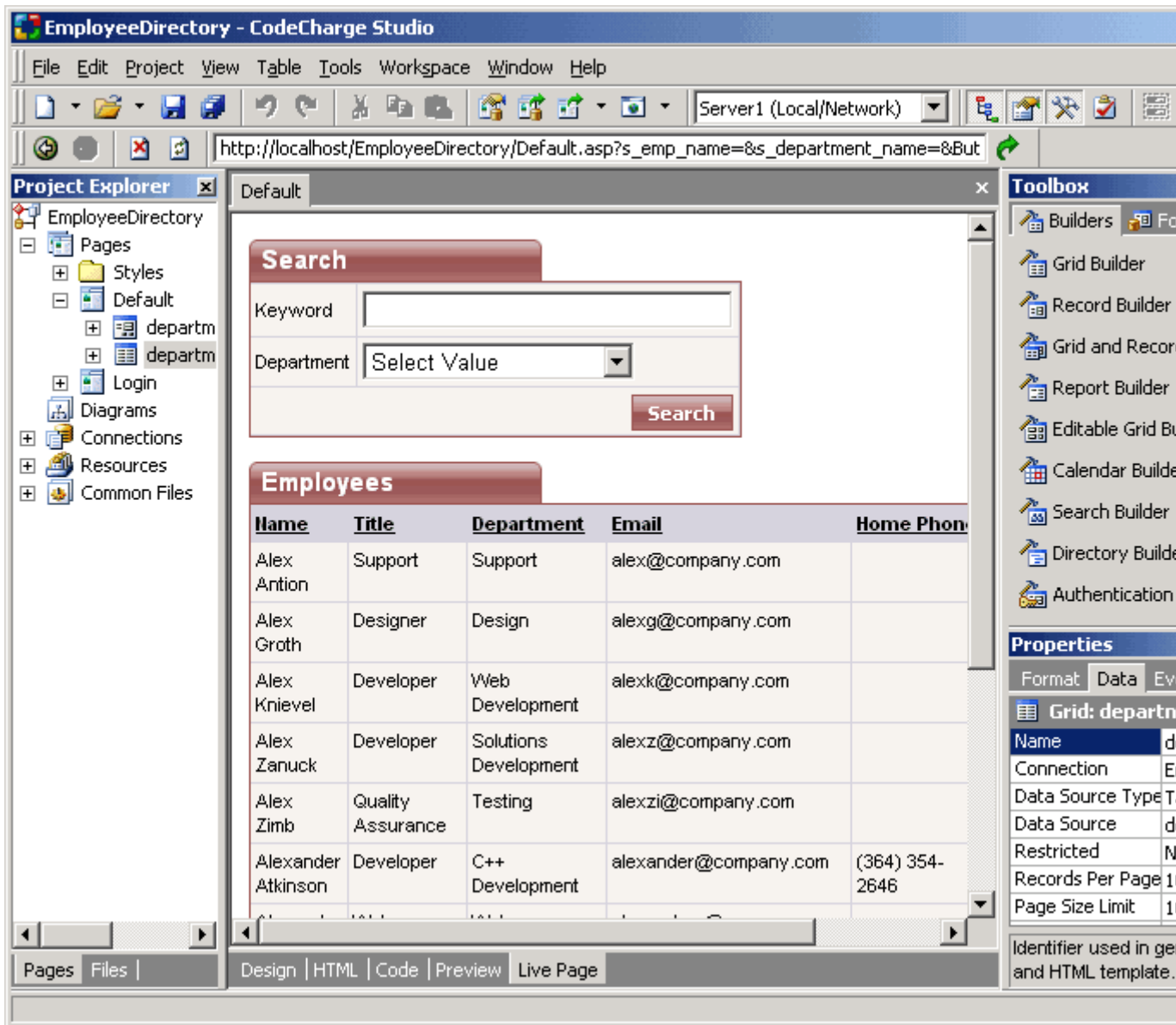
<u>Name</u>	<u>Title</u>	<u>Department</u>	<u>Email</u>	<u>Home Phone</u>	<u>Work Phone</u>
Alex Knievel	Developer	Web Development	alexk@company.com		
Arty Blake	Developer	Web Development	arty@company.com		
Ignace Shaw	Developer	Web Development	ignace@company.com	(564) 343-3367	(364) 134-5411
Michael Koenig	Developer	Web Development	michael@company.com		
Oleg Tim	Developer	Web Development	olegt@company.com		
Stan Simon	Developer	Web Development	stan@company.com		
Victor Tomlin	Developer	Web Development	victor@company.com		
Vitas Reynolds	Developer	Web Development	vas@company.com		

⏪ ⏩ 1 of 1 ⏪ ⏩

Next: [Conclusion](#)

Conclusion

Congratulations! During the course of this brief tutorial, you've created an Employee Directory application made of a searchable grid. Click on the **Live Page** tab to view the results or open the page in your browser.



Back to: [Tutorials](#)

Creating a Task Management System with the Application Builder

Step 1

Create a New Project

1. Start CodeCharge Studio
2. Select **New project** in the initial screen.

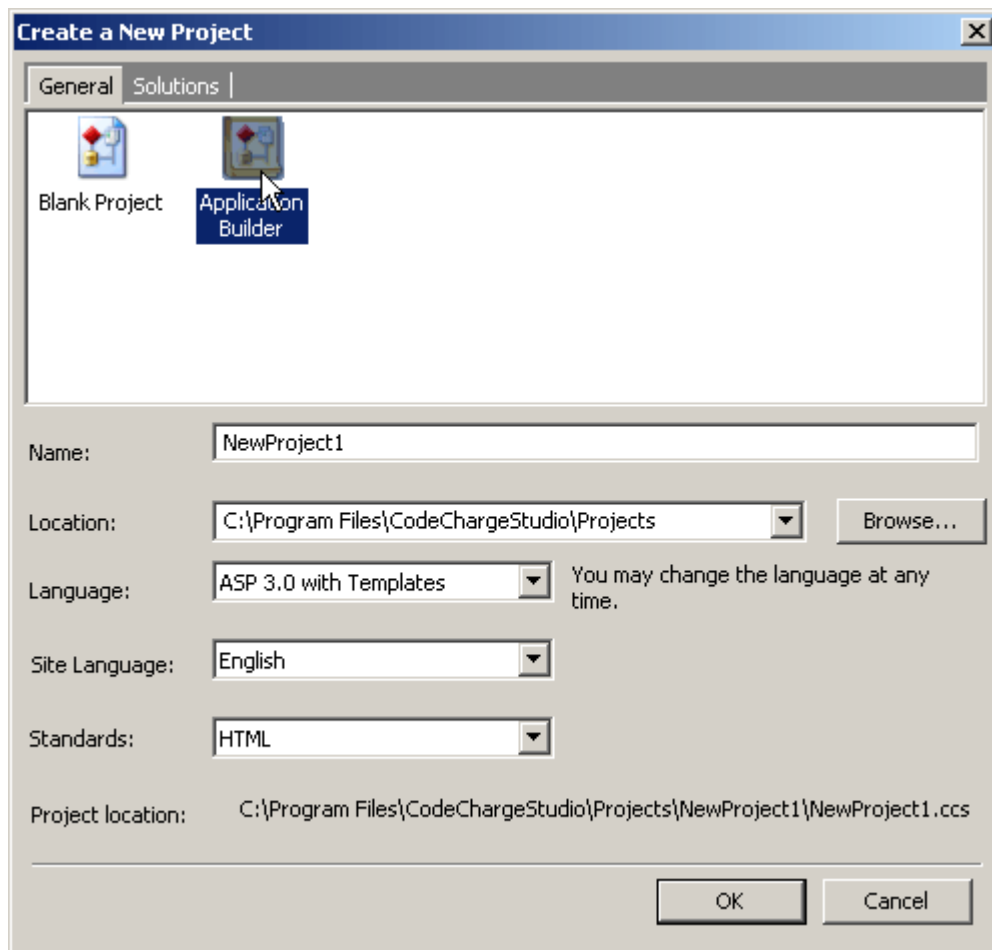


Next: [Launch the Application Builder](#)

Launch the Application Builder

To start the Application Builder:

1. Specify the name of the new project (*TaskManager*), the location on the disk where the project will be published, and the programming language.
2. Double-click on the **Application Builder** icon.
3. Select the **Application Builder** icon then click the **OK** button to launch it.



Next: [Specify Project Properties](#)

Specify Project Properties

Specify a number of parameters required by the Application Builder for generating the site.

1. **Code Language:**

Programming language or technology to be generated. Currently supported technologies are:

- *ASP 3.0 with Templates* - generates ASP 3.0 code and separate .html templates.
- *ASP.Net C#* - generates .aspx files with C# code.
- *ASP.Net VB* - generates .aspx files with VB.Net code.
- *CFML 4.0.1* - generates ColdFusion 4.0.1 code.
- *CFML 4.0.1 with Templates* - generates ColdFusion 4.0.1 code (.cfm) and separate .html template files.
- *JSP 1.1 JDK 1.3* - generates JSP 1.1 code.
- *PHP4/PHP5 with Templates* - generates PHP code (.php) and separate .html template files.
- *Servlets 2.2 JDK 1.3 with Templates* - generates Java code that utilizes .html templates.

2. **Site Language:**

Specify the spoken language to be used when generating text messages for the site. For example the text *No records* that appears when no more records

are to be displayed in a grid could be generated in any one of the supported languages.

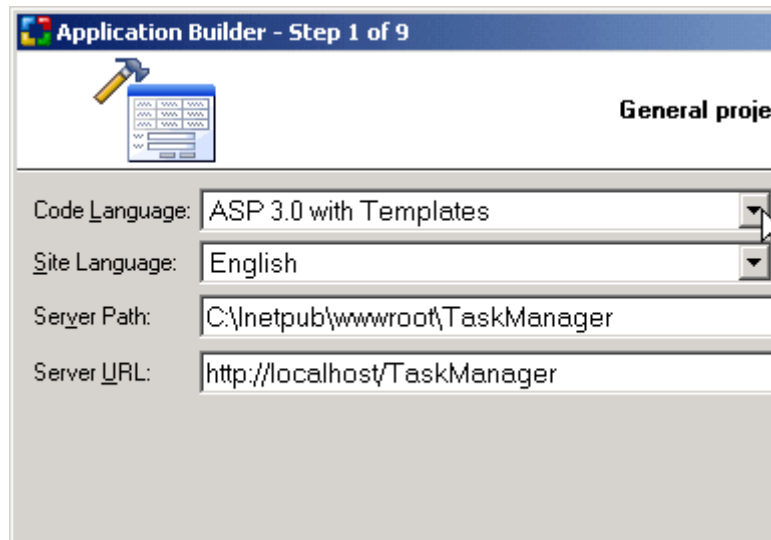
3. **Server Path:**

The full path where generated files should be published (locally). This path is usually preset by the Application Builder and can be left without changes.

4. **Server URL:**

The web address corresponding to the **Server Path**. This URL will be used to view the pages in **Live Page** mode. The Application Builder automatically defaults to the appropriate URL that matches the server path.

5. Click **Next** to proceed.



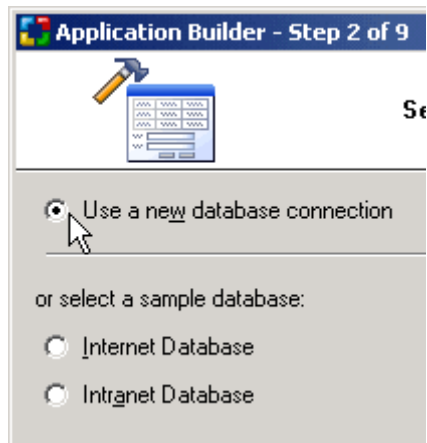
Next: [Select Database Connection](#)

Select Database Connection

This application uses the *Intranet* sample database so we don't have to create a new database connection. The *Intranet* sample database contains the *tasks*, *priorities* and *employees* tables which will be used in this project.

1. Select the **Intranet Database** checkbox.
2. Click **Next** to proceed.

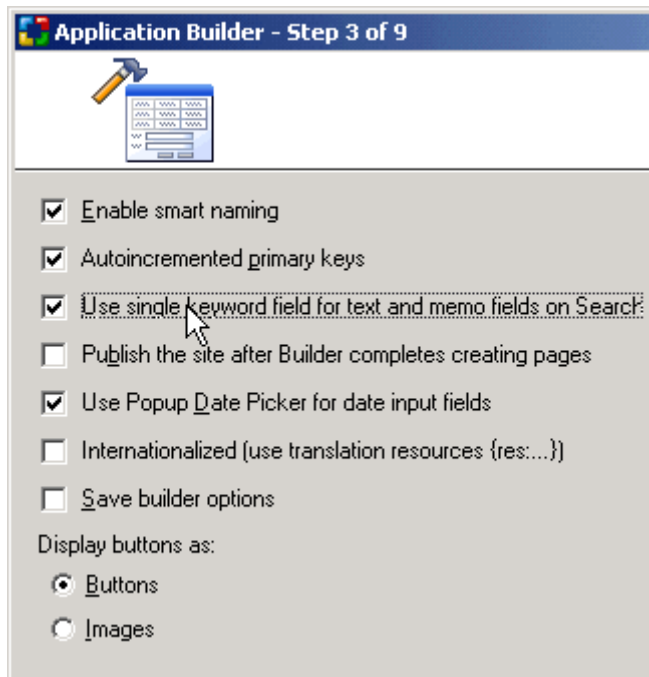
Note: You can use this sample database regardless of the language you selected in the step above. This is because the design time connection string is made by CodeCharge Studio and therefore can uniform regardless of the language being used. If you use a language such as PHP and ColdFusion, you will later have to configure the server side connection to connect to MySQL or other database.



Next: [Configure the Application Builder](#)

Configure the Application Builder

1. Configure the Application Builder by specifying configuration options as follows:
 - **Enable smart naming**
Select this option so that the Application Builder will automatically convert table names to English captions, for example *employees* table will be shown as a grid with the title "List of Employees". Fields like *task_name* will be converted to column headings like "Task Name"
 - **Autoincremented Primary Keys**
Select this option to specify that the database tables contain key fields that are autoincremented. The Application Builder will then hide the key fields from the record maintenance forms since users do not need to enter key values.
 - **Use single keyword field for text and memo fields on search forms**
Select this option to generate a single search field that searches against all the text and memo fields on the tables/grids. If this option is deselected, the Application Builder creates a search section with multiple search fields - one search field for each text or memo field in the database table.
 - **Publish the site after Application Builder completes creating pages**
Specify that you want to publish the site as soon as the Application Builder creates all necessary pages. This way you don't have to worry about forgetting to generate/publish some of the files needed for the application to run.
 - **Use Popup Date Picker for date input field**
Select this option so that fields that are detected as being date fields are created with a corresponding Date Picker component. The Date Picker component allows the user to easily specify date values.
 - **Internationalized**
Select this option when developing a multilingual application so that the locale can be switched at run-time
 - **Save builder options**
Select this option to save current builder settings so that they will be selected by default during next builder runs.
2. Click **Next** to proceed.

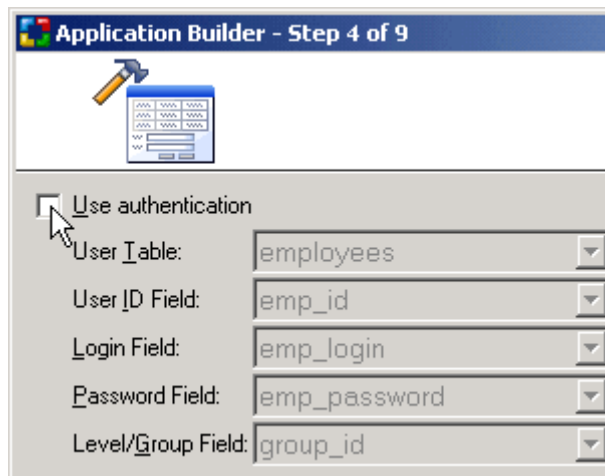


Next: [Setup Site Security and Authentication](#)

Setup Site Security and Authentication

In this step, you can specify if you want to use authentication and check users' access privileges before allowing them to access certain pages.

1. Select **Use authentication** and leave all default options. The Application Builder will then create the Login page and will allow you to specify a security level for each of the pages created.
2. Click **Next** to proceed.

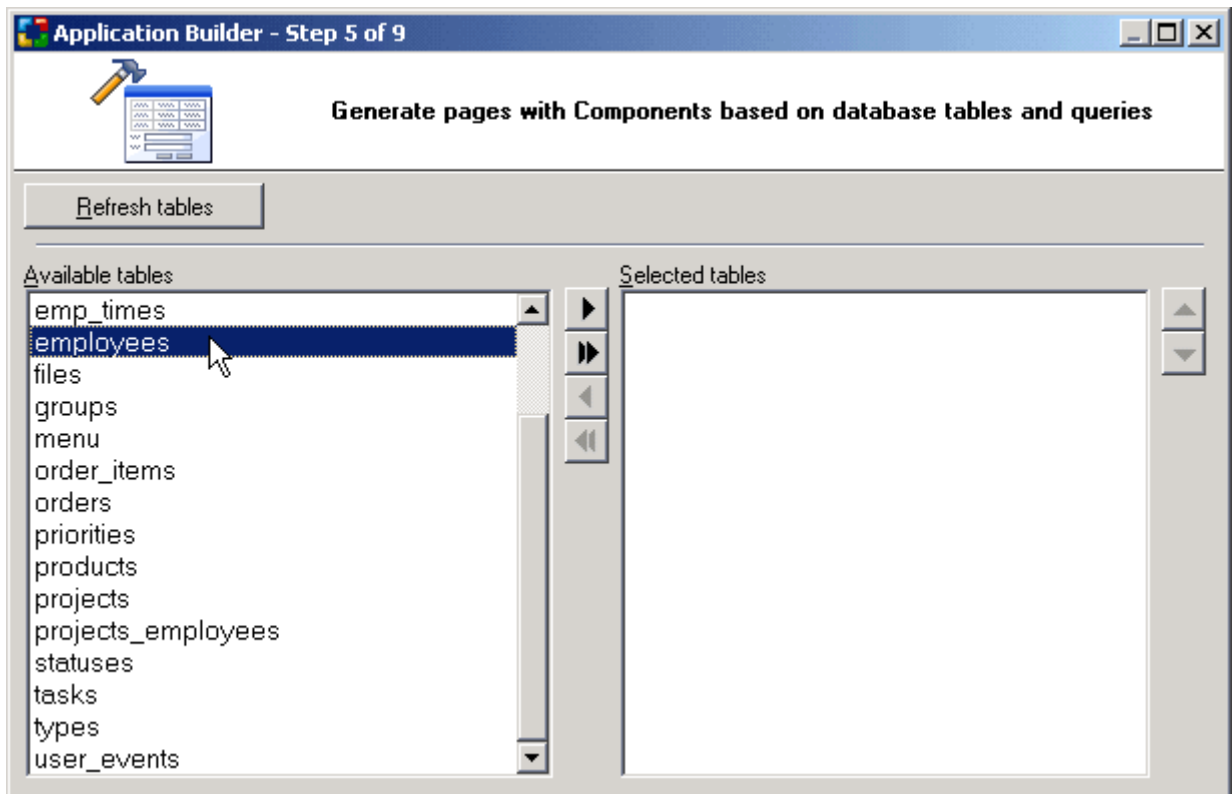


Next: [Select Database Tables](#)

Select Database Tables

1. Select the following database tables based on which the application will be built:
 - o *employees*
 - o *priorities*
 - o *projects*

- *statuses*
 - *tasks*
2. Click **Next** to proceed.



Next: [Configure Site Pages](#)

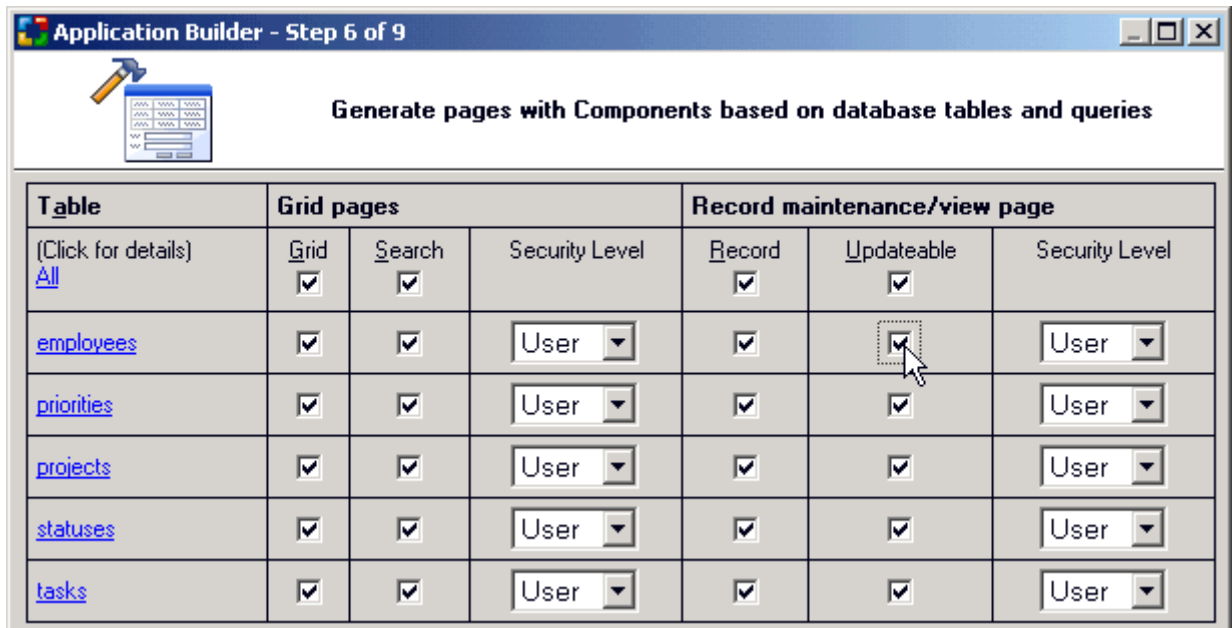
Configure Site Pages

Now configure your site by specifying options and security settings for each of the pages. The Application Builder creates two pages for each of the tables:

- *Search and Grid page*
- *Record Maintenance page*

By clicking on a name of any of the tables, you can specify whether the Search, Grid and Record forms should all be on the same page for each of the tables.

1. For this tutorial, configure the pages as shown below. The Application Builder will then convert the tables to web pages as follows:
 - *employees* table: searchable list of employees page and employee information page, accessible only by authorized users
 - *priorities* table: list of priorities page and priority maintenance page, accessible only by administrators
 - *projects* table: list of projects page and project maintenance page, accessible only by administrators
 - *statuses* table: list of statuses page and status maintenance page, accessible only by administrators
 - *tasks* table: searchable list of tasks accessible by anyone, and task maintenance page accessible by authorized users
2. Click **Next** to proceed.

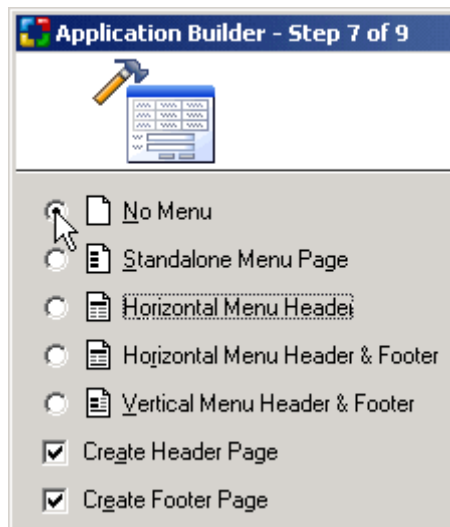


Next: [Specify Site Layout and Menu](#)

Specify Site Layout and Menu

The Application Builder automatically creates a header page with a menu, which is then included in all other pages for easy navigation.

1. Select horizontal position of the menu for all pages.
2. Click **Next**.

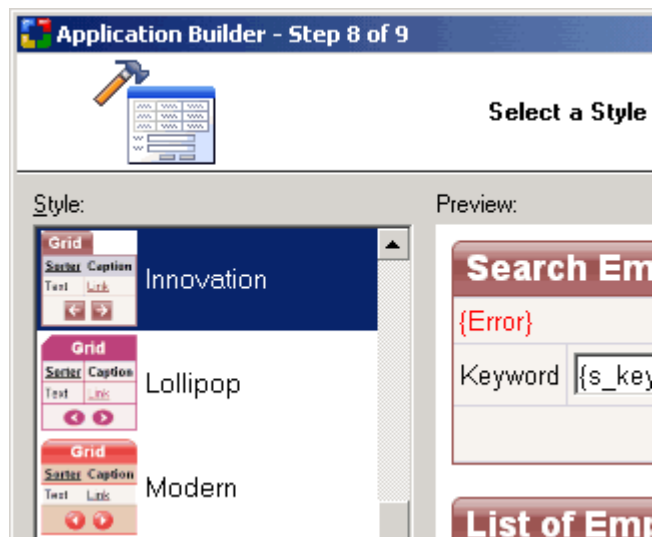


Next: [Select Site Style](#)

Select Site Style

Finally,

1. Select the *Basic* style to apply to the site.
The Application Builder will use the style to apply specific fonts, images and colors to each page.
2. Click **Next** to proceed.

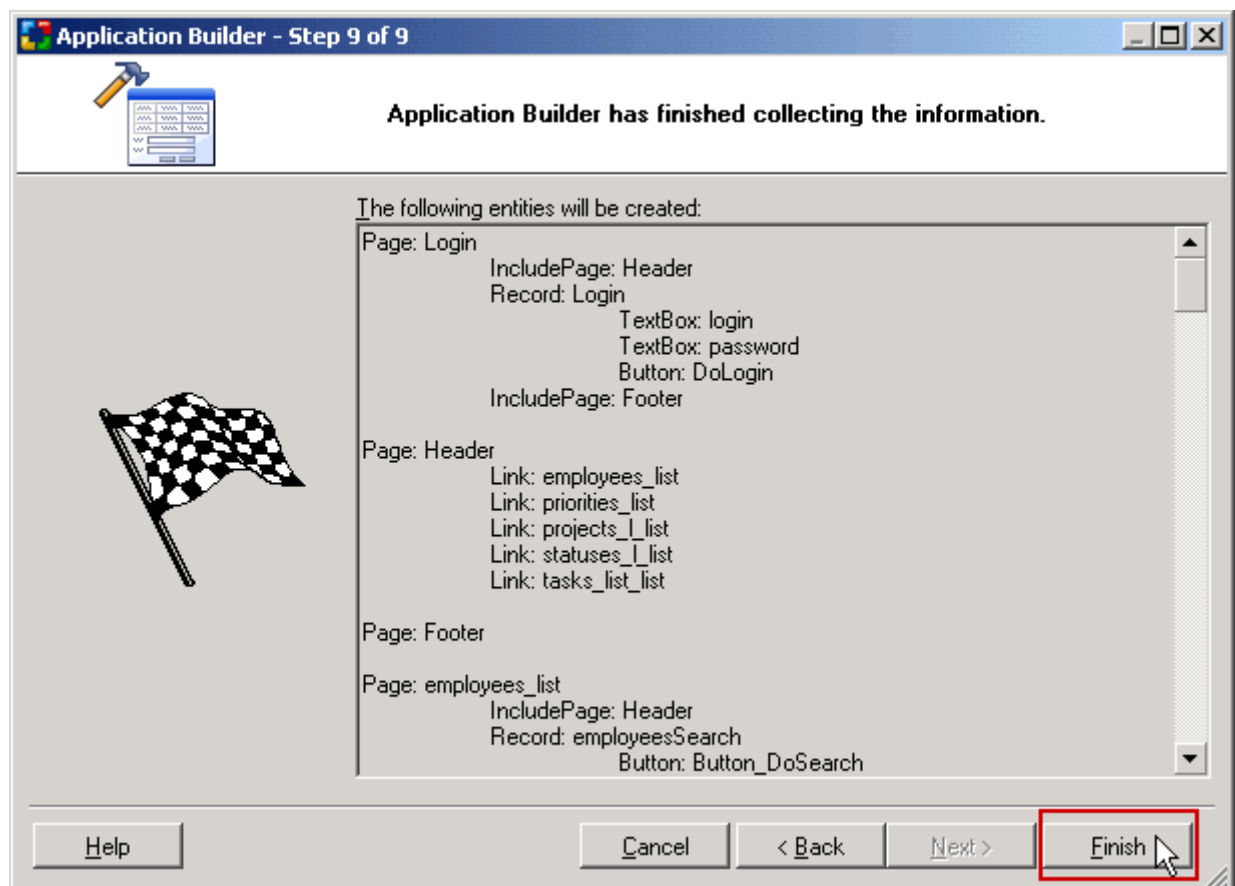


Next: [Review Pages and Create the Site](#)

Review Pages and Create the Site

The final window contains a summary of the application that will be built. To build the application:

1. Click on the **Finish** button.

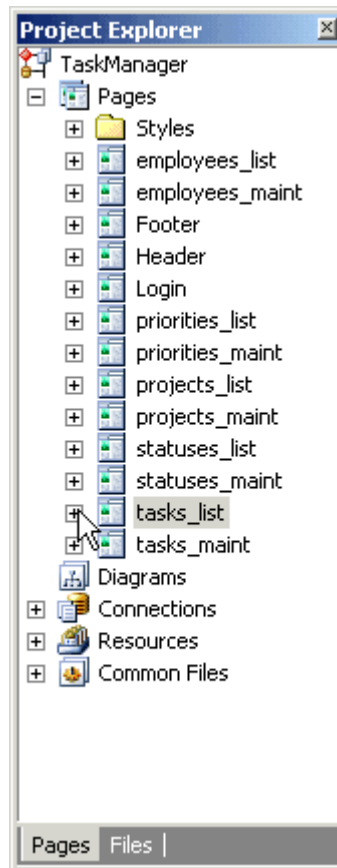


Next: [Customizing the Task List Page](#)

Step 2

Open the Task List Page

In the **Project Explorer** window click on the '+' sign next to the tasks_list page to open it for modification.



Next: [Test the Page](#)

Test the Page

1. Once the page is open click on **Live Page** tab to view and test the working page.
2. After the login you should see a page similar to that shown below.

The grid form in this page is based on the *tasks* database table. However, when the Application Builder created the form, it also took into consideration any relationships that existed between the *tasks* database table and other tables with similar field names. So for instance, the *Project Name* field below is created based on a relationship between the *tasks* table and the *projects* table which contains the names of all the projects.

Header

Search Tasks

{Error}

Keyword

List of Tasks

Id ▲▼	Project Name ▲▼	Priority Name ▲▼	Status Name ▲▼	Type Name ▲▼	Name ▲▼	User Id Assign
Begin Row						
{task_id}	project_name	priority_name	status_name	type_name	task_name	user_id_assign_b
End Row						
No records						
<input type="button" value="Add New"/> <input type="button" value="<<<"/> {Page_Number} of {Total_Pages} <input type="button" value=">>>"/>						

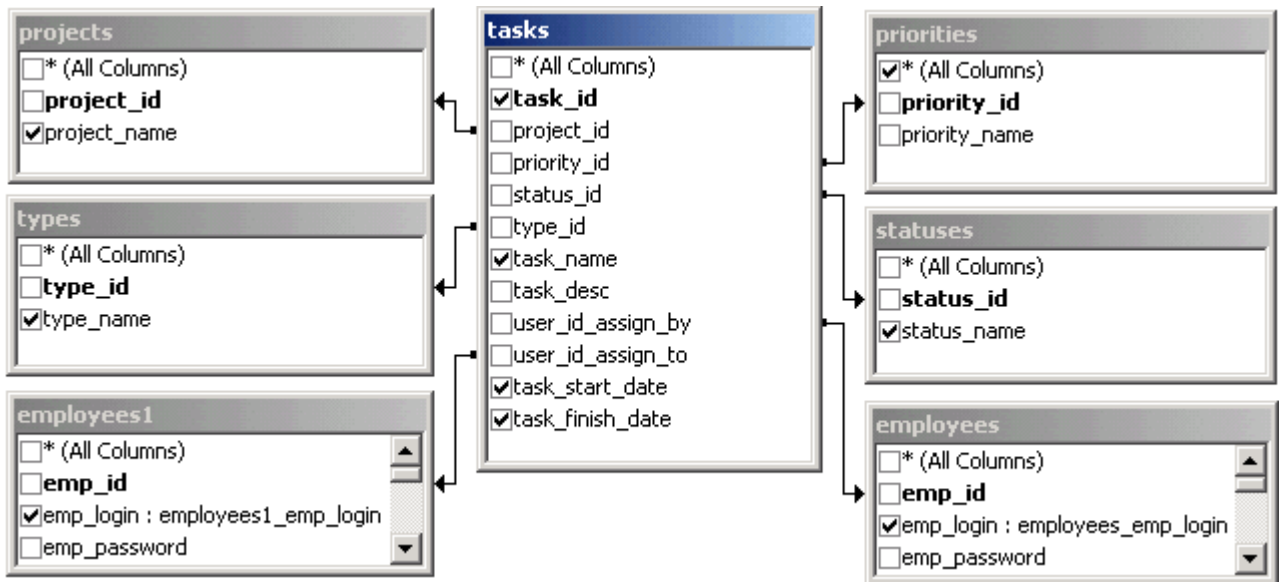
Footer



Next: [Implicit Relationships](#)

Implicit Relationships

In actual sense the grid form in the `tasks_list` is not based solely on the `tasks` table but rather on a query which draws together all the fields in the different tables related to the `tasks` table. When the Application Builder is creating a form, it attempts to discover any relationships between the base table, in this case the `tasks` table, and other tables within the database. If any relationships are discovered, the Application Builder creates a query to join the fields that make up the relationship. The diagram below shows the join query that is used for the grid form in the `tasks_list` page.



Note that the ability to discover relationships varies by database type. While it is possible to discover relationships in some databases such as Access, other databases may not support this feature. If the Application Builder is not able to discover the relationships, they have to be implemented manually using the **Data Source** property of the form concerned.

Unless you specify otherwise during the creation process, the Application Builder builds grid forms that include all the fields in a given table. However, you might not want some of the fields to be shown.

List of Tasks							
Id	Project Name	Priority Name	Status Name	Type Name	Name	User Id Assign By	User Id Assign
1	CodeCharge	High	On hold	Issue	Great Project needs to be greater	ken	helen
2	CodeCharge	Highest	Closed	Question	Fix ALL bugs	ken	george
3	CodeCharge	High	Closed	Task	Get ready to click	ken	peter
4	My Project	Highest	Open	Task	Finish My Project	ken	ignace
5	Test Project	High	In progress	Task	Test this project.	ken	ken
6	CodeCharge	Highest	Open	Task	Code with one hand.	ken	alexander
7	Test Project	Highest	On hold	Task	Get armed	ken	helen
8	Test Project	Highest	Open	Question	Write more code	ken	ignace
9	Super Project	Highest	In progress	Task	Code, code, code...	ken	george
10	Test Project	Lowest	On hold	Task	Sleep	ken	ken
11	Super Project	Highest	Open	Task	Have fun	ken	alexander

[Add New](#) 1 of 1

In our case, we do not want to show the *Type Name*, *User Id Aassign By*, *Start Date* and *Finish Date* fields.

Next: [Delete Unneeded Columns](#)

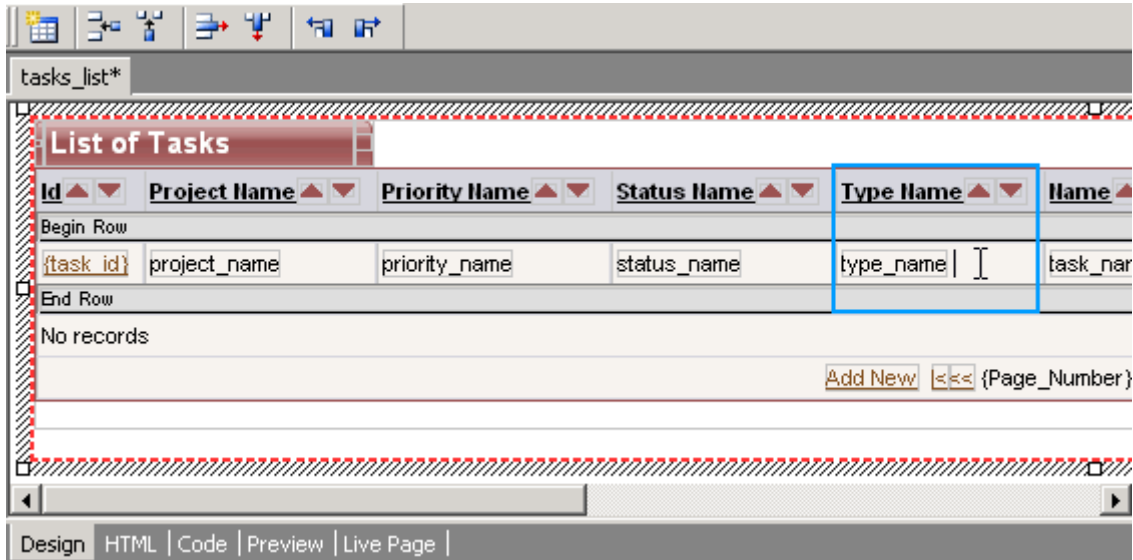
Delete Unneeded Columns

To delete the unwanted columns:

1. Revert to **Design** mode and select the unneeded column by clicking and positioning the cursor somewhere within the column.
2. Click on the **Delete Column** icon in the toolbar to remove the column.

Use this method to delete the following columns:

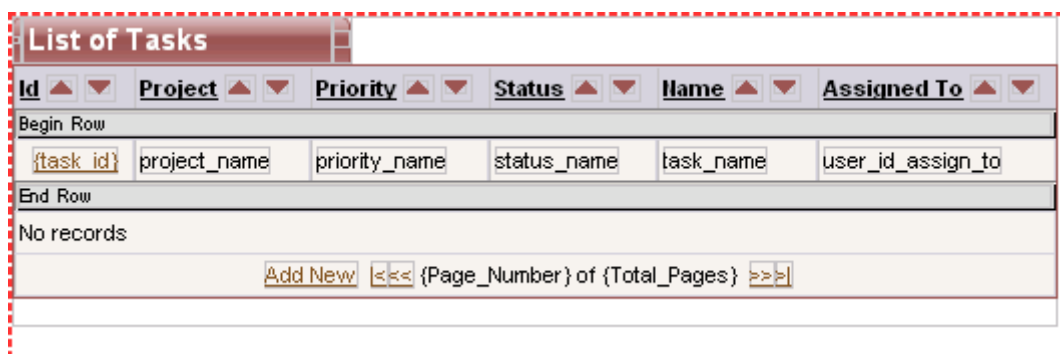
- *Type Name*
- *User Id Assign By*
- *Start Date*
- *Finish Date*



Next: [Change Field Caption](#)

Change Field Caption

1. Revert to the **HTML** mode.
2. Change the caption text for the following fields:
 - *Project Name* to *Project*
 - *Priority Name* to *Priority*
 - *Status Name* to *Status*
 - *User Id Assign To* to *Assigned To*



Next: [Synchronize HTML and Programming Code](#)

Synchronize HTML and Programming Code

Because of the fields we deleted earlier, we need to synchronize the programming code so that it matches the HTML code. To do so:

1. Click on the **Live Page** tab.

A message window will be shown alerting you that some of the components were not found in the HTML.

- Click **Yes** to confirm the removal of the columns and continue.



Next: [View and Test the Live Page](#)

View and Test the Live Page

Finally, you can view the working page with a grid containing the list of tasks that can be sorted by clicking on column headings or searched by entering a keyword.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Search Tasks

Keyword

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

1 of 1

Next: [Add ListBox Search for Project Names](#)

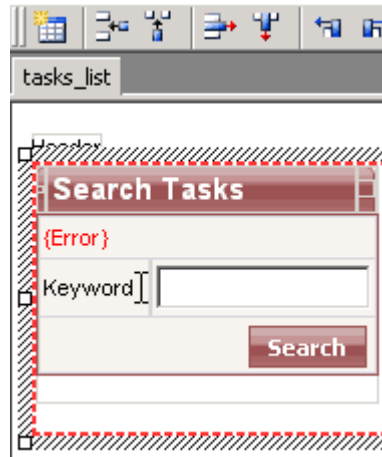
Step 3

Add ListBox Search for Project Names

Now let's add an additional search field to the search form. We shall add a listbox with project names so that users can filter the grid by selecting a project from the listbox.

We first need to add a row in the search form where the listbox will be placed.

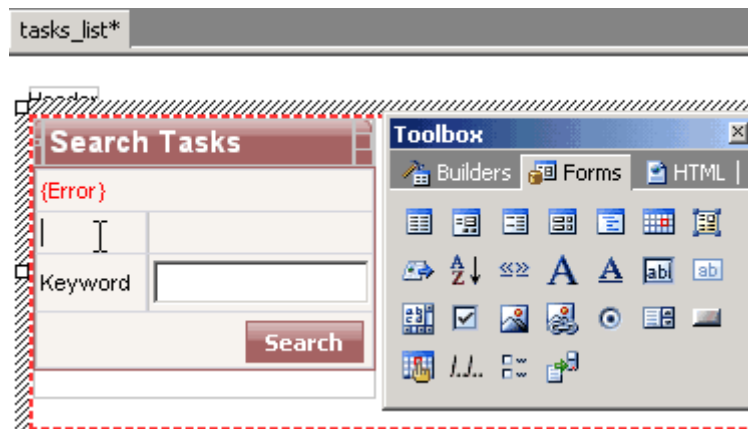
1. Position the cursor somewhere within the *Keyword* text by clicking on it.
2. Select the **Insert Row** icon to add a new table row at the top of the search section.



Next: [Add ListBox Search - Insert a ListBox Control](#)

Add ListBox Search - Insert a ListBox Control

1. Type the text *Project* within the newly created left table cell.
2. Position the cursor in the right cell as shown.
3. Click on the **Add ListBox** icon in the Toolbox to add it to the page.

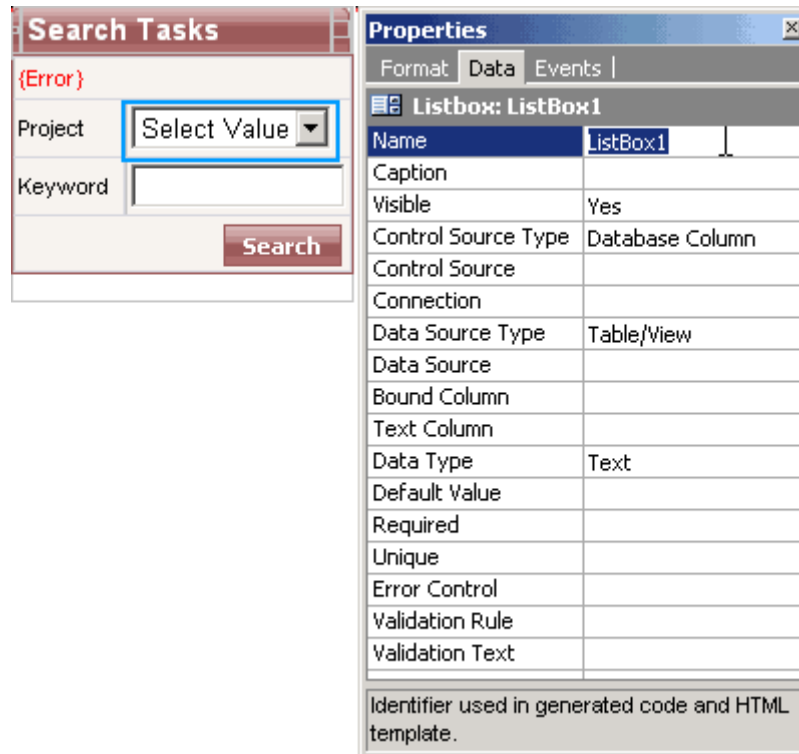


Next: [Add ListBox Search - Set ListBox Properties](#)

Add ListBox Search - Set ListBox Properties

1. Configure the ListBox properties by clicking on it and specifying the required values in the property editor as follows:
 - **Name:** *s_project* - this name will be used later as an input variable name for the selected value
 - **Connection:** *IntranetDB* - database connection to use for retrieving ListBox values
 - **Data Source:** *projects* - table containing ListBox values

- **Bound Column:** *project_id* - table field whose value will be used as the search parameter
- **Text Column:** *project_name* - table field whose value will be displayed in the ListBox
- **Data Type:** *Integer* - type of the value that will be used as the search parameter (*project_id* is numeric)



Next: [Add ListBox Search - Move Table Row](#)

Add ListBox Search - Move Table Row

Finally:

1. Move the table row containing the ListBox downwards by
 - right clicking near the ListBox
 - and selecting **Move Row Down**.

You can also do this by positioning the cursor next to the ListBox and using the *Alt + Down Arrow* keyboard keys.

Search Tasks

{Error}

Project| Select Value ▾

Keyword

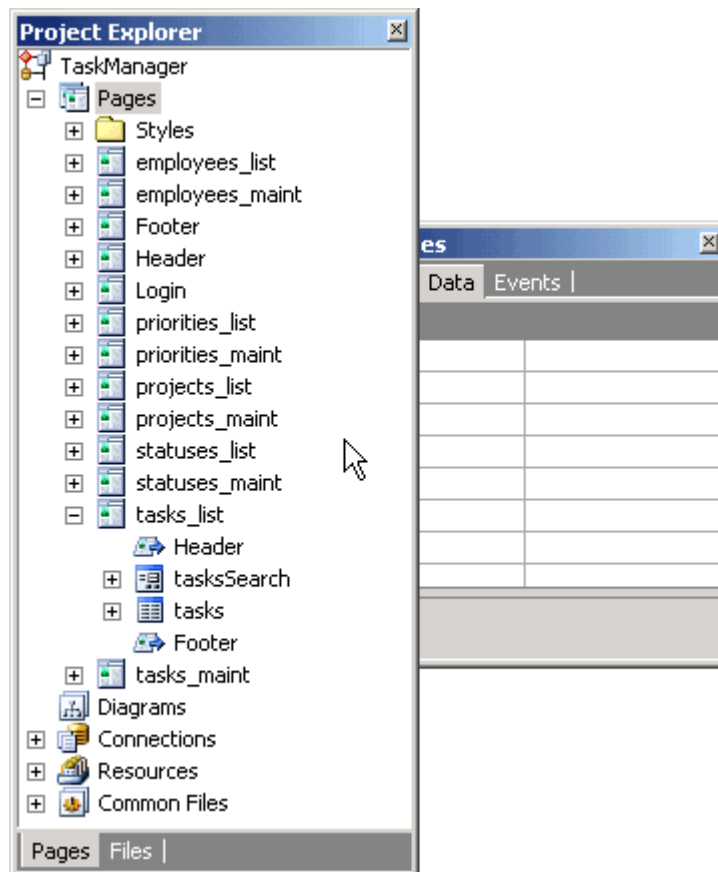
Search

Next: [Filter Grid Records - Select "Where" Property](#)

Filter Grid Records - Select "Where" Property

A working ListBox is now created on the page but it cannot be used to filter the grid's records until the grid itself uses the parameter passed by the ListBox. To setup input parameters you will need to set the WHERE criteria within the grid's data source.

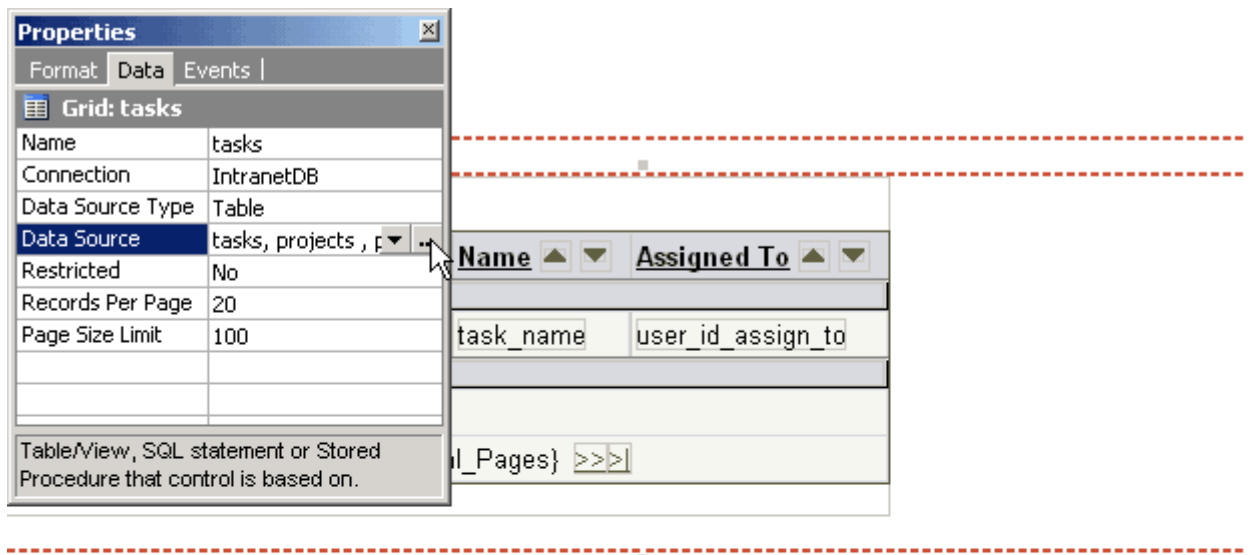
1. Select the grid by clicking anywhere within the grid's caption on the page or by selecting it in the **Project Explorer**.
2. Click on the [...] button of the **Data Source** property.



Next: [Filter Grid Records - Add Search Parameter](#)

Filter Grid Records - Add Search Parameter

1. Add a new search parameter to the grid by
 - selecting WHERE section,
 - removing unnecessary conditions,
 - double clicking on the **tasks.project_id** field,
 - and then specifying the *tasks.project_id* field to be matched against the *s_project* parameter, which is the name of the previously created ListBox.
2. The **Type** field for the parameter should also be set to *Integer*.
 The grid form will now receive this new parameter via the URL then show only matching results.
3. Click **OK** when finished entering the information.



Next: [Filter Grid Records - Group "Where" Parameters](#)

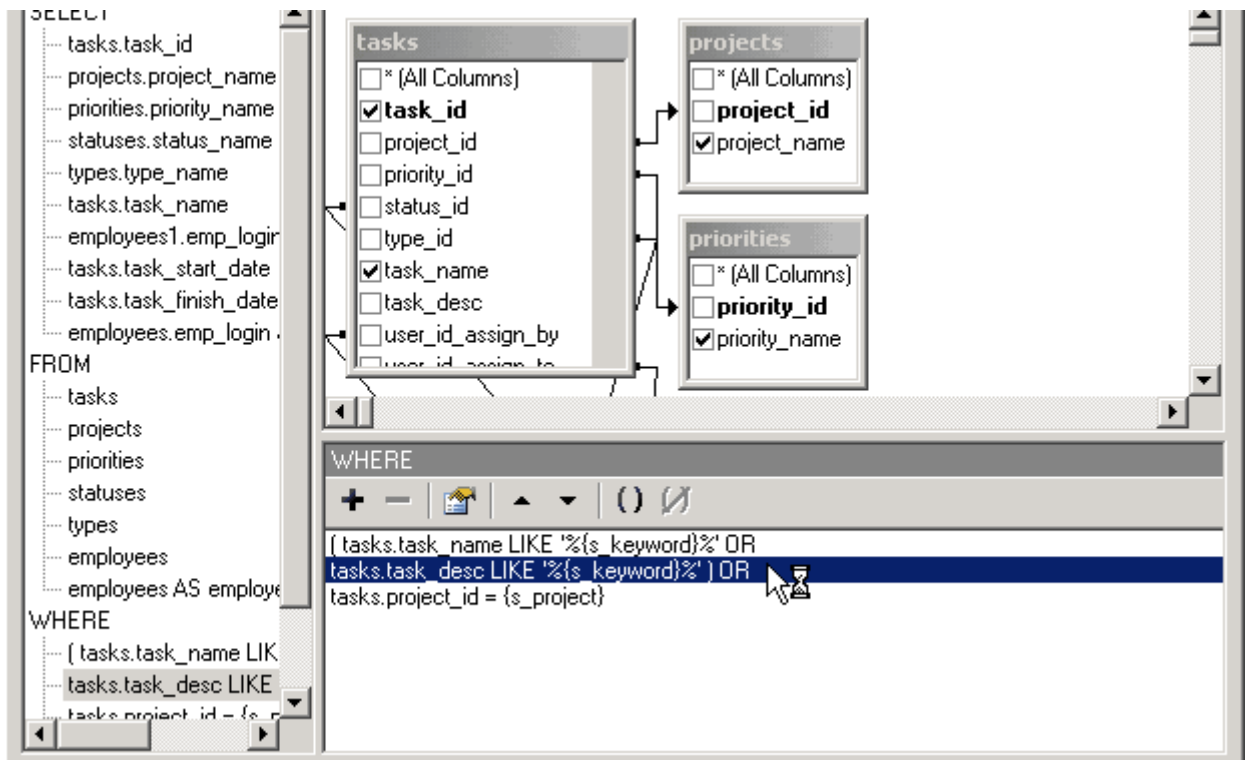
Filter Grid Records - Set AND Operator

Now that all search parameters are in place, the remaining task is to specify that the last keyword parameter should append the "AND" operator so that the full search parameters read as follows:

- *(task_name like '%{s_keyword}%' or*
- *task_desc like '%{s_keyword}%') and*
- *tasks.project_id equals (=) '{s_project}'*

To configure the operator:

1. Double-click on the task_desc parameter to open the **Table Parameter** window
2. Change the **Or** operator to **And**.



Next: [View the Working Page](#)

View the Working Page

Now your first page is complete. You can search and view the list of tasks as well as sort them, or click on a task to view more details about it.

1. On the generated page you can click on a Task Id for any of the tasks to test the Task Maintenance page.
2. Click on the **Generate Project** icon to generate the Project.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Search Tasks

Keyword

Project

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
2	CodeCharge	Highest	Closed	Fix ALL bugs	george

[Add New](#) 1 of 1

Next: [Login to the System](#)

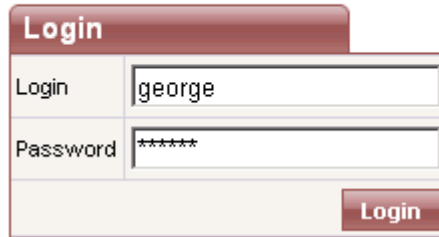
Login to the System

When you click on any of the tasks Ids on the task list page, you will be directed to the Login page where you can enter your login and password. This is because the record page for the *tasks* table (tasks_maint) was configured to allow only logged in users to access it.

1. Enter *george / george* to login as George Pennington.

Your entry will be stored in a session variable on the server, thus making it unnecessary to login again until your session expires.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)



Login

Login: george

Password: *****

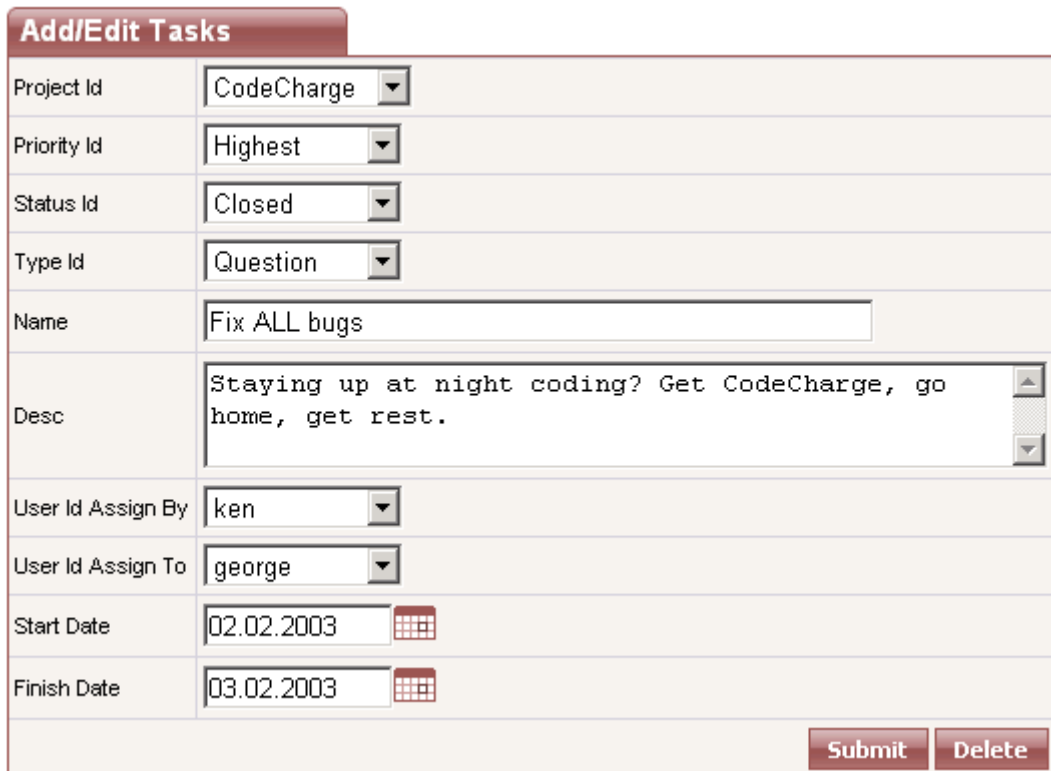
Login

Next: [Access Record Maintenance Page](#)

Access Record Maintenance Page

After selecting a task id on the task list page and going through the login page, you will arrive at the record maintenance page where you can view and update the task details. We shall now proceed to customize this page.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)



Add/Edit Tasks

Project Id: CodeCharge

Priority Id: Highest

Status Id: Closed

Type Id: Question

Name: Fix ALL bugs

Desc: Staying up at night coding? Get CodeCharge, go home, get rest.

User Id Assign By: ken

User Id Assign To: george

Start Date: 02.02.2003

Finish Date: 03.02.2003

Submit Delete

Next: [Changing Field Labels](#)

Step 4

Changing Field Labels

After using the Application Builder to generate the draft application and then adjusting the Task List page, we shall now make some customizations to the Task Maintenance page (tasks_maint).

The first order of business will be to change the labels for some of the fields to make them more meaningful.

1. In the **Project Explorer** window, double-click on the tasks_maint page to open it.
2. Proceed to change the field labels as follows:
 - *Project Id* to *Project*
 - *Priority Id* to *Priority*
 - *Status Id* to *Status*
 - *Type Id* to *Type*
 - *Desc* to *Description*
 - *User Id Assign By* to *Assigned By*
 - *User Id Assign To* to *Assigned To*

Header

Add/Edit Tasks

(Error)

Project	<input type="text" value="Select Value"/>
Priority	<input type="text" value="Select Value"/>
Status	<input type="text" value="Select Value"/>
Type	<input type="text" value="Select Value"/>
Name	<input type="text" value="{task_name}"/>
Description	<input style="height: 40px;" type="text" value="{task_desc}"/>
Assigned By	<input type="text" value="Select Value"/>
Assigned To	<input type="text" value="Select Value"/>
Start Date	<input type="text" value="{task_start_date}"/>
Finish Date	<input type="text" value="{task_finish_dat}"/>

Footer

Next: [Create Label Fields](#)

Create Label Fields

Some of the fields on the Task Maintenance page do not need to be updated manually, but could be updated automatically. For example if a person creates a new task, his name could be automatically entered as the creator of the task, along with the date and time when the task was created.

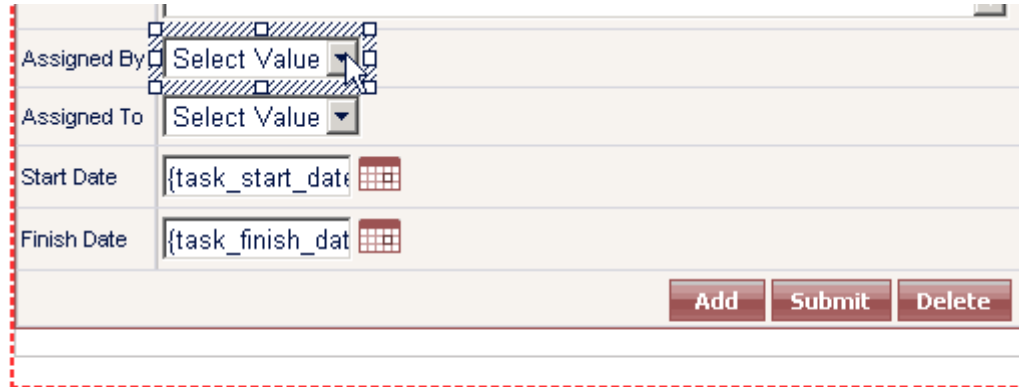
For now let's disable such fields by converting them to Label fields.

1. Right-click on the *user_id_assign_by* field.
2. Select **Change To > Label** .
3. Then do the same for the *task_start_date* field.

4. Since we have changed the *task_start_date* field to a Label field, we no longer need the Date Picker component that appears after the field. Label field do not allow user input so the Date Picker would not be useful.

Right-click on the Date Picker icon near the *DatePicker_task_start_date* label.

5. Select the **Delete Date Picker "DatePicker_task_start_date"** option.



The screenshot shows a form with four rows of fields. The first row is 'Assigned By' with a dropdown menu showing 'Select Value'. The second row is 'Assigned To' with a dropdown menu showing 'Select Value'. The third row is 'Start Date' with a text input field containing '{task_start_date}' and a calendar icon to its right. The fourth row is 'Finish Date' with a text input field containing '{task_finish_dat}' and a calendar icon to its right. At the bottom right of the form are three buttons: 'Add', 'Submit', and 'Delete'. A red dashed border highlights the 'Start Date' field and its calendar icon. A context menu is open over the calendar icon, showing options to delete the date picker.


Footer

Next: [Rearrange Label Fields](#)

Rearrange Label Fields

In order to facilitate easier input, we shall now move all the Label fields to the bottom of the form so that the fields that require input appear together at the top.

1. Place your cursor next to the *task_start_date* Label field.
2. Press the *Alt* and *Arrow Down* keyboard keys together to move the row downwards.
3. Repeat this procedure for the *user_id_assign_by* field to that it appears just above the *task_start_date* field.

Assigned By	user_id_assign_by
Assigned To	Select Value ▼
Start Date	task_start_date
Finish Date	{task_finish_dat 

Footer

Next: [Preview the Task Maintenance Page](#)

Preview the Task Maintenance Page

Congratulations! The basic version of your Task Management system is now ready.

1. Preview the working Task Maintenance page by
 1. selecting the *tasks_list* page in **Project Explorer**
 2. then selecting one of the existing tasks.
2. You can also add a new task by clicking on "Add New" below the *tasks* grid on the main *tasks_list* page.

Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

You may notice that the value of the *Assigned By* field is shown as a numeric value. We shall later implement a relationship so that it displays names. The next section addresses how to do this as well as some other improvements.

The language implementations are described in the following sections as follows:

- [ASP and VBScript](#)
- [ASP.NET \(C#\)](#)
- [JSP](#)
- [PHP](#)
- [ColdFusion](#)
- [Perl](#)

See also

[Common Errors](#)

Enhancing Application Functionality with Programming Events

ASP and VBScript

Step 1

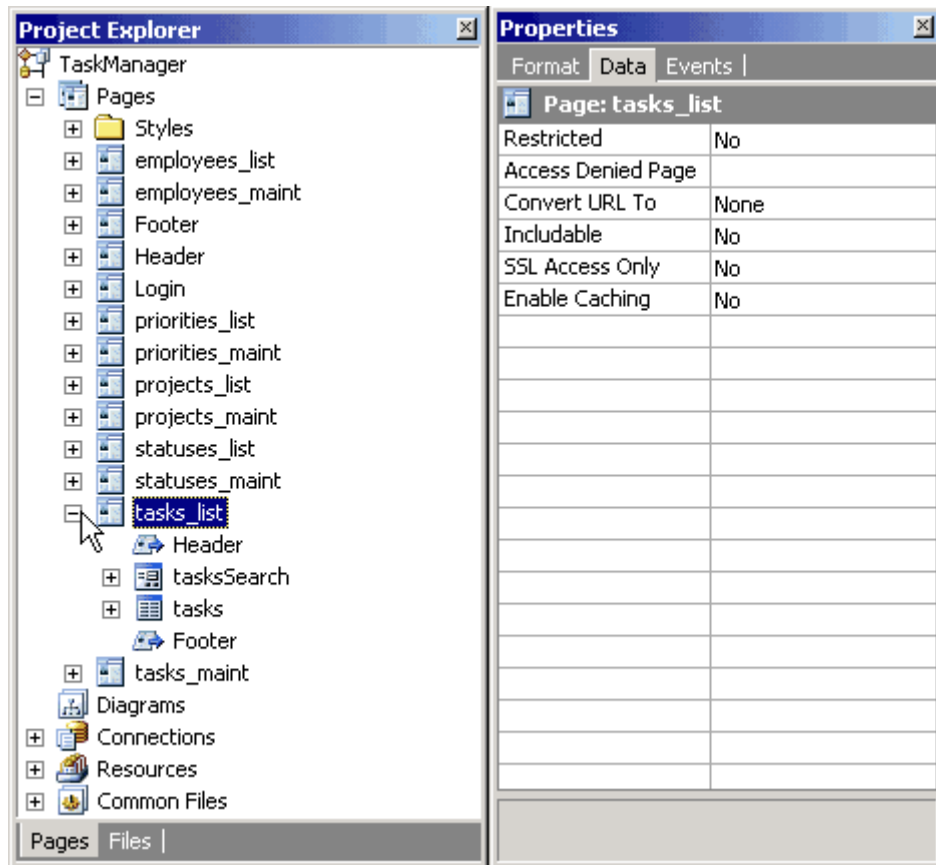
Use the Before Show Row Event to Alter Text Color

Let's start our basic programming with a simple task of altering the color of a grid field on our Task List page. To be more specific, we will mark the listed tasks assigned to you by showing your name in blue color within the grid.

1. Open the *tasks_list* page in the **Project Explorer**.
2. Expand the *tasks* grid.

3. In the **Project Explorer** right-click on the *task_name* field and select **Properties**.
4. Under the **Data** tab, set the value of the **Content** property to *HTML*.
5. Select *tasks* grid in the **Project Explorer**, or clicking anywhere within the form's area.
6. Select the **Events** tab in the **Properties** window.
7. Right-click on the **Before Show Row** event and select **Add Code...** .

The **Before Show Row** event occurs in the program after the field values are assigned, but before being output as HTML. By adding code into this event, you can modify the field value before it is shown.



Next: [Programmatically Control Field's Value](#)

Programmatically Control Field's Value

Once you add Custom Code to the Event, you will see the code-editing window with the appropriate place to enter the new code.

1. Replace this line of code:

```
' Write your own code here.
```

with the following lines (ASP/VBScript):

```
If tasks.user_id_assign_to.Value = Session("UserLogin") Then
    tasks.task_name.Value = "<b><font color=""blue"">" &
tasks.task_name.Value & "</font></b>"
End if
```

The following is an explanation of how the code added above works:

```
If tasks.user_id_assign_to.Value = Session("UserLogin") Then
```

This is an *if* condition that is true only if the value of *user_id_assign_to* label is equal to the login name of the employee that is currently logged into the system. Once you login to the system, the program will recognize your tasks by comparing your login name to the field value of the person that a task is assigned to.

UserLogin is one of the session variables used by CodeCharge-generated programs, and it holds the Login name of the currently logged in user until the session expires.

Note:

The following are all default session variables created by CodeCharge Studio:

- *UserID* - the primary key field value of the logged in user
- *UserLogin* - login name of the user currently logged into the system
- *GroupID* - security level/group of the user currently logged into the system

```
tasks.task_name.Value = "<b><font color=""blue"">" & tasks.task_name.Value & "</font></b>"
```

The above code is executed if the previous *if* condition is met. It modifies the value of the *task_name* field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML ** tag to make the font bold as well.

Notice that the word *blue* has double quotes around it, which will be replaced with a single quote. Since quotes mark the start and end of a string, using double quote allows you to insert a quote into a string.

Additionally, notice that the code is object-oriented and you specify that you want to assign a value to the *task_name* field in the *tasks* grid. *Value* is a property of that field, which you can both read and modify.

```
End if
```

This line marks the end of the *if* condition, so that the execution of the remaining code is not affected by this condition.

Next: [Preview Tasks List Page](#)

Preview Tasks List Page

1. Save your project.
2. Go to **Live Page** mode to view your working page. If the column "Name" doesn't have any task names highlighted, you are probably not logged in.
3. Since the menu doesn't contain a link to the Login page at this time, you can access it by trying to access one of the restricted pages, such as Task Maintenance.
4. Click on any of the project Ids and you should see the **Login** page.
5. Login as *george/george*, then click on the **Tasks** link on the menu to get back to the Task List page.

Now you should see some task names highlighted, which are the tasks of the user that you logged in as.

Search Tasks

Keyword

Project Select Value ▼

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

[Add New](#) 1 of 1

Next: [Modify a Label Field on the Task Maintenance Page](#)

Step 2

Modify a Label Field on the Task Maintenance Page

Now let's make one necessary modification on the Task Maintenance page where you might've noticed that the Label field *Assigned By* doesn't display the employee name, but the ID, as shown below. This is because the *tasks* table contains only the user ID, while the *employees* table contains the actual user names.

There are several potential methods of dealing with the above issue as explained below:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, just like you did with the grid on the Task List page. Unfortunately, queries that contain multiple tables may not be updateable by their nature, and thus your whole record form may stop working. In other words, if you specified that you want to use a query containing *tasks* and *employees* table in your record form, then if you assigned a task to someone else, the program wouldn't know if you wanted to update the *tasks* table with the new *employee_id*, or if you wanted to update the *employees* table and change employee's name.

Thus if you used multiple tables as a data source for the record form, you would also need to specify Custom Insert, Custom Update and Custom Delete operations in record form's properties, to specify which database fields should be updated with corresponding values entered on the page. This approach looks like too much effort just for displaying one additional value on the page.

2. Use an Event Procedure to insert custom code where you can programmatically output the desired value. This method is very flexible, as it allows you to extend the generated code by adding your own. The next step describes this method in detail.

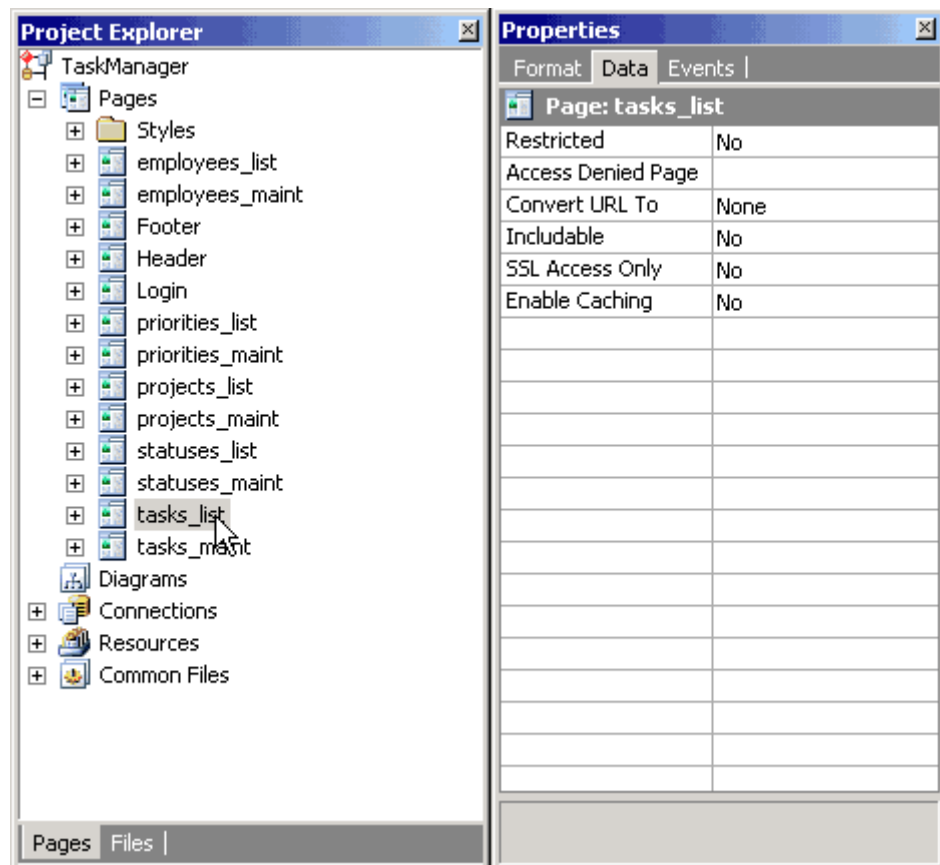
[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

Next: [Use the Before Show Event to Alter a Label's Value](#)

Use the Before Show Event to Alter a Label's Value

1. Select the Label *user_id_assign_by* in the **Project Explorer**.
2. In the **Properties** window click on the **Data** tab.
3. Select *Text* for the **Data Type** property.
4. In the **Properties** window click on the **Events** tab.
5. Right-click on **Before Show** event and select **Add Code....** .
CodeCharge Studio should then automatically switch to **Code** view.



6. Once in Code view, replace the following text:

```
'Write your own code here.'
```

with the following:

```
If tasks.EditMode Then
    tasks.user_id_assign_by.Value = CCDLookUp("emp_name",
"employees", "emp_id=" &_
        DBIntranetDB.ToSQL(tasks.user_id_assign_by.Value,
ccsInteger), DBIntranetDB)
Else
    tasks.user_id_assign_by.Value = CCDLookUp("emp_name",
"employees", "emp_id=" &_
        DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger),
DBIntranetDB)
End if
```

The above code consists of the following elements:

- *tasks* -the name of the record form on the page
- *EditMode* - property of the form, which specifies if the record is being edited. Depending on the value of this property, we either display the name of the person who originally submitted the task (Edit mode), or the person who is currently submitting the task (Insert mode).
- *user_id_assign_by* - the name of the Label within the Grid, and at the same time the name of the database field that was used to create this Label and which is now its data source.

- *Value* - the property of an object (in this case the Label), which can be read and/or modified.
- *tasks.user_id_assign_by.Value* - fully qualified *Value* property, which tells the program which object it belongs to. In other words, it is the *Value* property that belongs to *user_id_assign_by* field, which in turn belongs to the *tasks* Grid.
- *CCDLookup* - CodeCharge function that supports retrieving database value based on a field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table using the condition that the key (*emp_id*) equals the current value of the Label.
- *DBIntranetDB* - the name of the object that defines the database connection that you want to use in the *CCDLookup* function.
- *ToSQL* - Connection property that converts a value into the format supported by the database. This property requires a parameter that tells it if a value should be converted to a number (*ccsInteger*, *ccsFloat*), text (*ccsText*, *ccsMemo*), date (*ccsDate*), or boolean (*ccsBoolean*). In this case, this property converts the current Label value to a number that can be used with the *CCDLookup* function. It is advisable to always use this property together with *CCDLookup*.
- *CCGetUserID* - CodeCharge function that returns the ID of the user that is currently logged in.

The whole code reads approximately as follows:

- *If a record is being edited:* Assign the name of the person who originally submitted the issue to the *user_id_assign_by* Label, by looking up employee's name from *employees* table using *CCDLookup* function that uses the *IntranetDB* connection and the value of the *user_id_assign_by* Label.
- *If a new record is being created:* Assign current user to the *user_id_assign_by* Label by retrieving his/her name from *employees* table using *CCDLookup* function that uses the *IntranetDB* connection and *CCGetUserID* function that obtains the current user's ID.

Next: [Add a Hidden "Assigned By" Field to Auto-Update New Tasks](#)

Step 3

Add a Hidden "Assigned By" Field to Auto-Update New Tasks

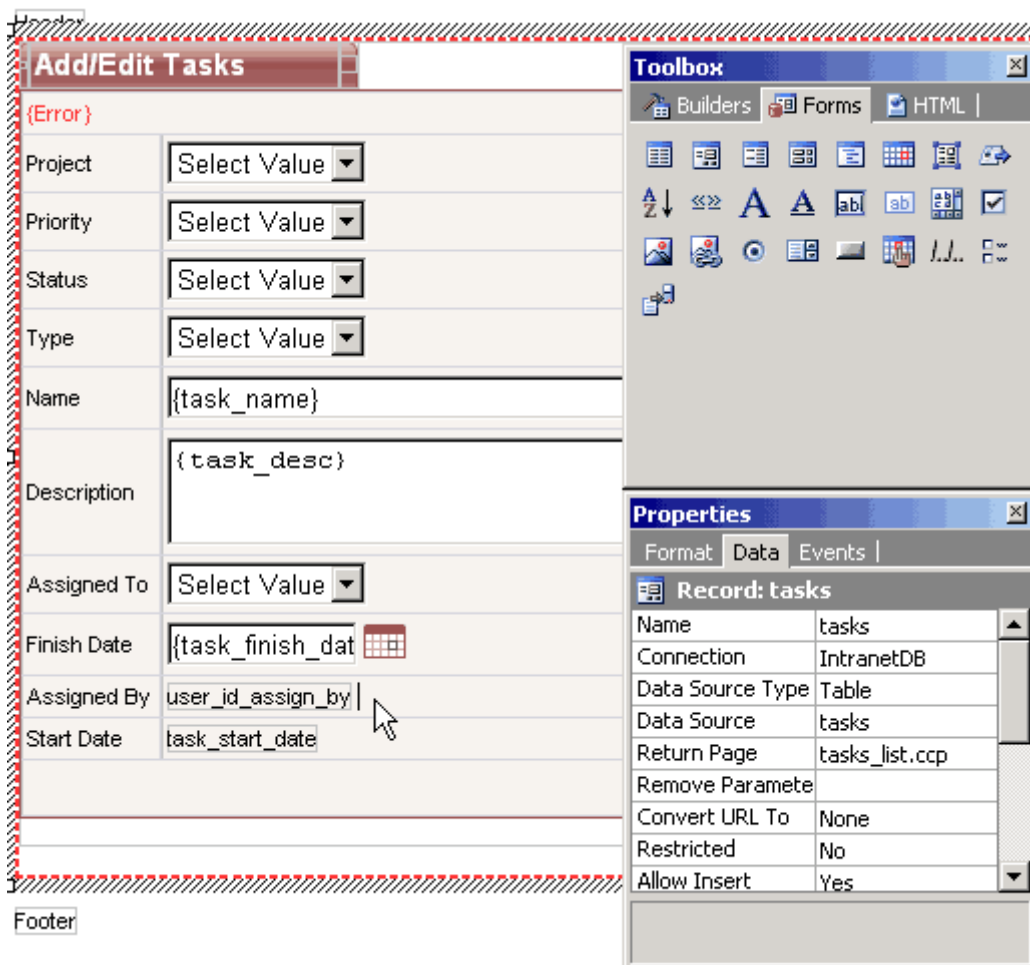
You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though the employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.

1. Add a Hidden field to your page from the **Forms** tab of the **Toolbox**.

This field type isn't visible in the browser, but will be used to store a value and update the database.

2. Configure the new field by setting its properties as follows:

- **Name:** *assigned_by* - the name of the newly added Hidden field. This can be any name you choose.
- **Control Source:** *user_id_assign_by* - the database field/column that will be used to retrieve field's value and will be updated with the new value, if it changes.
- **Data Type:** *Integer* - the type of the value bound to the control source. Our user/employee ID's are numeric.
- **Default:** *CCGetUserID* - *CCGetUserID* is a CodeCharge function that retrieves the ID of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.



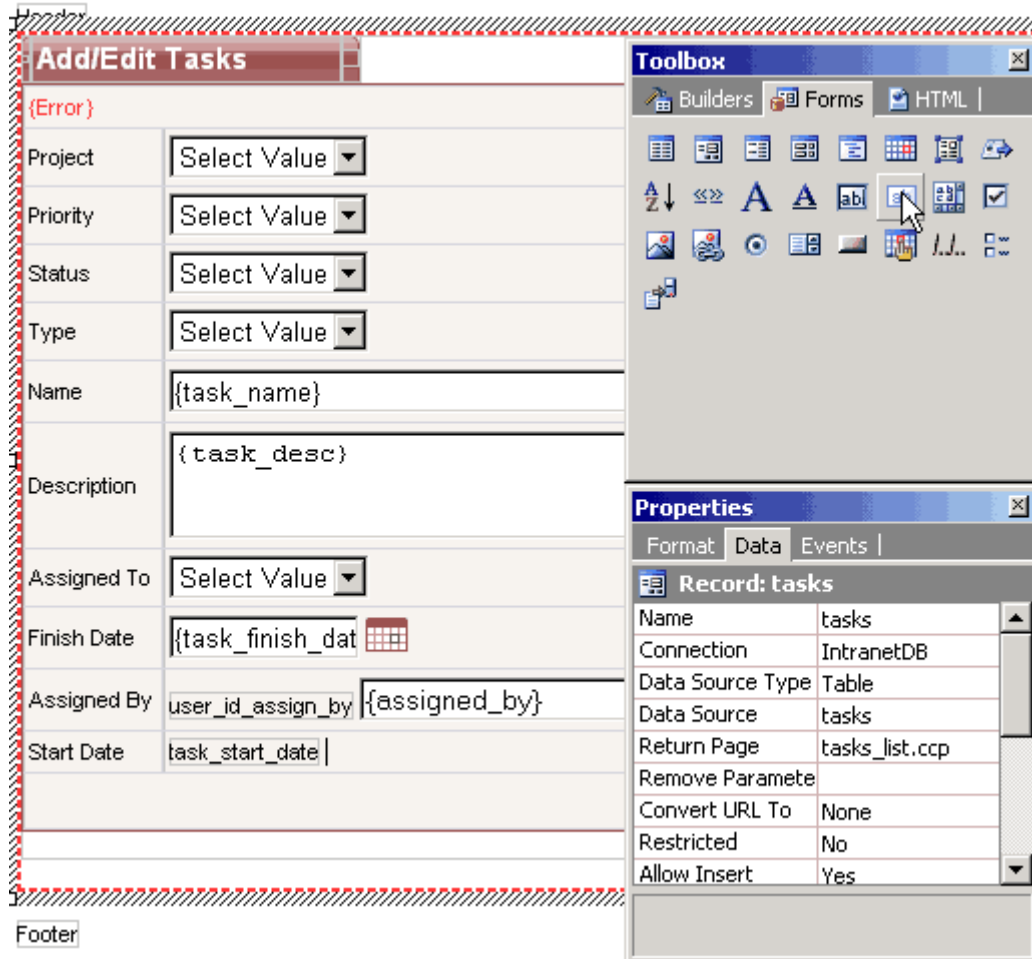
Next: [Add a Hidden "Date Created" Field to the Record Form](#)

Add a Hidden "Date Created" Field to the Record Form

Now add another Hidden field to your page, which will be used to submit the current date and time to the *task_start_date* field in the database.

1. Configure the new field as follows:
 - **Name:** *date_created*
 - **Control Source:** *task_start_date*
 - **Data Type:** *Date*

- **Default:** *CurrentDateTime* - The *CurrentDateTime* value allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.
2. Click on the *task_start_date* field.
 3. In the **Properties** window, set its Default value to *CurrentDateTime* just as you did with the *date_created* field. This is so that the Label field can display the date since the hidden field will not be visible to the user.




Next: [Test the Label and Hidden Fields](#)

Test the Label and Hidden Fields

Finally,

1. Switch to **Live Page** mode.
 2. Select or add a Task and see your Label display the name of the person who assigned the task.
- The basic version of your Task Manager is now completed.
3. Don't forget to save it!

Add/Edit Tasks	
Project	My Project
Priority	Highest
Status	In progress
Type	Task
Name	Finish my project
Description	Implement "Assigned By" Label to show values from another table
Assigned To	alexander
Finish Date	<input type="text"/> 
Assigned By	George Pennington
Start Date	13.09.2005 8:07:17
<input type="button" value="Add"/>	

Next: [Programming the Record Form](#)

Step 4

Programming the Record Form

Now you've created a simple task management application, but how do you extend it to be more practical and useful? In this section, you will get a glimpse of how to implement practical and sophisticated applications by adding programming code and actions that enhance the application's functionality.

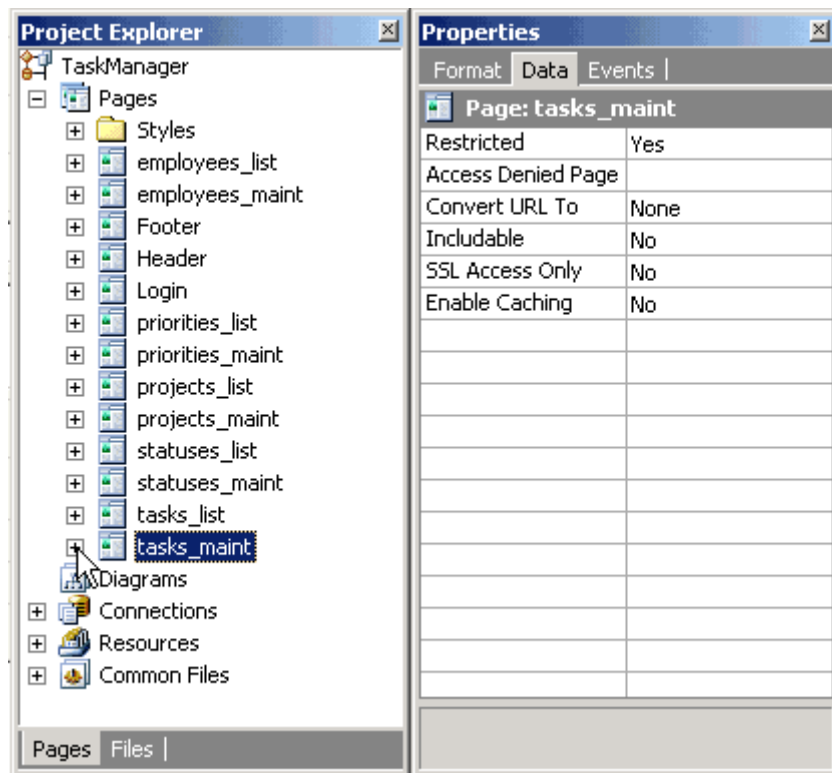
You will learn how to:

- Send email notifications to the person that the task is being assigned to
- Allow only the person assigned to the task to modify it

Next: [Add Code in the After Insert Event to Send Emails](#)

Add Code in the After Insert Event to Send Emails

1. Select the *tasks* form of the *tasks_maint* page by selecting it in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window click on the **Events** tab.
3. Select the **After Insert** event.
4. Click on the **[+]** button.
5. Select **Add Code...**



6. Once you are in the Code view, replace the generated comment:

```
' Write your own code here.
```

with the code below:

```
Dim Mailer
    Set Mailer = Server.CreateObject("Persits.MailSender")
    Mailer.From = CCDLookUp("email", "employees", "emp_id=" &_
        DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger),
        DBIntranetDB)
    Mailer.FromName = CCDLookUp("emp_name", "employees",
        "emp_id=" &_
        DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger),
        DBIntranetDB)
    Mailer.AddAddress CCDLookUp("email", "employees", "emp_id="
    &_
        DBIntranetDB.ToSQL(tasks.user_id_assign_to.Value,
        ccsInteger), DBIntranetDB)
    Mailer.Host = "mysmtphost.com"
    Mailer.IsHTML = True
    Mailer.Subject = "New task for you"
    Mailer.Body = "The following task was submitted:<br><br>" &_
        "Task ID: " & CCDLookUp("max(task_id)", "tasks",
        "user_id_assign_by=" &_
        DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger),
        DBIntranetDB) &_
        "<br><br>" & tasks.task_desc.Text
    Mailer.Send
    set Mailer = Nothing
```

As you may have realized by now, the above code sends emails to users to whom the new tasks are assigned. Here is additional information you should be aware of:

1. The above code requires that you install on your server the free Email component "ASPEmail", which you can obtain from <http://www.aspemail.com/>. There are many other email components and you can modify the above program by reading the documentation covering the component you choose to use.
2. You need to replace the parameter "mysmtphost.com" with a SMTP server that you are authorized to use. This usually may be the same server that you configure as "Outgoing Mail Server (SMTP)" in your email client, like MS Outlook or Outlook Express.

The following is an explanation of the above code:

```
Dim Mailer
```

Defines the Mailer object, which later will initialize the ASPEmail component.

```
Set Mailer = Server.CreateObject("Persits.MailSender")
```

Creates Mailer object and initializes the ASPEmail component.

```
Mailer.From = CCDLookUp("email", "employees", "emp_id=" &  
DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger), DBIntranetDB)
```

Sets the *From* email address to the value of the *email* field in the *employees* table where *emp_id* matches the current user. The CCDLookUp function is used to retrieve a database value, while CCGetUserID retrieves the id of the currently logged in user.

```
Mailer.FromName = CCDLookUp("emp_name", "employees", "emp_id=" &  
DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger), DBIntranetDB)
```

Sets the *From* name to the value of the *emp_name* field for the current user.

```
Mailer.AddAddress CCDLookUp("email", "employees", "emp_id=" &  
DBIntranetDB.ToSQL(tasks.user_id_assign_to.Value, ccsInteger),  
DBIntranetDB)
```

Sets the *To* email address to the email of the person that is assigned to the task. The CCDLookUp function is used here to retrieve the appropriate email address.

```
Mailer.Host = "mysmtphost.com"
```

Specifies the SMTP server that will be sending the email. (Replace this value with an SMTP host that you are authorized to use)

```
Mailer.IsHTML = True
```

Specifies that the email will be sent in HTML format (as opposed to plain text).

```
Mailer.Subject = "New task for you"
```

The subject of the email to be sent.

```
Mailer.Body = "The following task was submitted:<br><br>" & "Task ID:  
& CCDLookUp("max(task_id)", "tasks", "user_id_assign_by=" &
```

```
DBIntranetDB.ToSQL(CCGetUserID(), ccsInteger), DBIntranetDB) &  
"<br><br>" & tasks.task_desc.Text
```

The body of the email which consists of the task description and the task id. The last inserted task id can be obtained using different methods with different databases. Unfortunately, MS Access doesn't support the retrieval of the last inserted record, therefore you will need to use the CCDLookUp function to retrieve the largest task id submitted by the current user (assuming that task ids are created incrementally).

```
Mailer.Send
```

Sends the email.

```
set Mailer = Nothing
```

Disposes of the Mailer object to free computer resources.

Next: [Use the After Update Event to Send Emails](#)

Use the After Update Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. We shall now implement similar functionality in **After Update** Event to notify the assignee when an existing task is updated and reassigned to someone.

1. Click on the *tasks* form in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window, select the **Events** tab.
3. Add the following **Custom Code** in **After Update** event:

```
4.         Dim Mailer  
5.         If CCGetUserID() <> tasks.user_id_assign_to.Value Then  
6.             Set Mailer =  
Server.CreateObject("Persits.MailSender")  
7.             Mailer.From = CCDLookUp("email", "employees",  
"emp_id=" &  
8.                 DBIntranetDB.ToSQL(CCGetUserID(),  
ccsInteger), DBIntranetDB)  
9.             Mailer.FromName = CCDLookUp("emp_name", "employees",  
"emp_id=" &  
10.                DBIntranetDB.ToSQL(CCGetUserID(),  
ccsInteger), DBIntranetDB)  
11.            Mailer.AddAddress CCDLookUp("email", "employees",  
"emp_id=" &  
12.                DBIntranetDB.ToSQL(tasks.user_id_assign_to.Value, ccsInteger),  
DBIntranetDB)  
13.            Mailer.Host = "mysmtphost.com"  
14.            Mailer.IsHTML = True  
15.            Mailer.Subject = "A task was assigned to you"  
16.            Mailer.Body = "The following task was assigned to  
you:<br><br>" &  
17.                "Task ID: " & CCGetParam("task_id", Empty)& _
```

```

18.             "<br><br>" & tasks.task_desc.Text
19.             Mailer.Send
20.             set Mailer = Nothing

End if

```

The main differences between the above code and that which was used in the **After Insert** event are as follows:

1. An *if* condition was added to send an email only if a user assigns a task to someone other than himself/herself.
2. *task_id* is retrieved from the URL using the Request.QueryString function. We can use this method because tasks can be updated only if the user arrived at the current page via a URL that contains the task id to be updated. Such a URL would look like this:
http://localhost/TaskManager/tasks_maint.asp?task_id=9

Next: [Test Email Delivery](#)

Test Email Delivery

Before testing the system:

1. You should add new users to your database with correct email addresses, or modify the existing users by changing their email address.
 - a. You can do this by opening the Intranet.mdb database that is located in your project directory.
 - b. Alternatively, you may use the Task Manager itself. Go to the Employees page to view and modify user emails there.
2. Once you have users configured with test emails, save your project and switch to **Live Page** mode to test your system.

Note: You will need MS Access 2000 or higher to manually edit the database file. If your email code worked correctly, you should end up back at the Task List page after adding or modifying a task, and an email should be delivered to the person to whom the task was assigned.

Next: [Implement Record Security in After Initialize Event](#)

Step 5

Implement Record Security in After Initialize Event

Your Task Management system is now almost complete, except one possibly important feature- security. Currently everyone can modify and delete any of the tasks. You may want to limit the access so that only the employee assigned to as task can update their tasks. There are many ways of accomplishing this, and we will examine several of them.

1. Click on the *tasks_maint* page in the **Project Explorer**.
2. Select **Events** tab in the **Properties** window.
3. Add **Custom Code** to the **After Initialize** event of the page as follows:
4. Once in the **Code** mode, replace the generated comment:


```
' Write your own code here.
```

with the code below:

```
Dim current_task
Dim assigned_user
    current_task = CCGetParam("task_id", Empty)
    If IsNumeric(current_task) Then
        assigned_user = CCDLookUp("user_id_assign_to", "tasks",
"task_id=" &_
                                DBIntranetDB.ToSQL(current_task, ccsInteger),
DBIntranetDB)
        If CInt(current_task) <> 0 and CInt(CCGetUserID()) <>
CInt(assigned_user) Then
            tasks.Visible = False
            ' Redirect = "tasks_list.asp"
            ' tasks.UpdateAllowed = False
            ' tasks.DeleteAllowed = False
        End if
    End if
```

The above code allows you to test the following methods of implementing record security:

Do not show the Task (record form) on the page if the selected task doesn't belong to the current user. An unauthorized user should see a blank page.

You can hide any form on a page by assigning a *False* value to the *Visible* property of the form. The code *current_task <>0* in the *if* condition specifies that the code should be executed only if a user tries to modify an existing task and he/she is not assigned to it. The *if* also assures that all users can create new tasks. You can test this functionality by inserting the above code into the event, then switching to **Live Page** mode and trying to modify a task that is not assigned to you, in which case you should see an empty page. Although such functionality may not be very useful, it shows how you can hide forms on a page. You may consider adding another record form to your page that is not updateable and has just the Label fields that show information. Once you have two forms on the page, you can hide each form programmatically using mutually exclusive criteria.

Redirect unauthorized users to another page. Only users who are assigned to a task can view the page.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
Dim current_task
Dim assigned_user
    current_task = CCGetParam("task_id", Empty)
    If IsNumeric(current_task) Then
        assigned_user = CCDLookUp("user_id_assign_to", "tasks",
"task_id=" &_
                                DBIntranetDB.ToSQL(current_task, ccsInteger),
DBIntranetDB)
        If CInt(current_task) <> 0 and CInt(CCGetUserID()) <>
CInt(assigned_user) Then
            Redirect = "tasks_list.asp"
            UpdateAllowed = False
            DeleteAllowed = False
        End if
    End if
```

```

        DBIntranetDB.ToSQL(current_task, ccsInteger),
DBIntranetDB)
    If CInt(current_task) <> 0 and CInt(CCGetUserID()) <>
CInt(assigned_user) Then
        ' tasks.Visible = False
        Redirect = "tasks_list.asp"
        ' tasks.UpdateAllowed = False
        ' tasks.DeleteAllowed = False
    End if
End if

```

The above code shows that you should comment out the previously active line, and uncomment the line that starts with *Redirect*. *Redirect* is a variable used by CodeCharge Studio to determine if the current page should be redirected to another page, for example if a user is not logged in. This variable can be used only on pages that have restricted access and require users to login. You can simply assign the destination page to the *Redirect* variable and the page will be automatically redirected. Test this functionality by modifying the code as shown then switch to **Live Page** mode and try to modify a task that is not assigned to you.

Disallowed Update and Delete operations for unauthorized users. Only users who are assigned to a task can edit (delete) it.

You can implement and test this functionality by slightly modifying the above code as shown below:

```

Dim current_task
Dim assigned_user
    current_task = CCGetParam("task_id", Empty)
    If IsNumeric(current_task) Then
        assigned_user = CCDLookUp("user_id_assign_to", "tasks",
"task_id=" &_
        DBIntranetDB.ToSQL(current_task, ccsInteger),
DBIntranetDB)
    If CInt(current_task) <> 0 and CInt(CCGetUserID()) <>
CInt(assigned_user) Then
        ' tasks.Visible = False
        ' Redirect = "tasks_list.asp"
        tasks.UpdateAllowed = False
        tasks.DeleteAllowed = False
    End if
End if

```

This code shows how you can manipulate the *UpdateAllowed* and *DeleteAllowed* properties of a record form. These properties control record update/delete operations execution. If set to *false*, the operation will not be executed.

These properties control also the visibility of the **Update** and **Delete** buttons on the page.

Next: [Conclusion](#)

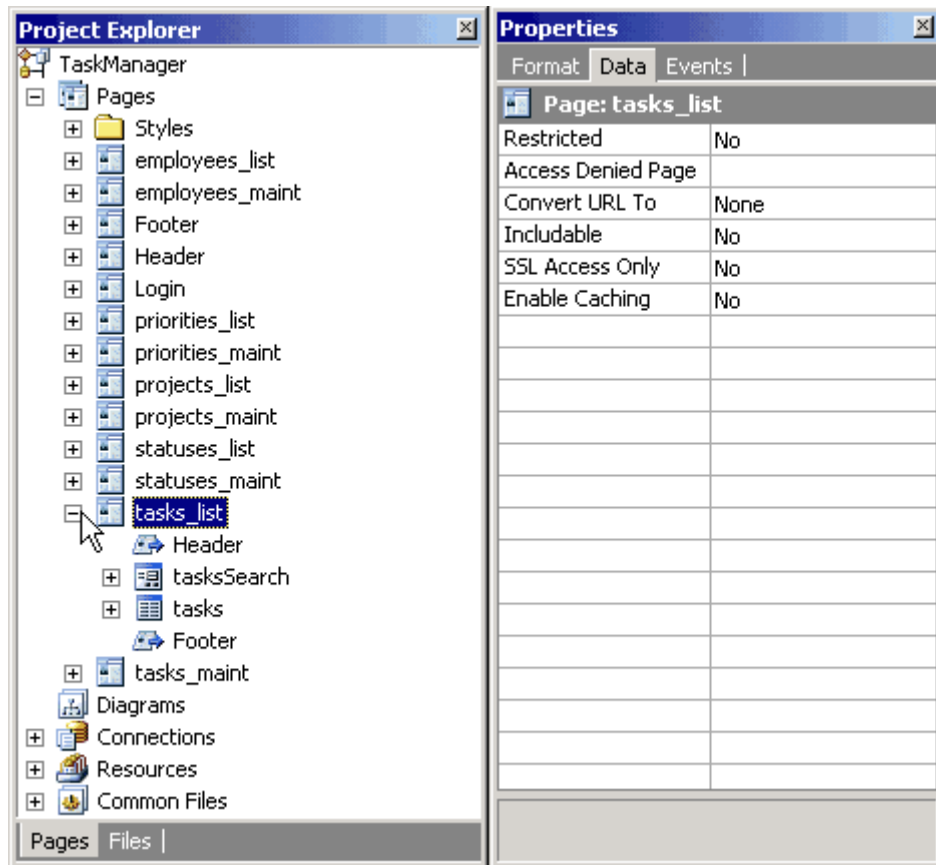
PHP

Step 1

Let's start our basic programming with a simple task of altering the color of a grid field on our Task List page. To be more specific, we will mark the listed tasks assigned to you by showing your name in blue color within the grid.

1. Open the *tasks_list* page in the **Project Explorer**.
2. Expand the *tasks* grid.
3. In the **Project Explorer** right-click on the *task_name* field and select **Properties**.
4. Under the **Data** tab, set the value of the **Content** property to *HTML*.
5. Select *tasks* grid in the **Project Explorer**, or clicking anywhere within the form's caption.
6. Select the **Events** tab in the **Properties** window.
7. Right-click on the **Before Show Row** event and select **Add Code...** .

The **Before Show Row** event occurs in the program after the field values are assigned, but before being output as HTML. By adding code into this event, you can modify the field value before it is shown.



Next: [Programmatically Control Field's Value](#)

Programmatically Control Field's Value

Once you add Custom Code to the Event, you will see the code-editing window with the appropriate place to enter the new code.

1. Replace this line of code:

```
///Write your own code here.
```

with the following lines (PHP):

```
global $tasks;
    if (strlen(CCGetSession("UserLogin")) && $tasks->user_id_assign_to->GetValue() == CCGetSession("UserLogin"))
    {
        $tasks->task_name->SetValue("<b><font color=\"blue\">" .
        $tasks->task_name->GetValue(). "</font></b>");
    }
}
```

The following is how the above PHP code works:

```
if (strlen(CCGetSession("UserLogin")) && $tasks->user_id_assign_to->GetValue() == CCGetSession("UserLogin"))
```

This is an *if* condition that is true only if the value of the *user_id_assign_to* field is equal to the login name of the employee that is currently logged into the system. The *employees1_emp_login* database field provides the value for the *user_id_assign_to* field in the grid form. Once you login to the system, the program will recognize your tasks by comparing your login name to the *emp_login* value of the person that a task is assigned to. *UserLogin* is one of the session variables used by CodeCharge-generated programs, and it holds the Login name of the currently logged in user until the session expires.

Note:

The following are all default session variables created by CodeCharge Studio:

- *UserID* - the primary key field value of the logged in user
- *UserLogin* - login name of the user currently logged into the system
- *GroupID* - security level/group of the user currently logged into the system

```
$tasks->task_name->SetValue("<b><font color=\"blue\">" . $tasks->task_name->GetValue(). "</font></b>");
```

This code is executed if the previous *if* condition is met. It modifies the value of the *task_name* field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML ** tag to make the font bold as well. Additionally, notice that the code is object-oriented and you specify that you want to assign a value to the *task_name* field in the *tasks* grid. *SetValue* is a method of an object, which can be used to modify the object's value.

Next: [Preview Tasks List Page](#)

Preview Tasks List Page

1. Save your project.
2. Go to **Live Page** mode to view your working page. If the column "Name" doesn't have any task names highlighted, you are probably not logged in.
3. Since the menu doesn't contain a link to the Login page at this time, you can access it by trying to access one of the restricted pages, such as Task Maintenance.
4. Click on any of the project Ids and you should see the **Login** page.
5. Login as *george/george*, then click on the **Tasks** link on the menu to get back to the Task List page.

Now you should see some task names highlighted, which are the tasks of the user that you logged in as.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Search Tasks

Keyword

Project

Select Value
▼

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

[Add New](#) 1 of 1

Next: [Modify a Label Field on the Task Maintenance Page](#)

Step 2

Modify a Label Field on the Task Maintenance Page

Now let's make one necessary modification on the Task Maintenance page where you might've noticed that the Label field *Assigned By* doesn't display the employee name, but the ID, as shown below. This is because the *tasks* table contains only the user ID, while the *employees* table contains the actual user names.

There are several potential methods of dealing with the above issue as explained below:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, just like you did with the grid on the Task List page. Unfortunately, queries that contain multiple tables may not be updateable by their nature, and thus your whole record form may stop working. In other words, if you specified that you want to use a query containing *tasks* and *employees* table in your record form, then if you assigned a task to someone else, the program wouldn't know if you wanted to update the *tasks* table with the new *employee_id*, or if you wanted to update the *employees* table and change employee's name.

Thus if you used multiple tables as a data source for the record form, you would also need to specify Custom Insert, Custom Update and Custom Delete operations in record form's properties, to specify which database fields should

be updated with corresponding values entered on the page. This approach looks like too much effort just for displaying one additional value on the page.

2. Use an Event Procedure to insert custom code where you can programmatically output the desired value. This method is very flexible, as it allows you to extend the generated code by adding your own. The next step describes this method in detail.

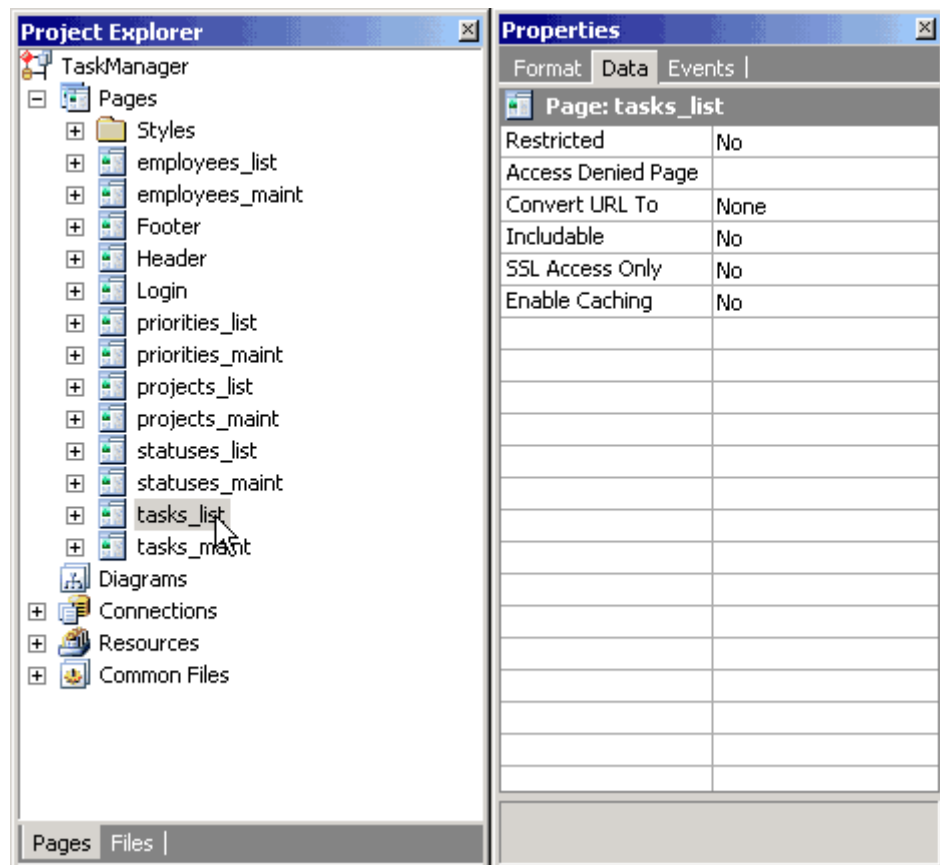
[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

Next: [Use the Before Show Event to Alter a Label's Value](#)

Use the Before Show Event to Alter a Label's Value

1. Select the Label `user_id_assign_by` in the **Project Explorer**.
2. In the **Properties** window click on the **Data** tab.
3. Select `Text` for the **Data Type** property.
4. In the **Properties** window click on the **Events** tab.
5. Right-click on **Before Show** event and select **Add Code....** .
CodeCharge Studio should then automatically switch to **Code** view.



6. Once in Code view, if you generate PHP, you should see the *tasks_maint_events.php* file, with the following lines of code:

```

7.      //Custom Code @28-73254650
8.      //-----
9.          //Write your own code here.
10.     //-----

```

```
//End Custom Code
```

Replace the text:

```
// Write your own code here.
```

with the following:

```

global $tasks;
global $DBIntranetDB;
if ($tasks->EditMode)
{
    $tasks->user_id_assign_by->SetValue(CCDLookUp("emp_name",
"employees", " emp_id=");
    $DBIntranetDB->ToSQL($tasks->user_id_assign_by->GetValue(),
ccsInteger), $DBIntranetDB));
}
else
{
    $tasks->user_id_assign_by->SetValue(CCDLookUp("emp_name",
"employees", " emp_id=");

```

```
$DBIntranetDB->ToSQL(CCGetUserID(), ccsInteger),
$DBIntranetDB));
}
```

The above code consists of the following elements:

- *tasks* - the name of the record form on the page
- *EditMode* - property of the form, which specifies if the record is being edited. Depending on the value of this property, we either display the name of the person who originally submitted the task (Edit mode), or the person who is currently submitting the task (Insert mode).
- *user_id_assign_by* - the name of the Label within the Grid, and at the same time the name of the database field that was used to create this Label and which is now its data source.
- *SetValue* - a method of an object (in this case the Label), which can be used to modify the object's value.
- *\$tasks->user_id_assign_by->SetValue* - fully qualified *SetValue* method, which tells the program which object it refers to. In other words, it is the *SetValue* method that affects *user_id_assign_by* field, which in turn belongs to the *tasks* grid.
- *CCDLookup* - CodeCharge function that supports retrieving a database value based on a field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table using the condition that the key (*emp_id*) equals the current value of the Label.
- *ToSQL* - CodeCharge function that converts a value into the format supported by the database. This function requires a parameter that tells it if a value should be converted to a number (Integer) or text. In this case, this function converts the current Label value to a number that can be used with the *CCDLookup* function. It is advisable to always use this function together with *CCDLookup*.
- *\$DBIntranetDB* - the name of the object that defines the database connection that you want to use in *CCDLookup* function.
- *CCGetUserID()* - CodeCharge function that returns the ID of the user that is currently logged in.

The whole code reads approximately as follows:

- If a record is being edited: Assign the name of the person who originally submitted the issue to the *user_id_assign_by* Label, by looking up employee's name from *employees* table using *CCDLookup* function that uses the *IntranetDB* connection and the value of the *user_id_assign_by* Label.
- If a new record is being created: Assign current user to the *user_id_assign_by* Label by retrieving his/her name from *employees* table using *CCDLookup* function that uses *IntranetDB* connection and *CCGetUserID* function that obtains current user's ID.

Next: [Add a Hidden "Assigned By" Field to Auto-Update New Tasks](#)

Step 3

Add a Hidden "Assigned By" Field to Auto-Update New Tasks

You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though the employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.

1. Add a Hidden field to your page from the **Forms** tab of the **Toolbox**.

This field type isn't visible in the browser, but will be used to store a value and update the database.

2. Configure the new field by setting its properties as follows:

- **Name:** *assigned_by* - the name of the newly added Hidden field. This can be any name you choose.
- **Control Source:** *user_id_assign_by* - the database field/column that will be used to retrieve field's value and will be updated with the new value, if it changes.
- **Data Type:** *Integer* - the type of the value bound to the control source. Our user/employee ID's are numeric.
- **Default:** *CCGetUserID()* - *CCGetUserID()* is a CodeCharge function that retrieves the ID of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.

The screenshot shows a web application interface with a form titled "Add/Edit Tasks" and a toolbox. The form contains several fields: Project, Priority, Status, Type, Name, Description, Assigned To, Finish Date, Assigned By, and Start Date. The "Assigned By" field is set to "user_id_assign_by". The toolbox is open to the "Forms" tab, and a hidden field is visible in the "Properties" window. The "Properties" window shows the following settings for the "Record: tasks":

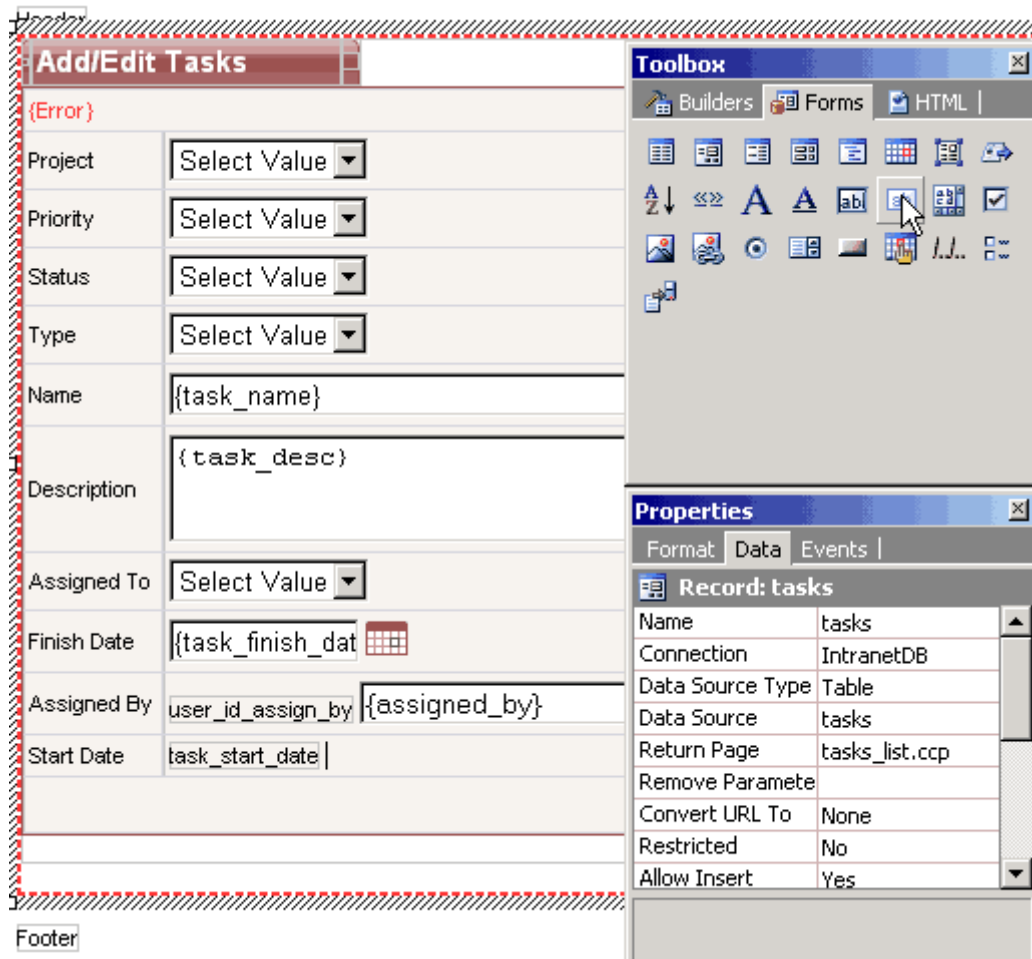
Record: tasks	
Name	tasks
Connection	IntranetDB
Data Source Type	Table
Data Source	tasks
Return Page	tasks_list.ccp
Remove Paramete	
Convert URL To	None
Restricted	No
Allow Insert	Yes

Next: [Add a Hidden "Date Created" Field to the Record Form](#)

Add a Hidden "Date Created" Field to the Record Form

Now add another Hidden field to your page, which will be used to submit the current date and time to the *date_assign* field in the database.

1. Configure the new field as follows:
 - o **Name:** *date_created*
 - o **Control Source:** *task_start_date*
 - o **Data Type:** *Date*
 - o **Default:** *CurrentDateTime* - The *CurrentDateTime* value allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.
2. Click on the *task_start_date* field.
3. In the **Properties** window, set its Default value to *CurrentDateTime*. This is so that the label field can display the date since the hidden field will not be visible to the user.



Next: [Test the Label and Hidden Fields](#)

Test the Label and Hidden Fields

Finally,


1. Switch to **Live Page** mode.

2. Select or add a Task and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.

3. Don't forget to save it!

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Add/Edit Tasks	
Project	My Project
Priority	Highest
Status	In progress
Type	Task
Name	Finish my project
Description	Implement "Assigned By" Label to show values from another table
Assigned To	alexander
Finish Date	<input type="text"/> 
Assigned By	George Pennington
Start Date	13.09.2005 8:07:17
<input type="button" value="Add"/>	

Next: [Programming the Record Form](#)

Step 4

Programming the Record Form

Now you've created a simple task management application, but how do you extend it to be more practical and useful? In this section, you will get a glimpse of how to implement practical and sophisticated applications by adding programming code and actions that enhance the application's functionality.

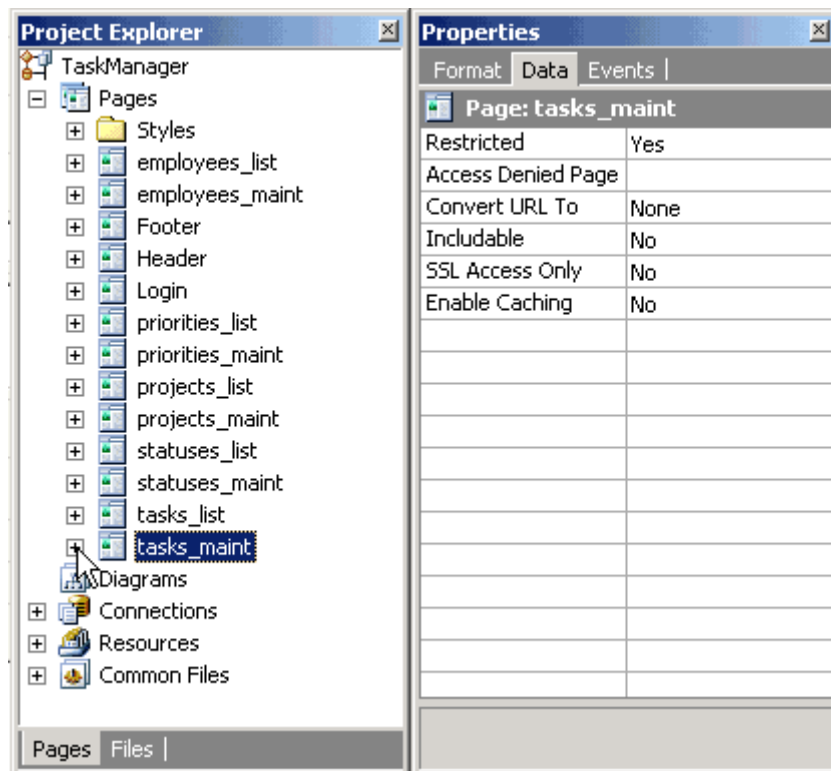
You will learn how to:

- Send email notifications to the person that the task is being assigned to.
- Allow only the person assigned to the task to modify it

Next: [Add Code in the After Insert Event to Send Emails](#)

Add Code in the After Insert Event to Send Emails

1. Select the *tasks* form by selecting it in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window click on the **Events** tab.
3. Select the **After Insert** event.
4. Click on the **[+]** button.
5. Select **Add Code...**



6. Once you are in the **Code** mode, replace the generated comment:

```
// Write your own code here.
```

with the code below:

```
global $DBIntranetDB;
global $tasks;
$from_name = CCDLookUp("emp_name", "employees", "emp_id=".
    $DBIntranetDB->ToSQL(CCGetUserID(), ccsInteger),
    $DBIntranetDB);
$from_email = CCDLookUp("email", "employees", "emp_id=".
    $DBIntranetDB->ToSQL(CCGetUserID(), ccsInteger),
    $DBIntranetDB);
$to_email = CCDLookUp("email", "employees", "emp_id=".
    $DBIntranetDB->ToSQL($tasks->user_id_assign_to-
    >GetValue(), ccsInteger), $DBIntranetDB);

$headers = "From: ".$from_name."<".$from_email.">\r\n";
$headers .= "Content-Type: text/html";
$subject = "New task for you";
$message = "The following task was submitted:<br><br>".
    "Task ID: ".CCDLookUp("max(task_id)", "tasks",
    "user_id_assign_by=".
    $DBIntranetDB->ToSQL(CCGetUserID(), ccsInteger),
    $DBIntranetDB)."<br><br>".
    $tasks->task_desc->GetText();

mail ($to_email,$subject,$message,$headers);
```

As you may have realized by now, the above code sends emails to users to

whom the new tasks are assigned.

The following is an explanation of the code.

```
$from_name = CCDLookUp("emp_name", "employees",  
"emp_id=".$DBIntranetDB->ToSQL (CCGetUserID(),  
ccsInteger), $DBIntranetDB);
```

Sets *from_name* to the value of the *emp_name* field for the current user.

```
$from_email = CCDLookUp("email", "employees", "emp_id=".$DBIntranetDB->  
>ToSQL(CCGetUserID(), ccsInteger), $DBIntranetDB);
```

Sets *from_email* to the value of the *email* field in the *employees* table where *emp_id* matches the current user. The *CCDLookUp* function is used to retrieve a database value, while *CCGetUserID* retrieves the id of the currently logged in user.

```
$to_email = CCDLookUp("email", "employees", "emp_id=".$DBIntranetDB->  
>ToSQL($tasks->user_id_assign_to->GetValue(), ccsInteger),  
$DBIntranetDB);
```

Sets *to_email* to the email of the person that is assigned to the task. The *CCDLookUp* function is used here to retrieve the appropriate email address.

```
$headers = "From: ".$from_name."<".$from_email.">";
```

Add details about the sender to the email header

```
$headers .= "Content-Type: text/html";
```

Specifies that the email will be sent in HTML format (as opposed to plain text).

```
$subject = "New task for you";
```

The subject of the email to be sent.

```
$message = "The following task was submitted:<br><br>". "Task ID: ".  
CCDLookUp("max(task_id)", "tasks", "user_id_assign_by=".$DBIntranetDB->  
>ToSQL(CCGetUserID, ccsInteger), $DBIntranetDB)."<br><br>". $tasks->  
>task_desc->GetText();
```

The variable *message* contains the body of the email which will be sent. The *CCDLookUp* function is used to retrieve the largest task id submitted by the current user (assuming that task ids are created incrementally).

```
mail ($to_email, $subject, $message, $headers);
```

Sends the email using the variables created in the above code.

Next: [Use the After Update Event to Send Emails](#)

Use the After Update Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. We shall now implement similar functionality in **After Update** Event to notify the assignee when an existing task is updated and reassigned to someone.

1. Click on the *tasks* form in the **Project Explorer**.

2. In the **Properties** window select the **Events** tab.
3. Add the following **Custom Code** in **After Update** event:

```

4.     global $DBIntranetDB;
5.     global $tasks;
6.     if (CCGetUserID() != $tasks->user_id_assign_to-
>GetValue())
7.     {
8.         $from_name = CCDLookUp("emp_name", "employees",
"emp_id=");
9.         $DBIntranetDB->ToSQL(CCGetUserID(),
ccsInteger), $DBIntranetDB);
10.        $from_email = CCDLookUp("email", "employees",
"emp_id=");
11.        $DBIntranetDB->ToSQL(CCGetUserID(),
ccsInteger), $DBIntranetDB);
12.        $to_email = CCDLookUp("email", "employees", "emp_id=");
13.        $DBIntranetDB->ToSQL($tasks-
>user_id_assign_to->GetValue(), ccsInteger), $DBIntranetDB);
14.
15.        $headers = "From:
".$from_name."<".$from_email.">;\r\n";
16.        $headers .= "Content-Type: text/html;";
17.        $subject = "A task was assigned to you";
18.        $message = "The following task was assigned to
you:<br><br>".
19.                "Task ID: ".CCGetFromGet ("task_id", " "
)."<br><br>".
20.        $tasks->task_desc->GetText();
21.        mail($to_email, $subject, $message, $headers);
}

```

The main differences between the above code and that which was used in the **After Insert** event are as follows:

1. An *if* condition was added to send an email only if a user assigns a task to someone other than himself/herself.
2. *task_id* is retrieved from the URL using the `CCGetFromGet` function. We can use this method because tasks can be updated only if the user arrived at the current page via a URL that contains the task id to be updated. Such a URL would look like this:
http://localhost/TaskManager/tasks_maint.php?task_id=9

Next: [Test Email Delivery](#)

Test Email Delivery

Before testing the system:

1. You should add new users to your database with correct email addresses, or modify the existing users by changing their email address.

- a. You can do this by opening the Intranet.mdb database that is located in your project directory.
 - b. Alternatively, you may use the Task Manager itself. Go to the Employees page to view and modify user emails there.
2. Once you have users configured with test emails, save your project and switch to **Live Page** mode to test your system.

(Note: You will need MS Access 2000 or higher to manually edit the database file.) If your email code worked correctly, you should end up back at the Task List page after adding or modifying a task, and an email should be delivered to the person to whom the task was assigned.

Next: [Implement Record Security in After Initialize Event](#)

Step 5

Implement Record Security in After Initialize Event

Your Task Management system is now almost complete, except one possibly important feature- security.

Currently everyone can modify and delete any of the tasks. You may want to limit the access so that only the employee assigned to as task can update their tasks. There are many ways of accomplishing this, and we will examine several of them.

1. Click on the *tasks_maint* page in the **Project Explorer**.
2. Select **Events** tab in the **Properties** window.
3. Add **Custom Code** to the **After Initialize** event of the page as follows:
4. Once in the **Code** mode, replace the generated comment:

```
// Write your own code here.
```

with the code below:

```
global $tasks;
global $Redirect;
global $DBIntranetDB;
$current_task = CCGetParam("task_id", "");
if ($current_task != 0 && CCGetUserID() !=
CCDLookUp("user_id_assign_to", "tasks", "task_id=" .
$DBIntranetDB->ToSQL($current_task, ccsInteger),
$DBIntranetDB))
{
    $tasks->Visible = false;
    // $Redirect = "tasks_list.php";
    // $tasks->UpdateAllowed = false;
    // $tasks->DeleteAllowed = false;
}
```

The above code allows you to test the following methods of implementing record security:

Do not show the Task (record form) on the page if the selected task doesn't belong to the current user. An unauthorized user should see a blank page.

You can hide any form on a page by assigning a *False* value to the *Visible* property of the form. The code `$current_task != 0` in the *if* condition specifies that the code should be executed only if the user tries to modify an existing task and he/she is not assigned to it. The *if* also assures that all users can create new tasks. You can test this functionality by inserting the above code into the event, then switching to **Live Page** mode and trying to modify a task that is not assigned to you, in which case you should see an empty page. Although such functionality may not be very useful, it shows how you can hide forms on a page. You may consider adding another record form to your page that is not updateable and has just the Label fields that show information. Once you have two forms on the page, you can hide each form programmatically using mutually exclusive criteria.

Redirect unauthorized users to another page. Only users who are assigned to a task, can view the page.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
global $tasks;
global $Redirect;
global $DBIntranetDB;
$current_task = CCGgetParam("task_id", "");
if ($current_task != 0 && CCGetUserID() !=
CCDLookUp("user_id_assign_to", "tasks", "task_id=" .
$DBIntranetDB->ToSQL($current_task, ccsInteger), $DBIntranetDB))
{
    // $tasks->Visible = false;
    $Redirect = "tasks_list.php";
    // $tasks->UpdateAllowed = false;
    // $tasks->DeleteAllowed = false;
}
```

The above code shows that you should comment out the previously active line, and uncomment the line that starts with `$Redirect`. `$Redirect` is a variable used by CodeCharge Studio to determine if the current page should be redirected to another page, for example if a user is not logged in. This variable can be used only on pages that have restricted access and require users to login. You can simply assign the destination page to the `$Redirect` variable and the page will be automatically redirected. Test this functionality by modifying the code as shown then switch to **Live Page** mode and trying to modify a task that is not assigned to you.

Disallowed Update and Delete operations for unauthorized users. Only users who are assigned to a task, can edit (delete) it.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
global $tasks;
global $Redirect;
global $DBIntranetDB;
```



```

$current_task = CCGetParam("task_id", "");

if ($current_task != 0 && CCGetUserID() !=
CCDLookUp("user_id_assign_to", "tasks", "task_id=")

$DBIntranetDB->ToSQL($current_task, ccsInteger), $DBIntranetDB))
{
    // $tasks->Visible = false;
    // $Redirect = "tasks_list.php";
    $tasks->UpdateAllowed = false;
    $tasks->DeleteAllowed = false;
}

```

This code shows how you can manipulate the *UpdateAllowed* and *DeleteAllowed* properties of a record form. These properties control record update/delete operations execution. If set to *false*, the operation will not be executed. These properties also control the visibility of the **Update** and **Delete** buttons on the page.

Next: [Conclusion](#)

Perl

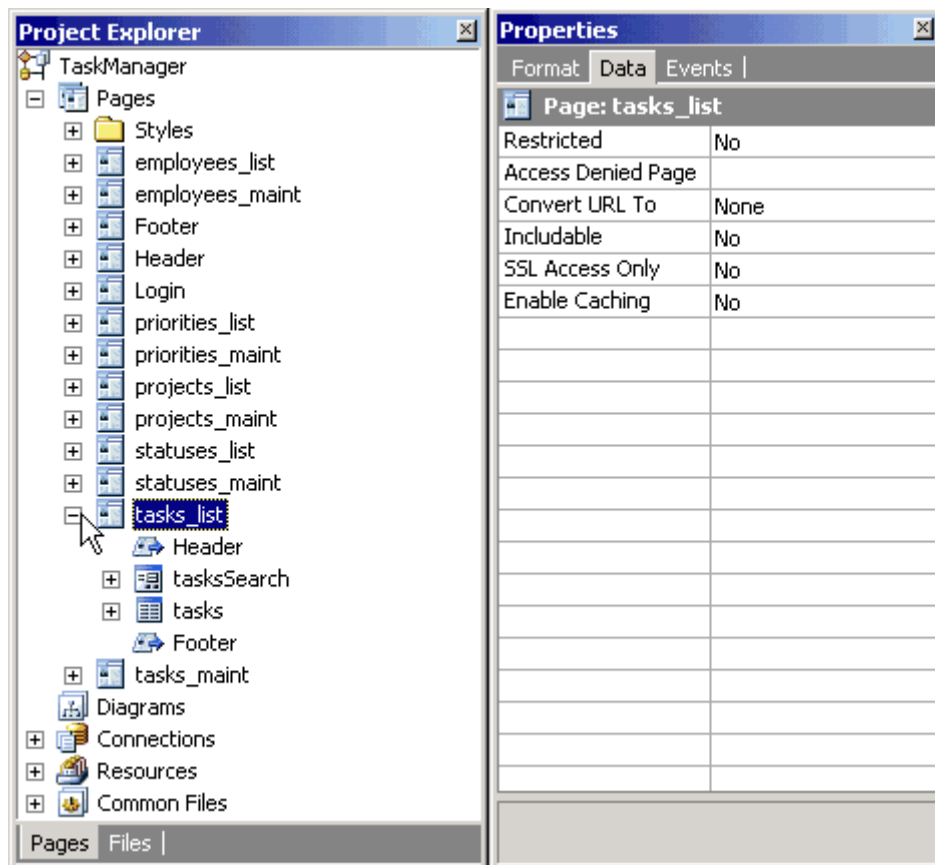
Step 1

Use the Before Show Event to Alter Text Color

Let's start our basic programming with a simple task of altering the color of a grid field on our Task List page. To be more specific, we will mark the listed tasks assigned to you by showing your name in blue color within the grid.

1. Open the *tasks_list* page in the **Project Explorer**.
2. Expand the *tasks* grid.
3. In the **Project Explorer** right-click on the *task_name* field and select **Properties**.
4. Under the **Data** tab, set the value of the **Content** property to *HTML*.
5. Select *tasks* grid in the **Project Explorer**, or clicking anywhere within the form's caption.
6. Select the **Events** tab in the **Properties** window.
7. Right-click on the **Before Show Row** event and select **Add Code...**

The **Before Show Row** event occurs in the program after the field values are assigned, but before being output as HTML. By adding code into this event, you can modify the field value before it is shown.



Next: [Programmatically Control Field's Value](#)

Programmatically Control Field's Value

Once you add Custom Code to the Event, you will see the code-editing window with the appropriate place to enter the new code.

1. Replace this line of code:

```
# Write your own code here.
```

with the following lines (Perl):

```
if ($tasks->{user_id_assign_to}->GetValue() eq
CCGetSession("UserLogin"))
{
    $tasks->{task_name}->SetValue("<b><font color='blue'>" .
        $tasks->{task_name}->GetValue()."</font></b>");
}
}
```

The following is how the above Perl code works:

```
if ($tasks->{user_id_assign_to}->GetValue() eq CCGetSession("UserLogin"))
```

This is an *if* condition that is true only if the value of the *task_name* field is equal to the login name of the employee that is currently logged into the system. The database field provides the value for the *task_name* field in the grid form. Once you login to the system, the program will recognize your tasks by comparing your login name to the emp_login value of the person that a task is assigned to. *UserLogin* is one of the session variables used by CodeCharge-generated programs, and it holds the Login name of the currently logged in user until the session expires.

Note:

The following are all default session variables created by CodeCharge Studio:

- *UserID* - the primary key field value of the logged in user
- *UserLogin* - login name of the user currently logged into the system
- *GroupID* - security level/group of the user currently logged into the system

```
$tasks->{task_name}->SetValue("<b><font color='blue'>" . $tasks->{task_name}->GetValue() . "</font></b>");
```

This code is executed if the previous *if* condition is met. It modifies the value of the *task_name* field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML ** tag to make the font bold as well. Additionally, notice that the code is object-oriented and you specify that you want to assign a value to the *task_name* field in the *tasks* grid. *SetValue* is a method of an object, which can be used to modify the object's value.

Next: [Preview Tasks List Page](#)

Preview Tasks List Page

1. Save your project.
2. Go to **Live Page** mode to view your working page. If the column "Name" doesn't have any task names highlighted, you are probably not logged in.
3. Since the menu doesn't contain a link to the Login page at this time, you can access it by trying to access one of the restricted pages, such as Task Maintenance.
4. Click on any of the project Ids and you should see the **Login** page.
5. Login as *george/george*, then click on the **Tasks** link on the menu to get back to the Task List page.

Now you should see some task names highlighted, which are the tasks of the user that you logged in as.

Search Tasks

Keyword

Project Select Value ▼

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

[Add New](#) 1 of 1

Next: [Modify a Label Field on the Task Maintenance Page](#)

Step 2

Modify a Label Field on the Task Maintenance Page

Now let's make one necessary modification on the Task Maintenance page where you might've noticed that the Label field *Assigned By* doesn't display the employee name, but the ID, as shown below. This is because the *tasks* table contains only the user ID, while the *employees* table contains the actual user names.

There are several potential methods of dealing with the above issue as explained below:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, just like you did with the grid on the Task List page. Unfortunately, queries that contain multiple tables may not be updateable by their nature, and thus your whole record form may stop working. In other words, if you specified that you want to use a query containing *tasks* and *employees* table in your record form, then if you assigned a task to someone else, the program wouldn't know if you wanted to update the *tasks* table with the new *employee_id*, or if you wanted to update the *employees* table and change employee's name.

Thus if you used multiple tables as a data source for the record form, you would also need to specify Custom Insert, Custom Update and Custom Delete operations in record form's properties, to specify which database fields should be updated with corresponding values entered on the page. This approach looks like too much effort just for displaying one additional value on the page.

2. Use an Event Procedure to insert custom code where you can programmatically output the desired value. This method is very flexible, as it allows you to extend the generated code by adding your own. The next step describes this method in detail.

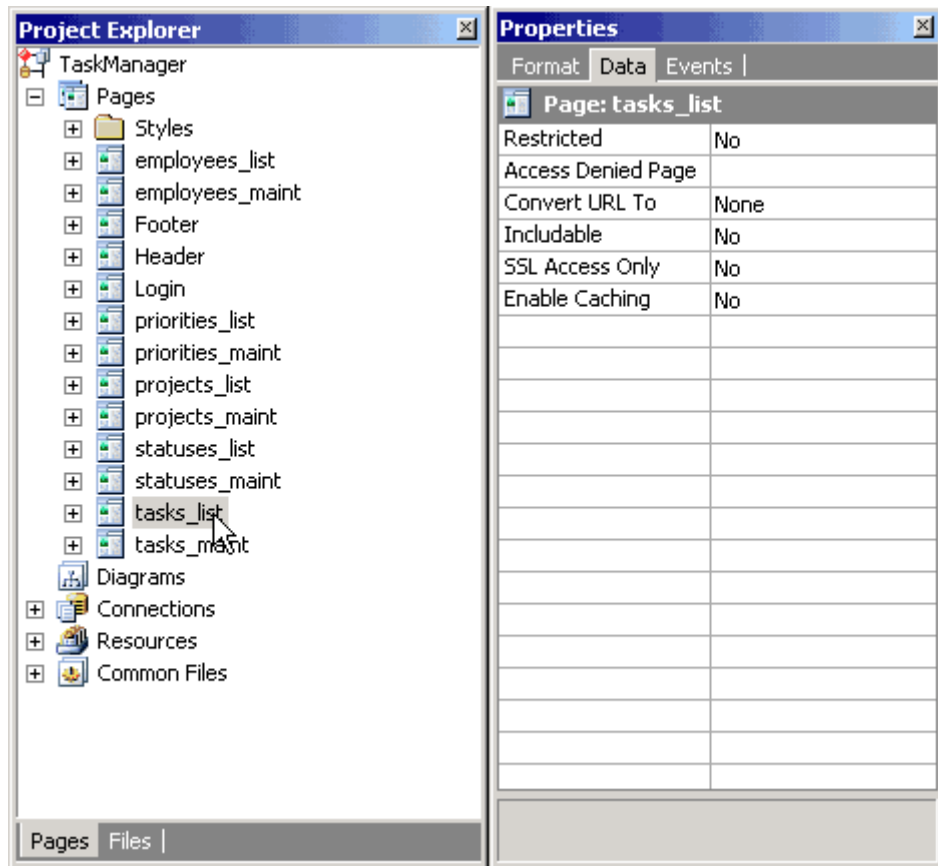
[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

Next: [Use the Before Show Event to Alter a Label's Value](#)

Use the Before Show Event to Alter a Label's Value

1. Select the Label *user_id_assign_by* in the **Project Explorer**.
2. Click on the **Data** tab in the **Properties** window.
3. Select *Text* as the **Data Type**.
4. In the **Properties** window click on **Events** tab.
5. Right-click on **Before Show** event and select **Add Code....** .
CodeCharge Studio should then automatically switch to **Code** view.



6. Once in Code view, if you generate Perl, you should see the *tasks_maint_events.pl* file, with the following lines of code:

```
7.      # -----
8.      # Write your own code here.
```

```
# -----
```

Replace the text:

```
# Write your own code here.
```

with the following:

```
$DBIntranetDB = clsDBIntranetDB->new();
if ( $tasks->{EditMode} )
{
    $tasks->{user_id_assign_by}->SetValue(CCDlookUp( "emp_name" ,
"employees", " emp_id=" .
        $DBIntranetDB->ToSQL($tasks->{user_id_assign_by}-
>GetValue(), $ccsInteger),$DBIntranetDB));
}
else
{
    $tasks->{user_id_assign_by}->SetValue(CCDlookUp( "emp_name" ,
"employees", " emp_id=" .
        $DBIntranetDB->ToSQL(CCGetSession( "UserID" ) ,
$ccsInteger), $DBIntranetDB));
}
```

The above code consists of the following elements:

- *tasks* -the name of the record form on the page
- *EditMode* - property of the form, which specifies if the record is being edited. Depending on the value of this property, we either display the name of the person who originally submitted the task (Edit mode), or the person who is currently submitting the task (Insert mode).
- *user_id_assign_by* - the name of the Label within the Grid, and at the same time the name of the database field that was used to create this Label and which is now its data source.
- *SetValue* - a method of an object (in this case the Label), which can be used to modify the object's value.
- *\$tasks->{user_id_assign_by}->SetValue* - fully qualified *SetValue* method, which tells the program which object it refers to. In other words, it is the *SetValue* method that affects *user_id_assign_by* field, which in turn belongs to the *tasks* grid.
- *CCDlookup* - CodeCharge function that supports retrieving a database value based on a field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table using the condition that the key (*emp_id*) equals the current value of the Label.
- *ToSQL* - CodeCharge function that converts a value into the format supported by the database. This function requires a parameter that tells it if a value should be converted to a number (Integer) or text. In this case, this function converts the current Label value to a number that can be used with the *CCDlookup* function. It is advisable to always use this function together with *CCDlookup*.
- *\$DBIntranetDB* - the name of the object that defines the database connection that you want to use in *CCDlookup* function.
- *CCGetSession("UserID")* - CodeCharge function that returns the ID of the user that is currently logged in.

The whole code reads approximately as follows:

- If a record is being edited: Assign the name of the person who originally submitted the issue to the *user_id_assign_by* Label, by looking up employee's name from *employees* table using *CCDlookup* function that uses the *IntranetDB* connection and the value of the *user_id_assign_by* Label.
- If a new record is being created:
Assign current user to the *user_id_assign_by* Label by retrieving his/her name from *employees* table using *CCDlookup* function that uses *IntranetDB* connection and *CCGetSession("UserID")* function that obtains current user's ID.

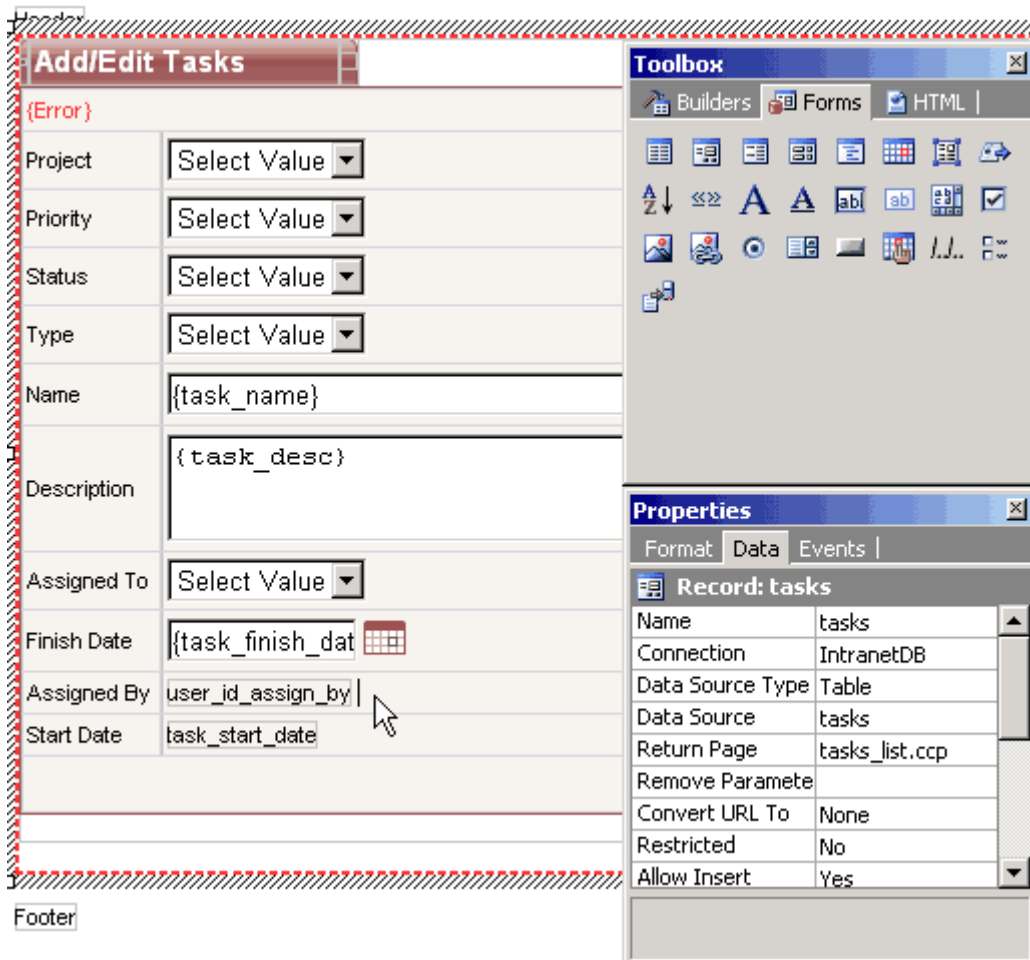
Next: [Add a Hidden "Assigned By" Field to Auto-Update New Tasks](#)

Step 3

Add a Hidden "Assigned By" Field to Auto-Update New Tasks

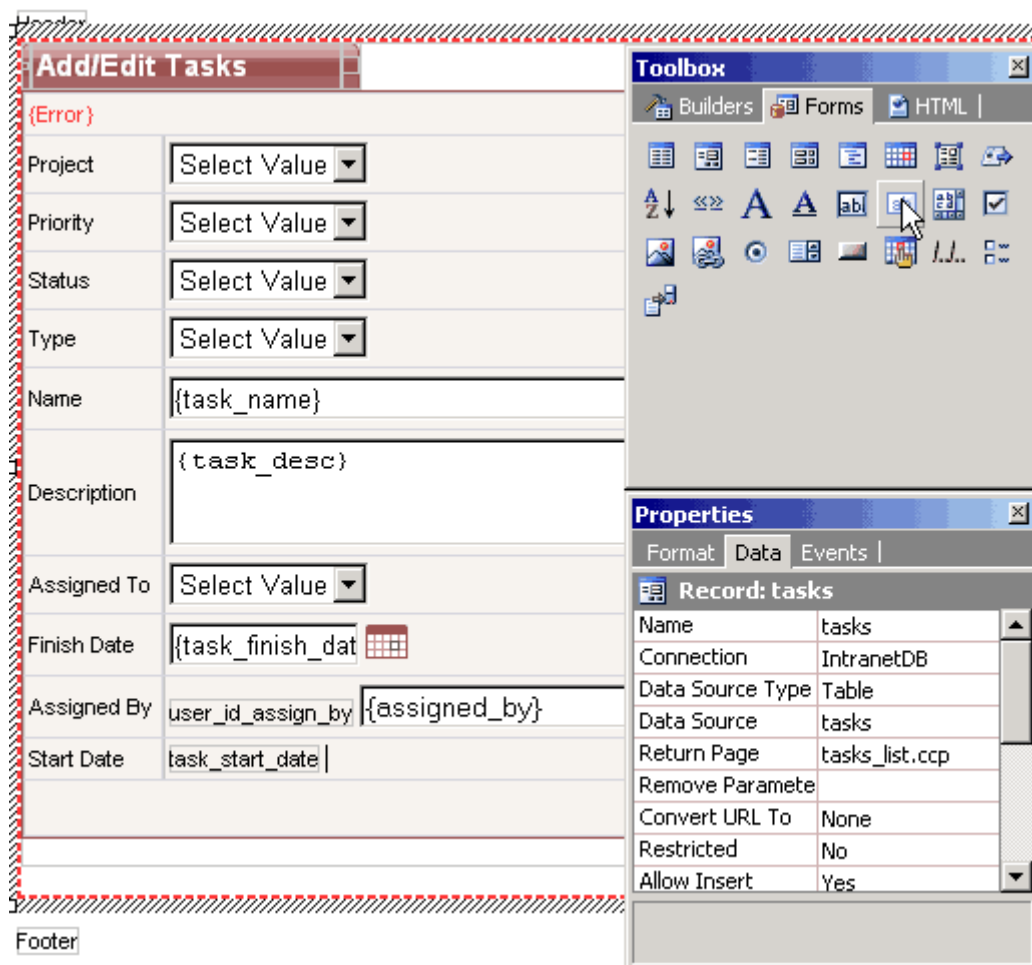
You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though the employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.

1. Add a Hidden field to your page from the **Forms** tab of the **Toolbox**.
This field type isn't visible in the browser, but will be used to store a value and update the database.
2. Configure the new field by setting its properties as follows:
 - **Name:** *assigned_by* - the name of the newly added Hidden field. This can be any name you choose.
 - **Control Source:** *user_id_assign_by* - the database field/column that will be used to retrieve field's value and will be updated with the new value, if it changes.
 - **Data Type:** *Integer* - the type of the value bound to the control source. Our user/employee ID's are numeric.
 - **Default:** *CCGetSession("UserID")* - *CCGetSession("UserID")* is a CodeCharge function that retrieves the ID of the user that is currently logged into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.



Next: [Add a Hidden "Date Created" Field to the Record Form](#)

1. Configure the new field as follows:
 - **Name:** *date_created*
 - **Control Source:** *task_start_date*
 - **Data Type:** *Date*
 - **Default:** *CurrentDateTime* - CurrentDateTime is a predefined default function that obtains the server's current date and time. Using this function allows you to automatically assign the current date and time to new tasks. The Default property doesn't affect existing records, thus the date of existing tasks won't be modified during updates.
2. Click on the *task_start_date* field.
3. In the **Properties** window, set its Default value to *CurrentDateTime*. This is so that the label field can display the date since the hidden field will not be visible to the user.




Next: [Test the Label and Hidden Fields](#)

Test the Label and Hidden Fields

Finally,

1. Switch to **Live Page** mode.
2. Select or add a Task and see your Label display the name of the person who assigned the task.
The basic version of your Task Manager is now completed.
3. Don't forget to save it!

Add/Edit Tasks	
Project	My Project
Priority	Highest
Status	In progress
Type	Task
Name	Finish my project
Description	Implement "Assigned By" Label to show values from another table
Assigned To	alexander
Finish Date	<input type="text"/> 
Assigned By	George Pennington
Start Date	13.09.2005 8:07:17
<input type="button" value="Add"/>	

Next: [Programming the Record Form](#)

Step 4

Programming the Record Form

Now you've created a simple task management application, but how do you extend it to be more practical and useful? In this section, you will get a glimpse of how to implement practical and sophisticated applications by adding programming code and actions that enhance the application's functionality.

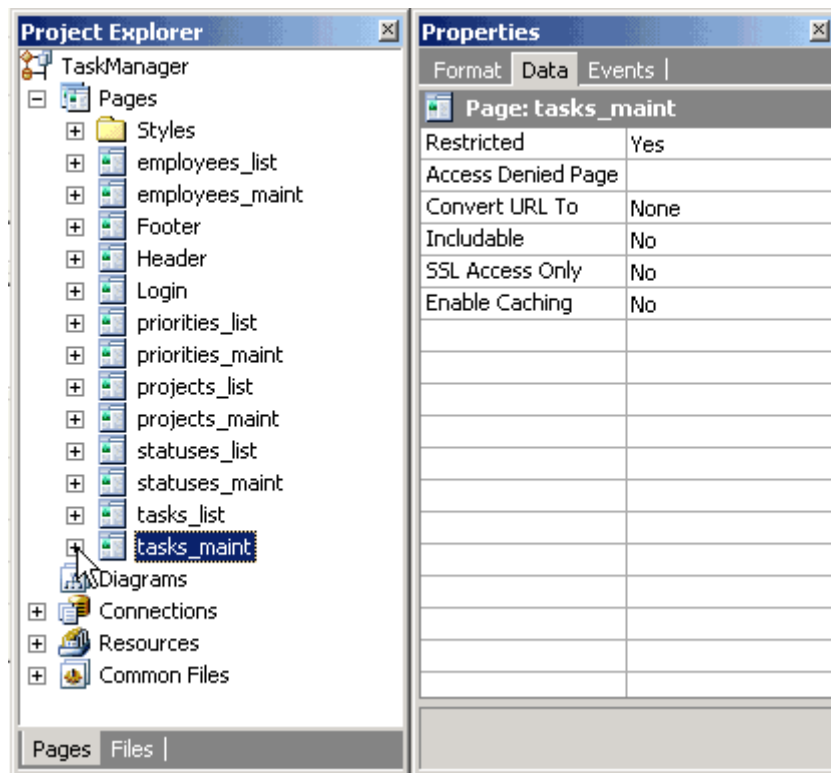
You will learn how to:

- Send email notifications to the person that the task is being assigned to.
- Allow only the person assigned to the task to modify it.

Next: [Add Code in the After Insert Event to Send Emails](#)

Add Code in the After Insert Event to Send Emails

1. Select the *tasks* form by selecting it in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window click on the **Events** tab.
3. Select the **After Insert** event.
4. Click on the **[+]** button.
5. Select **Add Code...**



6. Once you are in the **Code** mode, replace the generated comment:

```
// Write your own code here.
```

with the code below:

```
$DBIntranetDB = clsDBIntranetDB->new();

$from_name = CCDlookUp("emp_name", "employees", "emp_id=" .
$DBIntranetDB->ToSQL(CCGetSession("UserID"), $ccsInteger),
$DBIntranetDB);
$from_email = CCDlookUp("email", "employees", "emp_id=" .
$DBIntranetDB->ToSQL(CCGetSession("UserID"), $ccsInteger),
$DBIntranetDB);
$to_email= CCDlookUp("email", "employees", "emp_id=" .
$DBIntranetDB->ToSQL($tasks->{user_id_assign_to}-
>GetValue(), $ccsInteger), $DBIntranetDB);

$subject = "New task for you";
$message = "The following task was submitted:<br><br>" .
"Task ID: " . CCDlookUp("max(task_id)", "tasks",
"user_id_assign_by=" .
$DBIntranetDB->ToSQL(CCGetSession("UserID"),
$ccsInteger), $DBIntranetDB) . "<br><br>" .

$tasks->{task_desc}->GetValue();

if(open(SENDMAIL, "| sendmail $from_name"))
{
    print SENDMAIL "From: $from_email\nTo: $to_email\nSubject:
$subject\n\n$message\n.\n";
    close(SENDMAIL);
}
```

```
}
```

As you may have realized by now, the above code sends emails to users to whom the new tasks are assigned. The following is an explanation of the code.

```
$from_name = CCDlookUp("emp_name", "employees", "emp_id=" .  
$DBIntranetDB->ToSQL(CCGetSession("UserID"), $ccsInteger),  
$DBIntranetDB);
```

Sets *from_name* to the value of the *emp_name* field for the current user.

```
$from_email = CCDlookUp("email", "employees", "emp_id=" .  
$DBIntranetDB->ToSQL(CCGetSession("UserID"), $ccsInteger),  
$DBIntranetDB);
```

Sets *from_email* to the value of the *email* field in the *employees* table where *emp_id* matches the current user. The CCDlookUp function is used to retrieve a database value, while CCGetSession("UserID") retrieves the id of the currently logged in user.

```
$to_email= CCDlookUp("email", "employees", "emp_id=" . $DBIntranetDB->  
>ToSQL($tasks->{user_id_assign_to}->GetValue(), $ccsInteger),  
$DBIntranetDB);
```

Sets *to_email* to the email of the person that is assigned to the task. The CCDlookUp function is used here to retrieve the appropriate email address.

```
$subject = "New task for you";
```

The subject of the email to be sent.

```
$message = "The following task was submitted:<br><br>" . "Task ID: " .  
CCDlookUp("max(task_id)", "tasks", "user_id_assign_by=" .  
$DBIntranetDB->ToSQL(CCGetSession("UserID"), $ccsInteger),  
$DBIntranetDB) . "<br><br>" . $tasks->{task_desc}->GetValue();
```

The variable *message* contains the body of the email which will be sent. The CCDlookUp function is used to retrieve the largest task id submitted by the current user (assuming that task ids are created incrementally).

```
if(open(SENDMAIL, "| sendmail $from_name"))  
{  
    print SENDMAIL "From: $from_email\nTo: $to_email\nSubject:  
$subject\n\n$message\n.\n";  
    close(SENDMAIL);  
}
```

Sends the email using the variables created in the above code.

Next: [Use the After Update Event to Send Emails](#)

Use the After Update Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. We shall now implement similar functionality in **After Update** Event to notify the assignee when an existing task is updated and reassigned to someone. <

1. Click on the *tasks* form in the **Project Explorer**.

2. In the **Properties** window select the **Events** tab.
3. Add the following **Custom Code** in **After Update** event:

```

4.         $DBIntranetDB = clsDBIntranetDB->new();
5.
6.         if (CCGetSession("UserID") != $tasks-
>{user_id_assign_to}->GetValue())
7.         {
8.             $from_name = CCDlookUp("emp_name", "employees",
"emp_id=" .
9.                 $DBIntranetDB-
>ToSQL(CCGetSession("UserID"), $ccsInteger), $DBIntranetDB);
10.            $from_email = CCDlookUp("email", "employees",
"emp_id=" .
11.                $DBIntranetDB-
>ToSQL(CCGetSession("UserID"), $ccsInteger), $DBIntranetDB);
12.            $to_email= CCDlookUp("email", "employees", "emp_id=" .
13.                $DBIntranetDB->ToSQL($tasks-
>{user_id_assign_to}->GetValue(), $ccsInteger),
$DBIntranetDB);
14.            $subject = "A task was assigned to you";
15.            $message = "The following task was assigned to
you:<br><br>" .
16.                "Task ID: " .CCGetFromGet("task_id","")
. "<br><br>" .
17.                $tasks->{task_desc}->GetValue();
18.
19.            if (open(SENDMAIL, "| sendmail $from_name"))
20.            {
21.                print SENDMAIL "From: $from_email\nTo:
$to_email\nSubject: $subject\n\n$message\n.\n";
22.                close(SENDMAIL);
23.            }
}

```

The main differences between the above code and that which was used in the **After Insert** event are as follows:

1. An *if* condition was added to send an email only if a user assigns a task to someone other than himself/herself
2. *task_id* is retrieved from the URL using the `CCGetFromGet` function. We can use this method because tasks can be updated only if the user arrived at the current page via a URL that contains the task id to be updated. Such a URL would look like this:
http://localhost/TaskManager/tasks_maint.cgi?task_id=9

Next: [Test Email Delivery](#)

Test Email Delivery

Before testing the system:

1. You should add new users to your database with correct email addresses, or modify the existing users by changing their email address.
 - a. You can do this by opening the Intranet.mdb database that is located in your project directory.
 - b. Alternatively, you may use the Task Manager itself. Go to the Employees page to view and modify user emails there.
2. Once you have users configured with test emails, save your project and switch to **Live Page** mode to test your system.

Note: You will need MS Access 2000 or higher to manually edit the database file. If your email code worked correctly, you should end up back at the Task List page after adding or modifying a task, and an email should be delivered to the person to whom the task was assigned.

Next: [Implement Record Security in After Initialize Event](#)

Step 5

Implement Record Security in After Initialize Event

Your Task Management system is now almost complete, except one possibly important feature- security.

Currently everyone can modify and delete any of the tasks. You may want to limit the access so that only the employee assigned to as task can update their tasks. There are many ways of accomplishing this, and we will examine several of them.

1. Click on the *tasks_maint* page in the **Project Explorer**.
2. Select **Events** tab in the **Properties** window.
3. Add **Custom Code** to the **After Initialize** event of the page as follows:
4. Once in the **Code** mode, replace the generated comment:

```
# Write your own code here.
```

with the code below:

```
$DBIntranetDB = clsDBIntranetDB->new();
$current_task = CCGetParam("task_id", "");
$task_user_id = CCDlookUp("user_id_assign_to", "tasks",
"task_id=" .
                $DBIntranetDB->ToSQL($current_task,
$ccsInteger), $DBIntranetDB);

if ( ($current_task != 0) && (CCGetSession("UserID") !=
$task_user_id) )
{
    $tasks->{Visible} = 0;
    # $Redirect = "tasks_list.cgi";
    # $tasks->{UpdateAllowed} = 0;
    # $tasks->{DeleteAllowed} = 0;
}
```

The above code allows you to test the following methods of implementing record security:

Do not show the Task (record form) on the page if the selected task doesn't belong to the current user. An unauthorized user should see a blank page.

You can hide any form on a page by assigning a 0 value to the *Visible* property of the form.

The code `$current_task != 0` in the *if* condition specifies that the code should be executed only if the user tries to modify an existing task and he/she is not assigned to it. The *if* also assures that all users can create new tasks. You can test this functionality by inserting the above code into the event, then switching to **Live Page** mode and trying to modify a task that is not assigned to you, in which case you should see an empty page. Although such functionality may not be very useful, it shows how you can hide forms on a page. You may consider adding another record form to your page that is not updateable and has just the Label fields that show information. Once you have two forms on the page, you can hide each form programmatically using mutually exclusive criteria.

Redirect unauthorized users to another page. Only users who are assigned to a task, can view the page.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
$DBIntranetDB = clsDBIntranetDB->new();
$current_task = CCGgetParam("task_id", "");
$task_user_id = CCDlookup("user_id_assign_to", "tasks", "task_id=" .
    $DBIntranetDB->ToSQL($current_task, $ccsInteger),
    $DBIntranetDB);

if ( ($current_task != 0) && (CCgetSession("UserID") != $task_user_id)
)
{
    # $tasks->{Visible} = 0;
    $Redirect = "tasks_list.cgi";
    # $tasks->{UpdateAllowed} = 0;
    # $tasks->{DeleteAllowed} = 0;
}
```

The above code shows that you should comment out the previously active line, and uncomment the line that starts with `$Redirect`. `$Redirect` is a variable used by CodeCharge Studio to determine if the current page should be redirected to another page, for example if a user is not logged in. This variable can be used only on pages that have restricted access and require users to login. You can simply assign the destination page to the `$Redirect` variable and the page will be automatically redirected. Test this functionality by modifying the code as shown then switch to **Live Page** mode and trying to modify a task that is not assigned to you.

Disallowed Update and Delete operations for unauthorized users. Only users who are assigned to a task, can edit (delete) it.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
$DBIntranetDB = clsDBIntranetDB->new();
$current_task = CCGetParam("task_id", "");
$task_user_id = CCDlookup("user_id_assign_to", "tasks", "task_id=" .
                        $DBIntranetDB->ToSQL($current_task, $ccsInteger),
                        $DBIntranetDB);

if ( ($current_task != 0) && (CCGetSession("UserID") != $task_user_id)
)
{
    # $tasks->{Visible} = 0;
    # $Redirect = "tasks_list.cgi";
    $tasks->{UpdateAllowed} = 0;
    $tasks->{DeleteAllowed} = 0;
}
```

This code shows how you can manipulate the *UpdateAllowed* and *DeleteAllowed* properties of a record form. These properties control record update/delete operations execution. If set to *false*, the operation will not be executed. These properties also control the visibility of the **Update** and **Delete** buttons on the page.

Next: [Conclusion](#)

CFML

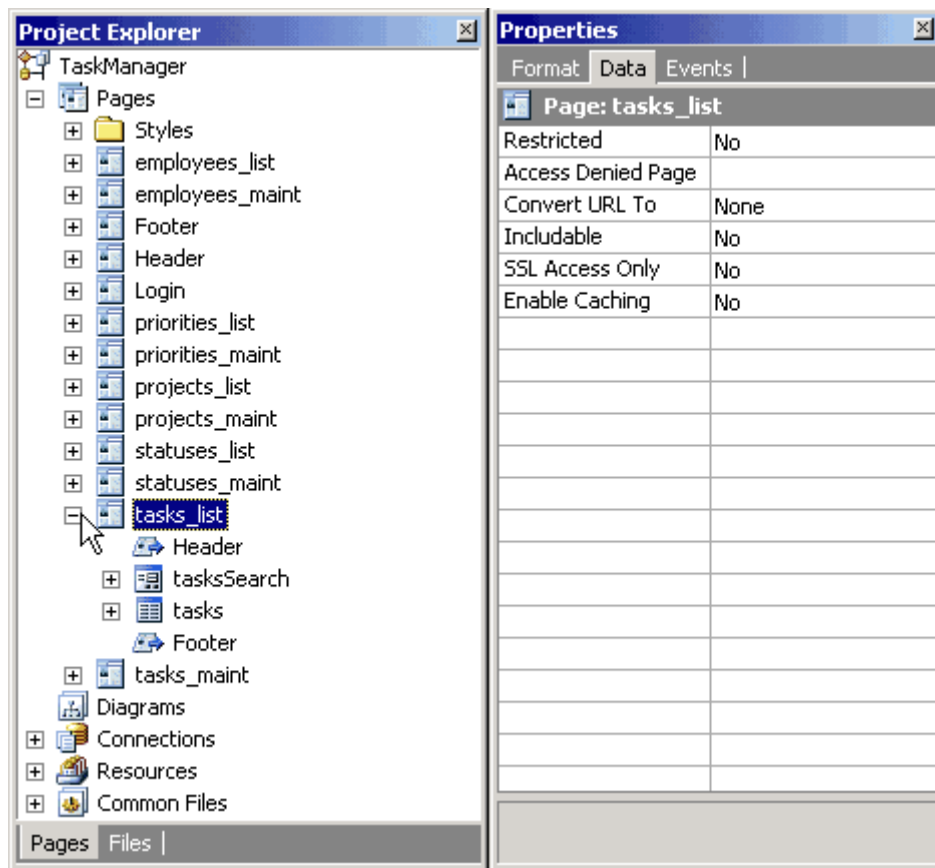
Step 1

Use the Before Show Event to Alter Text Color

Let's start our basic programming with a simple task of altering the color of a grid field on our Task List page. To be more specific, we will mark the listed tasks assigned to you by showing your name in blue color within the grid.

1. Open the *tasks_list* page in the **Project Explorer**.
2. Expand the *tasks* grid.
3. In the **Project Explorer** right-click on the *task_name* field and select **Properties**.
4. Under the **Data** tab, set the value of the **Content** property to *HTML*.
5. Select *tasks* grid in the **Project Explorer**, or clicking anywhere within the form's caption.
6. Select the **Events** tab in the **Properties** window.
7. Right-click on the **Before Show Row** event and select **Add Code...**

The **Before Show Row** event occurs in the program after the field values are assigned, but before being output as HTML. By adding code into this event, you can modify the field value before it is shown.



Next: [Programmatically Control Field's Value](#)

Programmatically Control Field's Value

Once you add Custom Code to the Event, you will see the code-editing window with the appropriate place to enter the new code.

1. Replace this line of code:

```
<!-- write your own code here -->
```

with the following lines (ColdFusion):

```
<CFPARAM Name="Session.UserLogin" Default="">
<CFIF flduser_id_assign_to EQ Session.UserLogin AND
Session.UserLogin NEQ "">
  <CFSET fldtask_name= '<b><font color="blue">' & fldtask_name
& '</font></b>'>
</CFIF>
```

The following is how the above code works:

```
<CFPARAM Name="Session.UserLogin" Default="">
```

This code checks the existence of the UserLogin session variable. If the variable does not exist, it is created and its values set to an empty string.

```
<CFIF fldtask_name EQ Session.UserLogin AND Session.UserLogin NEQ "">
```

This is a conditional statement that is true only if the value of the *fldtask_name* field is equal to the login name of the employee that is currently logged into the system. The *employees1_emp_login* database field provides the value for the *fldtask_name* field in

the grid form. Once you login to the system, the program will recognize your tasks by comparing your login name to the `emp_login` value of the person that a task is assigned to. `UserLogin` is one of the session variables used by CodeCharge-generated programs, and it holds the Login name of the currently logged in user until the session expires.

Note:

The following are all default session variables created by CodeCharge Studio:

- `UserID` - the primary key field value of the logged in user
- `UserLogin` - login name of the user currently logged into the system
- `GroupID` - security level/group of the user currently logged into the system

```
<CFSET fldtask_name= '<b><font color="blue">' & fldtask_name & '</font></b>''>
```

This code is executed if the previous *if* condition is met. It modifies the value of the `task_name` field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML `` tag to make the font bold as well.

```
</CFIF>
```

This line marks the end of the *if* condition, so that the execution of the remaining code is not affected by this condition.

Next: [Preview Tasks List Page](#)

Preview Tasks List Page

1. Save your project.
2. Go to **Live Page** mode to view your working page. If the column "Name" doesn't have any task names highlighted, you are probably not logged in.
3. Since the menu doesn't contain a link to the Login page at this time, you can access it by trying to access one of the restricted pages, such as Task Maintenance.
4. Click on any of the project Ids and you should see the **Login** page.
5. Login as *george/george*, then click on the **Tasks** link on the menu to get back to the Task List page.

Now you should see some task names highlighted, which are the tasks of the user that you logged in as.

Search Tasks

Keyword

Project Select Value ▼

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

[Add New](#) 1 of 1

Next: [Modify a Label Field on the Task Maintenance Page](#)

Step 2

Modify a Label Field on the Task Maintenance Page

Now let's make one necessary modification on the Task Maintenance page where you might've noticed that the Label field *Assigned By* doesn't display the employee name, but the ID, as shown below. This is because the *tasks* table contains only the user ID, while the *employees* table contains the actual user names.

There are several potential methods of dealing with the above issue as explained below:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, just like you did with the grid on the Task List page. Unfortunately, queries that contain multiple tables may not be updateable by their nature, and thus your whole record form may stop working. In other words, if you specified that you want to use a query containing *tasks* and *employees* table in your record form, then if you assigned a task to someone else, the program wouldn't know if you wanted to update the *tasks* table with the new *employee_id*, or if you wanted to update the *employees* table and change employee's name.

Thus if you used multiple tables as a data source for the record form, you would also need to specify Custom Insert, Custom Update and Custom Delete operations in record form's properties, to specify which database fields should be updated with corresponding values entered on the page. This approach looks like too much effort just for displaying one additional value on the page.

2. Use an Event Procedure to insert custom code where you can programmatically output the desired value. This method is very flexible, as it allows you to extend the generated code by adding your own. The next step describes this method in detail.

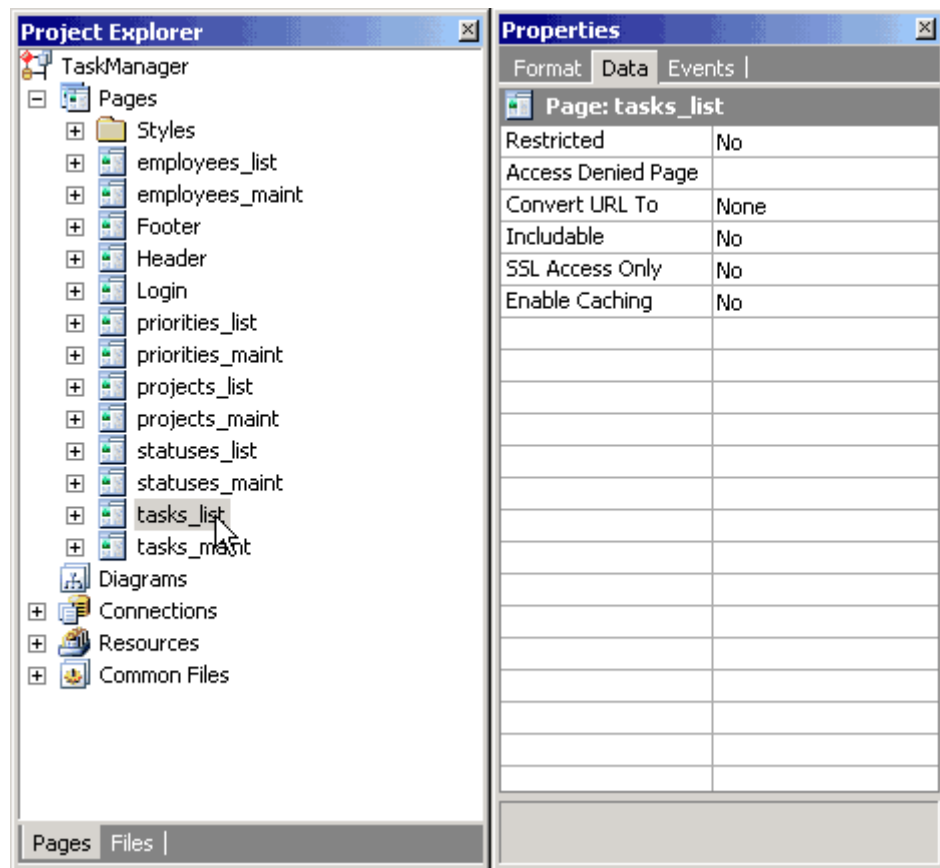
[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

Next: [Use the Before Show Event to Alter a Label's Value](#)

Use the Before Show Event to Alter a Label's Value

1. Select the Label *user_id_assign_by* in the **Project Explorer**.
2. In the **Properties** window click on the **Data** tab.
3. Select *Text* for the **Data Type** property.
4. In the **Properties** window click on the **Events** tab.
5. Right-click on **Before Show** event and select **Add Code....** .
CodeCharge Studio should then automatically switch to **Code** view.



6. Once in Code view, replace the following text:

```
<!-- write your own code here --->
```

with the following:

```
<CFIF blnEditModeTasks>
  <CF_CCToSQL Value= "#flduser_id_assign_by#" Type=
"#ccsInteger#" >
  <CF_CCDLookUp Field="emp_name" Table="employees"
Where="emp_id=#CCToSQL#" Connection="IntranetDB">
<CFSET flduser_id_assign_by=CCDLookUp>
<CFELSE>
  <CF_CCToSQL Value="#Session.UserID#" Type="#ccsInteger#" >
  <CF_CCDLookUp Field="emp_name" Table="employees"
Where="emp_id=#CCToSQL#" Connection="IntranetDB">
<CFSET flduser_id_assign_by=CCDLookUp>
</CFIF>
```

The above code consists of the following elements:

- *blnEditModeTasks* - form variable, which specifies if the record is being edited. Depending on the value of this property, the program displays either the name of the person who originally submitted the task (Edit mode), or the person who is currently submitting the task (Insert mode).
- *user_id_assign_by* - the name of the Label within the Grid, and at the same time the name of the database field that was used to create this Label and which is now its data source.

- *flduser_id_assign_by* - the name of the variable used by the generated program to refer to the Label and to the corresponding database. CodeCharge Studio constructs variable names by adding the "fld" prefix to field's name.
- *CCDLookup* -CodeCharge custom tag that supports retrieving a database value based on field name, table name, and a condition. Here, this function retrieves the Employee Name *emp_name* from the *employees* table using the condition that the key *emp_id* equals the current value of the Label.
- *CCToSQL* - CodeCharge custom tag that converts a value into the format supported by the database. This function requires a parameter that tells it if a value should be converted to a number (Integer) or text. In this case, this function converts the current Label value to a number that can be used with *CCDLookup* function. It is advisable to always use this function together with *CCDLookup*.
- *IntranetDB* - the name of the database connection that you want to use in the *CCDLookup* function.
- *Session.UserID* - the session variable that contains the ID of the user that is currently logged in.

The whole code reads approximately as follows:

- If a record is being edited:
Assign the name of the person who originally submitted the issue to the *user_id_assign_by* Label, by looking up employee's name from *employees* table using *CCDLookup* custom tag that uses the *IntranetDB* connection and the value of the *user_id_assign_by* Label.
- If a new record is being created:
Assign current user to the *user_id_assign_by* Label by retrieving his/her name from *employees* table using *CCDLookup* custom tag that uses *IntranetDB* connection and *CCGetUserID* custom tag that obtains current user's ID.

Next: [Add a Hidden "Assigned By" Field to Auto-Update New Tasks](#)

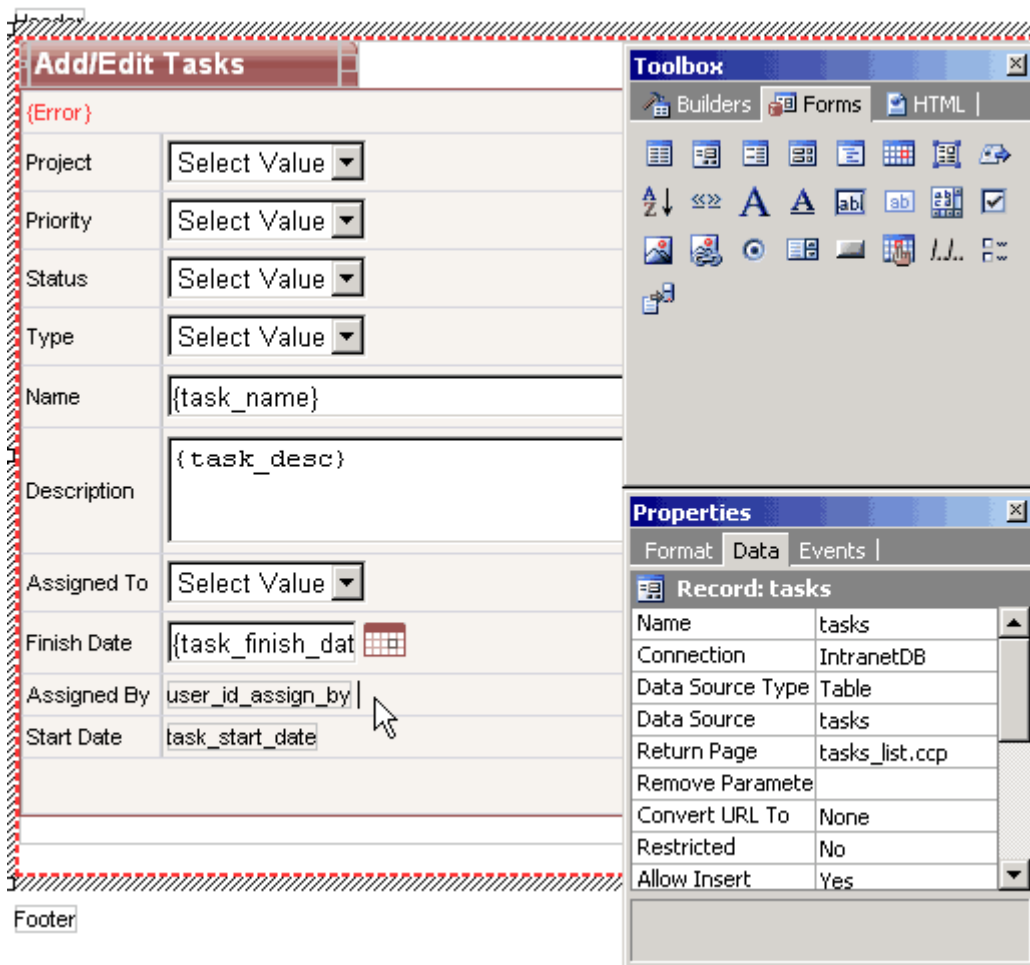
Step 3

Add a Hidden "Assigned By" Field to Auto-Update New Tasks

You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though the employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.

1. Add a Hidden field to your page from the **Forms** tab of the **Toolbox**.
This field type isn't visible in the browser, but will be used to store a value and update the database.
2. Configure the new field by setting its properties as follows:
 - **Name:** *assigned_by* - the name of the newly added Hidden field. This can be any name you choose.

- **Control Source:** *user_id_assign_by* - the database field/column that will be used to retrieve field's value and will be updated with the new value, if it changes.
- **Data Type:** *Integer* - the type of the value bound to the control source. Our user/employee ID's are numeric.
- **Default:** *Session.UserID* - *Session.UserID* is a CodeCharge created session variable that holds the ID of the user that is currently logged into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.



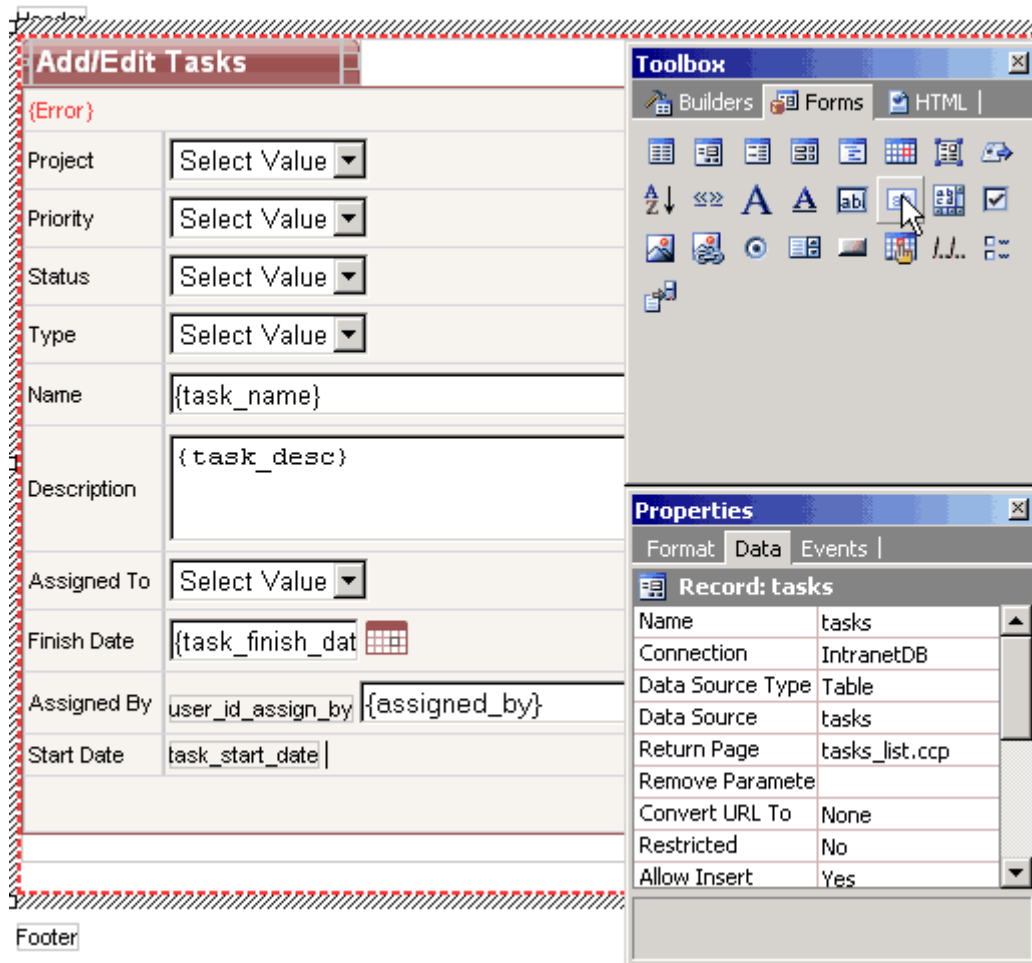
Next: [Add a Hidden "Date Created" Field to the Record Form](#)

Add a Hidden "Date Created" Field to the Record Form

Now add another Hidden field to your page, which will be used to submit the current date and time to the *date_assign* field in the database.

1. Configure the new field as follows:
 - **Name:** *date_created*
 - **Control Source:** *task_start_date*
 - **Data Type:** *Date*
 - **Default:** *CurrentDateTime* - The *CurrentDateTime* value allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.

2. Click on the *task_start_date* field.
3. In the **Properties** window, set its Default value to *CurrentDateTime*. This is so that the Label field can display the date since the hidden field will not be visible to the user.



Next: [Test the Label and Hidden Fields](#)


Test the Label and Hidden Fields

Finally,

1. Switch to **Live Page** mode.
2. Select or add a Task and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.

3. Don't forget to save it!

Add/Edit Tasks	
Project	My Project
Priority	Highest
Status	In progress
Type	Task
Name	Finish my project
Description	Implement "Assigned By" Label to show values from another table
Assigned To	alexander
Finish Date	<input type="text"/> 
Assigned By	George Pennington
Start Date	13.09.2005 8:07:17
<input type="button" value="Add"/>	

Next: [Programming the Record Form](#)

Step 4

Programming the Record Form

Now you've created a simple task management application, but how do you extend it to be more practical and useful? In this section, you will get a glimpse of how to implement practical and sophisticated applications by adding programming code and actions that enhance the application's functionality.

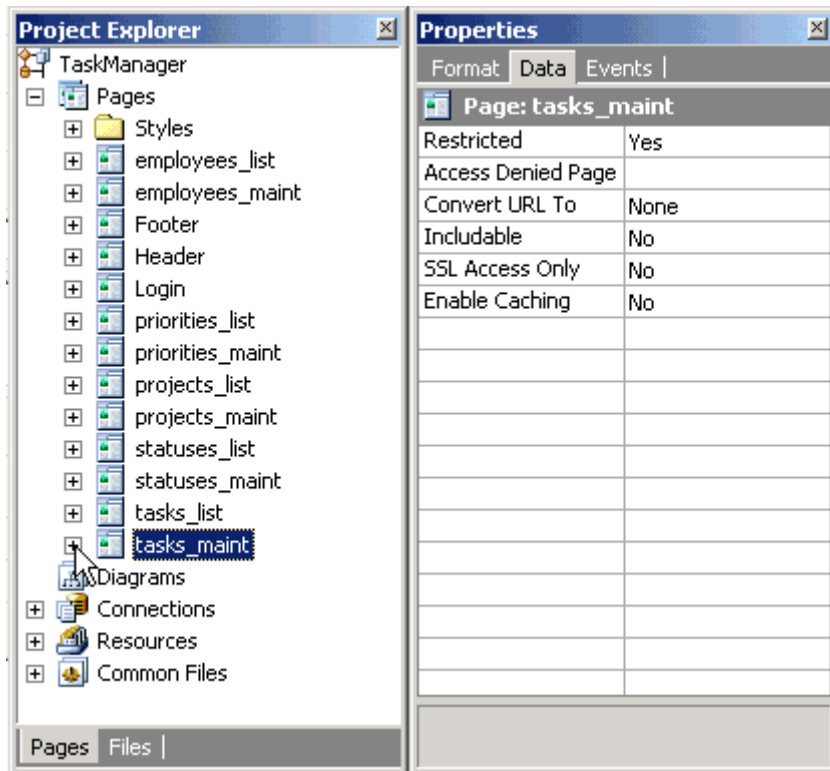
You will learn how to:

- Send email notifications to the person that the task is being assigned to
- Allow only the person assigned to the task to modify it

Next: [Add Code in the After Insert Event to Send Emails](#)

Add Code in the After Insert Event to Send Emails

1. Select the *tasks* form by selecting it in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window click on the **Events** tab.
3. Select the **After Insert** event.
4. Click on the **[+]** button, then select **Add Code...**



5. Once you are in the Code view, replace the generated comment:

```
<!-- write your own code here -->
```

with the code below:

```
<CF_CCToSQL Value="#flduser_id_assign_to#"
Type="#ccsInteger#">
<CF_CCDLookUp Field="email" Table="employees"
Where="emp_id=#CCToSQL#" Connection="IntranetDB">
<CFSET mail_To=CCDLookUp>
<CF_CCToSQL Value="#Session.UserID#" Type="#ccsInteger#">
<CF_CCDLookUp Field="email" Table="employees"
Where="emp_id=#CCToSQL#" Connection="IntranetDB">
<CFSET mail_From=CCDLookUp>
<CF_CCDLookUp Field="max(task_id)" Table="tasks"
Where="user_id_assign_by=#CCToSQL#" Connection="IntranetDB">
<CFMAIL TO="#mail_To#" FROM="#mail_From#" SUBJECT="New task
for you" TYPE="HTML" >
    The following task was submitted:<br><br>
    Task ID:#CCDLookUp#
    <br><br>#fldtask_desc#
</CFMAIL>
```

The following is the explanation of the above code:

```
<CF_CCToSQL Value="#flduser_id_assign_to#" Type="#ccsInteger#">
<CF_CCDLookUp Field="email" Table="employees" Where="emp_id=#CCToSQL#"
Connection="IntranetDB">
<CFSET mail_To= CCDLookUp>
```

Sets the *To* email address to the email of the person that is assigned to the task. The CCDLookUp custom tag is used here to retrieve the appropriate

email address.

```
<CF_CCToSQL Value="#Session.UserID#" Type="#ccsInteger#">
<CF_CCDLookUp Field="email" Table="employees" Where=
"emp_id=#CCToSQL#" Connection="quot;IntranetDB">
<CFSET mail_From= CCDLookUp>
```

Sets the *From* email address to the value of the *email* field in the *employees* table where *emp_id* matches the current user. The *CCDLookUp* function is used to retrieve a database value, while *Session.UserID* contains the id of the currently logged in user.

```
<CF_CCDLookUp Field="max(task_id)" Table="tasks"
Where="user_id_assign_by=#CCToSQL#" Connection="IntranetDB">
```

Retrieves the current Task Id. The last inserted task id can be obtained using different methods with different databases. Unfortunately, MS Access doesn't support the retrieval of the last inserted record, therefore you will need to use the *CCDLookUp* function to retrieve the largest task id submitted by the current user (assuming that task ids are created incrementally).

```
<CFMAIL TO="#mail_To#" FROM="#mail_From#" SUBJECT="New task for you"
TYPE="HTML" >
  The following task was submitted:<br><br>
  Task ID:#CCDLookUp#
  <br><br>#fldtask_desc#
</CFMAIL>
```

The *CFMail* tag sends out an email address using the variables set within the tag. The body of the tag makes up the body of the email address that will be sent.

Next: [Use the "After Update" Event to Send Emails](#)

Use the "After Update" Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. We shall now implement similar functionality in **After Update** Event to notify the assignee when an existing task is updated and reassigned to someone.

1. Click on the *tasks* form in the **Project Explorer**.
2. In the **Properties** window, select the **Events** tab.
3. Add the following **Custom Code** in **After Update** event:

```
4.         <CFIF flduser_id_assign_to NEQ Session.UserID>
5.             <CF_CCToSQL Value="#flduser_id_assign_to#"
              Type="#ccsInteger#">
6.             <CF_CCDLookUp Field="email" Table="employees"
              Where="emp_id=#CCToSQL#" Connection="IntranetDB">
7.             <CFSET mail_To=CCDLookUp>
8.             <CF_CCToSQL Value="#Session.UserID#"
              Type="#ccsInteger#">
9.             <CF_CCDLookUp Field="email" Table="employees"
              Where="emp_id=#CCToSQL#" Connection="IntranetDB">
```

```

10.      <CFSET mail_From=CCDLookUp>
11.      <CF_CCGetParam strName="task_id" def="">
12.      <CFMAIL TO="#mail_To#" FROM="#mail_From#" SUBJECT="New
task for you" TYPE="HTML">
13.          The following task was assigned to you:<br><br>
14.          Task ID:#CCGetParam#
15.          <br><br>#fldtask_desc#
16.      </CFMAIL>

```

```
</CFIF>
```

The main differences between the above code and that which was used in the **After Insert** event are as follows:

1. An *if* condition was added to send an email only if a user assigns a task to someone other than himself/herself
2. *task_id* is retrieved from the URL using the `CCGetParam` custom tag. We can use this method because tasks can be updated only if the user arrived at the current page via a URL that contains the task id to be updated. Such a URL would look like this:
http://localhost/TaskManager/tasks_maint.cfm?task_id=9

Next: [Test Email Delivery](#)

Test Email Delivery

Before testing the system:

1. You should add new users to your database with correct email addresses, or modify the existing users by changing their email address.
 - a. You can do this by opening the `Intranet.mdb` database that is located in your project directory.
 - b. Alternatively, you may use the Task Manager itself. Go to the Employees page to view and modify user emails there.
2. Once you have users configured with test emails, save your project and switch to **Live Page** mode to test your system.

Note: You will need MS Access 2000 or higher to manually edit the database file. If your email code worked correctly, you should end up back at the Task List page after adding or modifying a task, and an email should be delivered to the person to whom the task was assigned.

Next: [Implement Record Security in After Initialize Event](#)

Step 5

Implement Record Security in After Initialize Event

Your Task Management system is now almost complete, except one possibly important feature- security.

Currently everyone can modify and delete any of the tasks. You may want to limit the

access so that only the employee assigned to as task can update their tasks. There are many ways of accomplishing this, and we will examine several of them.

1. Click on the *tasks_maint* page in the **Project Explorer**.
2. Select **Events** tab in the **Properties** window.
3. Add **Custom Code** to the **After Initialize** event of the page as follows. Once in the **Code** mode, replace the generated comment:

```
4.      <!------->
5.      <!-- Write your own code here -->
```

```
<!------->
```

with the code below:

```
<CF_CCGetParam strName="task_id" outputvar="current_task">
<CFPARAM Name="Session.UserID" Default="">
<CF_CCToSQL Value="#current_task#" Type="#ccsInteger#">
<CF_CCDLookUp Field="user_id_assign_to" Table="tasks"
Where="task_id=#CCToSQL#">
<CFIF current_task NEQ 0 AND Session.UserID NEQ CCDLookUp>
  <CFSET hideTasks = True>
  <!--<CFSET strRedirect = "tasks_list.cfm"> -->
  <!--<CFSET blnUpdateAllowedTasks = False> -->
  <!--<CFSET blnDeleteAllowedTasks = False> -->
</CFIF>
```

The above code allows you to test the following methods of implementing record security:

Do not show the Task (record form) on the page if the selected task doesn't belong to the current user. An unauthorized user should see a blank page.

You can hide any form on a page by assigning *True* value to the *hideTasks* variable. The code "*current_task NEQ 0*" in the *CFIF* condition specifies that the code should be executed only if a user tries to modify an existing task and he/she is not assigned to it. *CFIF* also assures that all users can create new tasks. You can test this functionality by inserting the above code into the event, then switching to Live Page mode and trying to modify a task that is not assigned to you, in which case you should see an empty page. Although such functionality may not be very useful, it shows how you can hide forms on a page. You may consider adding another record form to your page that is not updateable and has just the Label fields that show information. Once you have two forms on the page, you can hide each form programmatically using mutually exclusive criteria.

Redirect unauthorized users to another page. Only users who are assigned to a task can view the page.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
<CF_CCGetParam strName="task_id" outputvar="current_task">
<CFPARAM Name="Session.UserID" Default="">
```

```
<CF_CCToSQL Value="#current_task#" Type="#ccsInteger#">
```

```
<CF_CCDLookUp Field="user_id_assign_to" Table="tasks"  
Where="task_id=#CCToSQL#">
```

```
<CFIF current_task NEQ 0 AND Session.UserID NEQ CCDLookUp>  
  <!--<CFSET hideTasks = True> --->  
  <CFSET strRedirect = "tasks_list.cfm">  
  <!--<CFSET blnUpdateAllowedTasks = False> --->  
  <!--<CFSET blnDeleteAllowedTasks = False> --->  
</CFIF>
```

The above code shows that you should comment out the previously active line, and uncomment the line that starts with *<CFSET strRedirect*. *strRedirect* is a variable used by CodeCharge Studio to determine if the user should be redirected to another page, for example if a user is not logged in. This variable can be used only on pages that have restricted access and require users to login. You can simply assign the destination page to the *strRedirect* variable and the user will be automatically redirected. Test this functionality by modifying the code as shown, then switch to **Live Page** mode and trying to modify a task that is not assigned to you.

Disallowed Update and Delete operations for unauthorized users. Only users who are assigned to a task can edit (delete) it.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
<CF_CCGetParam strName="task_id" outputvar="current_task">  
<CFPARAM Name="Session.UserID" Default="">  
<CF_CCToSQL Value="#current_task#" Type="#ccsInteger#">  
<CF_CCDLookUp Field="user_id_assign_to" Table="tasks"  
Where="task_id=#CCToSQL#">  
<CFIF current_task NEQ 0 AND Session.UserID NEQ CCDLookUp>  
  <!--<CFSET hideTasks = True> --->  
  <!--<CFSET strRedirect = "tasks_list.cfm"> --->  
  <CFSET blnUpdateAllowedTasks = False>  
  <CFSET blnDeleteAllowedTasks = False>  
</CFIF>
```

This code shows how you can manipulate the *blnUpdateAllowed* and *blnDeleteAllowed* variables of a record form. These properties control record update/delete operations execution. If set to *false*, the operation will not be executed. These properties control also the visibility of the **Update** and **Delete** buttons on the page.

Next: [Conclusion](#)

JSP

Step 1

Use the Before Show Row Event to Alter Text Color


```
String user_id_assign_to =
e.getGrid().getControl("user_id_assign_to").getFormattedValue(
);

String login = Utils.getUserLogin(e.getPage());

if ( user_id_assign_to != null &&
user_id_assign_to.equals(login))
{
String task_name =
e.getGrid().getControl("task_name").getFormattedValue();
task_name = "<b><font color=\"blue\">" + task_name +
"</font></b>";

e.getGrid().getControl("task_name").setFormattedValue(task_name);
}
}
```

Let's explain how the above JSP code works:

```
String user_id_assign_to =
e.getGrid().getControl("user_id_assign_to").getFormattedValue();
String login = Utils.getUserLogin(e.getPage());
if ( user_id_assign_to != null && user_id_assign_to.equals(login)) {
```

The above lines prepare and then execute the "if" condition that checks if the value of the current control (*user_id_assign_to*) is equal to the user login name. Once you login to the system, the program will recognize your tasks by comparing your login name to the *emp_login* value of the person that a task is assigned to. `Utils.getUserLogin(Page)` method is one of the static auxiliary methods in `Utils` class, and it holds the Login name of the currently logged in user until the session expires.

Note:

The following are different helpful functions from the `Utils` class:

- `getUserId(Page)` - the primary key field value of the logged in user
- `getUserLogin(Page)` - login name of the user currently logged into the system
- `getUserGroup(Page)` - security level/group of the user currently logged into the system

Page parameter is needed to get access to Session object, where these values are stored actually.

```
e.getGrid().getControl("task_name").setFormattedValue(task_name);
```

This code is executed if the previous "if" condition is met. It modifies the value of the current control (*user_id_assign_to*). The field value is replaced with the user login name value wrapped within HTML code that specifies the font color as blue, and adds HTML `` tag to make the font bold as well. Notice that the word *blue* has escaped quotes around it, which will be replaced with a single quote at runtime. Since quotes mark the start and end of a string, using escaped quote allows you to insert a quote into a string.

Next: [Preview the Tasks List Page](#)

Preview the Tasks List Page

1. Save your project.
2. Go to **Live Page** mode to view your working page. If the last column ("Assigned To") doesn't have any names highlighted, you are probably not logged in.
3. Since the menu doesn't contain a link to the Login page at this time, you can see it by trying to access one of the restricted pages, such as Task Maintenance.
4. Click on any of the project Ids and you should see the **Login** page.
5. Login as *george/george*.
6. Click on the *Tasks* link on the menu to get back to the Task List page.

Now you should see one of the names highlighted, which is the name of the user that you logged in as.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Search Tasks

Keyword

Project Select Value ▼

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

[Add New](#) 1 of 1

Next: [Modify a Label Field on the Task Maintenance Page](#)

Step 2

Modify a Label Field on the Task Maintenance Page

Now let's make one necessary modification on the Task Maintenance page where you might have noticed that the Label field "Assigned By" doesn't display the employee name, but the user id, as shown below. This is because *tasks* table contains only the user id, while *employees* table contains the user's name, just like "Assigned To" ListBox displays employee names from another table.

There are several potential methods of dealing with the above issue and let's explain each one in detail:

1. Create a Query that contains multiple tables and can be used as the data source for the record form.
Unfortunately, queries that contain multiple tables may not be updateable by their nature, and thus your whole record form may stop working. In other words, if you specified that you want to use a query containing *tasks* and *employees* table in your record form, then if you assigned a task to someone else, the program wouldn't know if you wanted to update the *tasks* table with the new *employee_id*, or if you wanted to update the *employees* table and change employee's name.

Thus if you used multiple tables as a data source for the record form, you would also need to specify Custom Insert, Custom Update and Custom Delete operations in record form's properties, to specify which database fields should be updated with corresponding values entered on the page.

This approach looks like too much effort just for displaying one additional value on the page.

2. Use an Event Procedure to insert custom code where you can programmatically output the desired value. This method is very flexible, as it allows you to extend the generated code by adding your own. The next step describes this method in detail.

[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

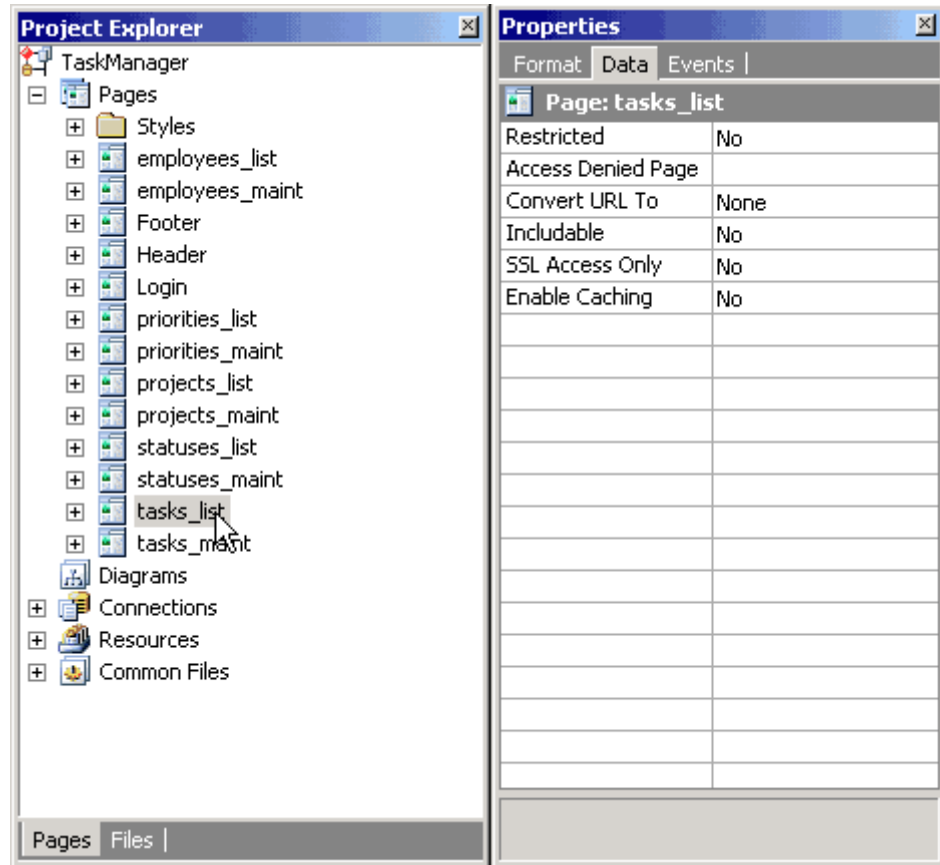
Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

Next: [Use the Before Show Event to Alter a Label's Value](#)

Use the Before Show Event to Alter a Label's Value

1. Select the Label *user_id_assign_by* in the **Project Explorer**.
2. Click on **Data** tab in **Properties** window.
3. Select *Text* as the **Data Type**.
4. In the **Properties** window click on the **Events** tab.
5. Right-click on **Before Show** event and select **Add Code....** .

CodeCharge Studio should then automatically switch to **Code** view.



CodeCharge Studio should then automatically switch to **Code** view.

6. Once in **Code** view, you should see the following lines of code:

```
7.         //user_id_assign_by Label Handler Head @13-91DFA71C-->
8.         public class tasksuser_id_assign_byLabelHandler
           implements LabelListener
9.         {
10.            public void beforeShow(CCSEvent e)
11.            {
12.                //End user_id_assign_by Label Handler Head-->
13.
14.                //Event BeforeShow Action Custom Code @13-4EFC2132-->
15.                /* Write your own code here. */
16.                //End Event BeforeShow Action Custom Code-->
17.
18.                //user_id_assign_by Label Handler Tail @13-F5FC18C5-->
19.            }
20.        }
```

```
//End user_id_assign_by Label Handler Tail-->
```

Replace the text:

```
/* Write your own code here. */
```

with the following:

```

if (e.getRecord().isEditMode())
{
    e.getControl().setValue(DBTools.dLookup("emp_name",
"employees", "emp_id=" +

DBTools.toSql(e.getControl().getFormattedValue(),JDBCConnectio
n.INTEGER,"IntranetDB"), "IntranetDB"));
}
else
{
    e.getControl().setValue(DBTools.dLookup("emp_name","employees"
, "emp_id="+

DBTools.toSql(Utils.getUserId(e.getPage()),JDBCConnection.INTE
GER,"IntranetDB") , "IntranetDB"));
}
}

```

In the above code, the first "if" block checks if the current record form is in the update mode and if so, it retrieves the name of the appropriate employee and assigns it to the current control's value. If the form is not in the update mode, the name of the currently logged in user is assigned as the control's value.

The following elements make up the code:

- *e* - the pointer to the Label *user_id_assign_by*, for which the event is called.
- *getFormattedValue* and *setFormattedValue* - the methods that read and modify the value of current control (in this case the Label). These methods optionally use a parameter that specifies if a value should be converted to a number (Integer) or text (String).
- *isEditMode* - property of the form (control's parent), which specifies if the record is being edited/updated. Depending on the value of this property, we either display the name of the person who originally submitted the task (Update mode), or the person who is currently submitting the task (Insert mode).
- *dLookup* - CodeCharge function that supports retrieving database value based on a field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table using the condition that the key (*emp_id*) equals the current value of the Label.
- *IntranetDB* - the name of the pooled connection object that defines the database connection used to retrieve employee's name.
- *Utils.getUserId(e.getPage())* - a method that retrieves the value of the "UserID" session, which contains the ID of the user that is currently logged in.

The whole code snippet reads approximately as follows:

- If record is being edited:
Assign the name of the person who originally submitted the issue to the *user_id_assign_by* Label, by looking up employee's name from *employees*

table using *IntranetDB* connection and the value of the *user_id_assign_by* Label.

- If new record is being created:
Assign current user to the *user_id_assign_by* Label by retrieving his/her name from *employees* table using *IntranetDB* connection and the "UserID" session that contains current user's ID.

Next: [Add a Hidden "Assigned By" Field to Auto-Update New Tasks](#)

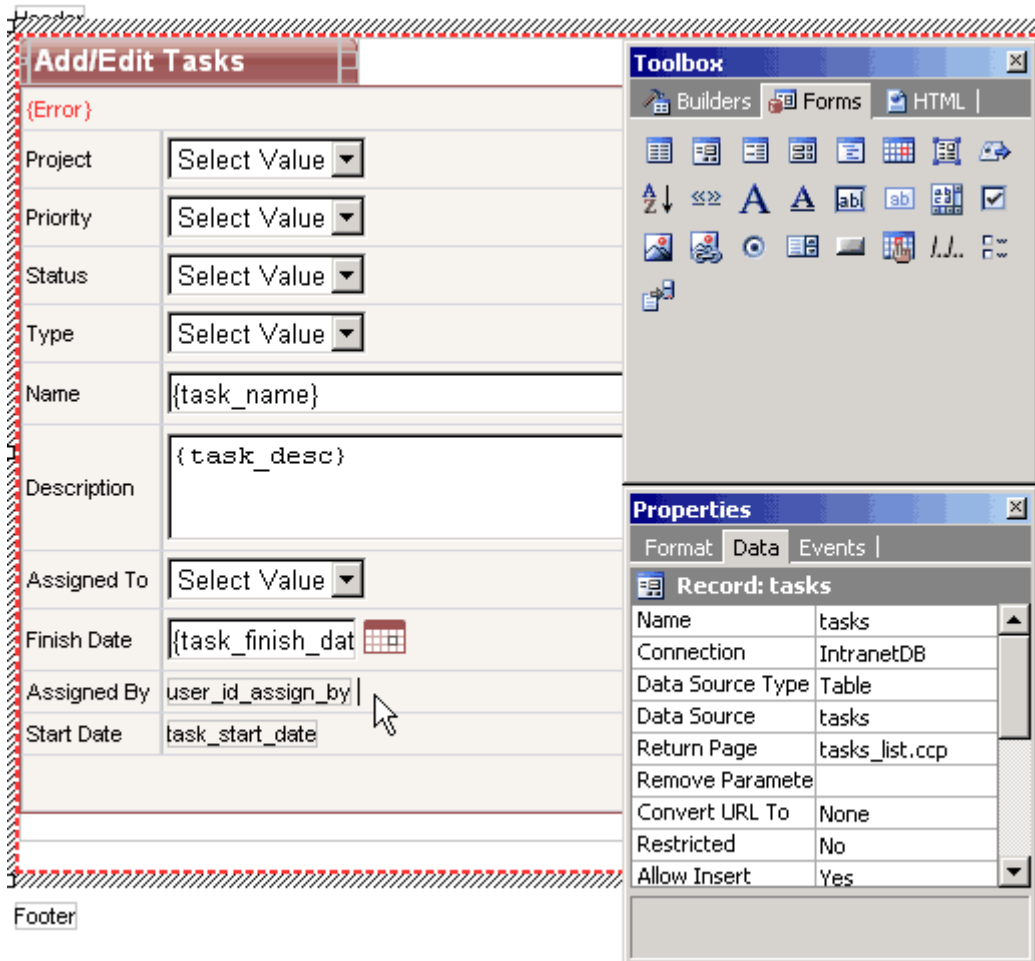
Step 3

Add a Hidden "Assigned By" Field to Auto-Update New Tasks

You've previously used the **Before Show** event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though employee's name is displayed on the page; it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.

1. Add a **Hidden** field to your page from the **Forms** tab of the **Toolbox** window.
This field type isn't visible in the browser, but will be used to store values and update the database.
2. Configure the new field by setting its properties as follows:
 - **Name:** *assigned_by* - the name of the newly added Hidden field. This can be any name you choose.
 - **Control Source:** *user_id_assign_by* - the database field/column that will be used to retrieve field's value and will be updated with the new value, if it changes.
 - **Data Type:** *Integer* - the type of the value bound to the control source. Our user/employee id's are numeric.
 - **Default:** *Utils.getUserId(e.getPage())* - default value for this field if empty.

This code retrieves the user id of the user that is currently logged into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.

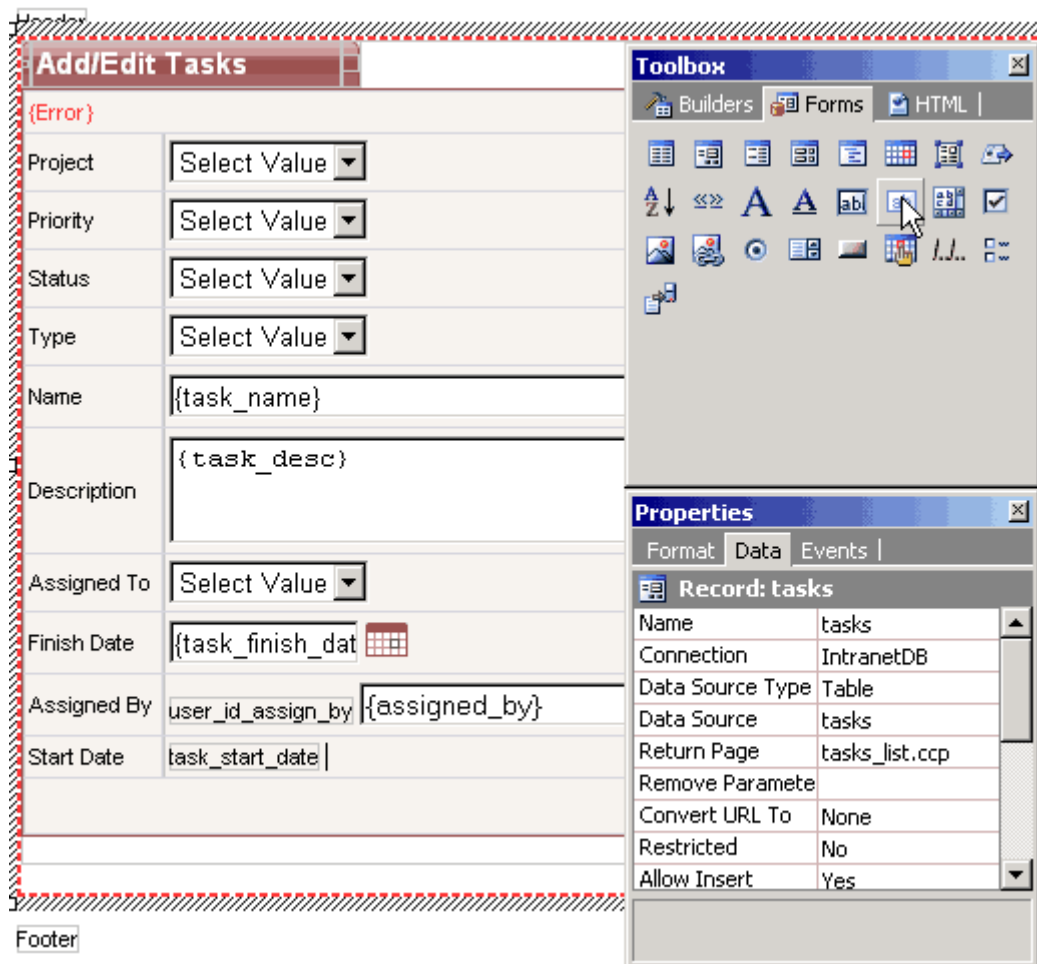


Next: [Add Hidden "Date Created" Field to the Record Form](#)

Add Hidden "Date Created" Field to the Record Form

Now add another "Hidden" field to your page, which will be used to submit the current date and time to the *date_assign* field in the database.

1. Configure the new field as follows:
 - o **Name:** *date_created*
 - o **Control Source:** *task_start_date*
 - o **Data Type:** *Date*
 - o **Default:** *CurrentDateTime* - The "CurrentDateTime" property allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.
2. Click on the *task_start_date* field.
3. In the **Properties** window, set its Default value to *CurrentDateTime*. This is so that the Label field can display the date since the hidden field will not be visible to the user.




Next: [Test the Label and Hidden Fields](#)

Test the Label and Hidden Fields

Finally,

1. you can switch to Live Page mode,
2. select a Task, Login,
3. and see your Label display the name of the person who assigned the task.
The basic version of your Task Manager is now completed.
4. Don't forget to save it!

Add/Edit Tasks	
Project	My Project
Priority	Highest
Status	In progress
Type	Task
Name	Finish my project
Description	Implement "Assigned By" Label to show values from another table
Assigned To	alexander
Finish Date	<input type="text"/> 
Assigned By	George Pennington
Start Date	13.09.2005 8:07:17
<input type="button" value="Add"/>	

Next: [Programming the Record Form](#)

Step 4

Programming the Record Form

Now you've created a simple task management application, but how do you extend it to be more practical and useful? In this section, you will get a glimpse of how to implement practical and sophisticated applications by adding programming code and actions that enhance the application's functionality.

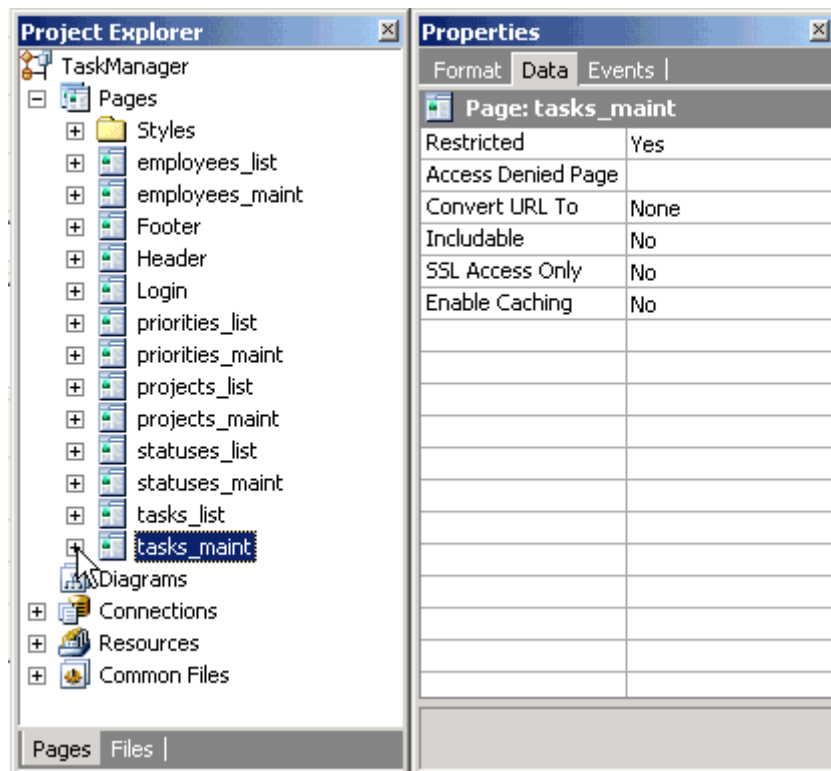
You will learn how to:

- Send email notifications to the person that the task is being assigned to
- Allow only the person assigned to the task to modify it

Next: [Add Code in the After Insert Event to Send Emails](#)

Add Code in the After Insert Event to Send Emails

1. Select the *tasks* form by selecting it in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window click on the **Events** tab.
3. Select the **After Insert** event.
4. Click on the **[+]** button.
5. Select **Add Code...**



6. Once you are in the **Code** mode, replace the generated comment:

```
// Write your own code here.
```

with the code below:

```
String uid = DBTools.toSql(Utils.getUserId(e.getPage()),
    JDBCConnection.INTEGER, "IntranetDB");
String from = (String)DBTools.dLookUp("email", "employees",
    "emp_id=" + uid, "IntranetDB");
String to = (String)DBTools.dLookUp("email", "employees",
    "emp_id=" +

DBTools.toSql(e.getRecord().getControl("user_id_assign_to").ge
    tFormattedValue(), JDBCConnection.INTEGER, "IntranetDB"),
    "IntranetDB");
String host = "mysmtphost.com";
String subject = "New task for you";
String body = "The following task was submitted:<br><br>" +
    "Task ID:" + DBTools.dLookUp("max(task_id)",
    "tasks", "user_id_assign_by=" + uid, "IntranetDB") +
    "<br><br>" +
    e.getRecord().getControl("task_desc").getFormattedValue();
SimpleMailer.sendEmail(from, to, subject, body, true, host);
```

As you may have realized by now, the above code sends emails to users to whom the new tasks are assigned. The following is an explanation of the code.

```
String uid = DBTools.toSql(Utils.getUserId(e.getPage()),
    JDBCConnection.INTEGER, "IntranetDB");
```

Sets *uid* variable to the value of the user id of the current user. The static `Utils.getUserId` function is used to retrieve the current user id.

```
String from = (String)DBTools.dLookUp("emp_name", "employees",
"emp_id=" + uid, "IntranetDB");
```

Sets *from* variable to the value of the *emp_name* field for the current user. The static `DBTools.dLookUp` function is used to retrieve a database value.

```
String to = (String)DBTools.dLookUp("email", "employees", "emp_id=" +
DBTools.toSql(e.getRecord().getControl("user_id_assign_to").getFormattedValue(),
JDBCConnection.INTEGER, "IntranetDB"), "IntranetDB");
```

Sets *to* variable to the email of the person that is assigned to the task. The `dLookUp` function is used here to retrieve the appropriate email address.

```
String host = "mysmtphost.com";
```

The SMTP server that will process the e-mail.

```
String subject = "New task for you";
```

The subject of the email to be sent.

```
String body = "The following task was submitted:<br><br>" + "Task ID:"
+ DBTools.dLookUp("max(task_id)", "tasks", "user_id_assign_by=" + uid,
"IntranetDB") + "<br><br>" +
e.getRecord().getControl("task_desc").getFormattedValue();
```

The variable *body* contains the body of the email which will be sent. The `dLookUp` function is used to retrieve the largest task id submitted by the current user (assuming that task ids are created incrementally).

```
SimpleMailer.sendEmail(from, to, subject, body, true, host);
```

Sends the email using the variables created in the above code. The fifth argument specifies that the mail will be sent in HTML format (as opposed to plain text).

Next: [Use the After Update Event to Send Emails](#)

Use the After Update Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. We shall now implement similar functionality in **After Update** Event to notify the assignee when an existing task is updated and reassigned to someone.

1. Click on the *tasks* form in the **Project Explorer**.
2. In the **Properties** window, select the **Events** tab.
3. Add the following **Custom Code** in **After Update** event:

```
4.     String uid = Utils.getUserId(e.getPage());
5.     if (!
uid.equals(e.getRecord().getControl("user_id_assign_to").getFormattedValue()))
6.     {
7.         String from = (String)DBTools.dLookUp("email",
"employees", "emp_id=" + DBTools.toSql(uid,
JDBCConnection.INTEGER, "IntranetDB"), "IntranetDB");
```

```

8.         String to = (String)DBTools.dLookUp("email",
        "employees", "emp_id=" +
9.         DBTools.toSql(e.getRecord().getControl("user_id_assign_to").ge
        tFormattedValue(), JDBCConnection.INTEGER, "IntranetDB"),
        "IntranetDB");
10.        String host = "mysmtphost.com";
11.        String subject = "A task was assigned to you";
12.        String body = "The following task was assigned to
        you:<br><br>" +
13.                "Task ID:" +
        e.getPage().getParameter("task_id") +
14.                "<br><br>" +
        e.getRecord().getControl("task_desc").getFormattedValue();
15.        SimpleMailer.sendEmail(from, to, subject, body, true,
        host);
    }

```

The main differences between the above code and that which was used in the **After Insert** event are as follows:

1. An *if* condition was added to send an email only if a user assigns a task to someone other than himself/herself.
2. *task_id* is retrieved from the URL using the `getParameter` function of Page object. We can use this method because tasks can be updated only if the user arrived at the current page via a URL that contains the task id to be updated. Such a URL would look like this:
http://localhost:8080/TaskManager/tasks_maint.jsp?task_id=9

Next: [Test Email Delivery](#)

Test Email Delivery

Before testing the system:

1. You should add new users to your database with correct email addresses, or modify the existing users by changing their email address.
 - a. You can do this by opening the `Intranet.mdb` database that is located in your project directory.
 - b. Alternatively, you may use the Task Manager itself. Go to the Employees page to view and modify user emails there.
2. Once you have users configured with test emails, save your project and switch to **Live Page** mode to test your system.

Note: You will need MS Access 2000 or higher to manually edit the database file. If your email code worked correctly, you should end up back at the Task List page after adding or modifying a task, and an email should be delivered to the person to whom the task was assigned.

Next: [Implement Record Security in "After Initialize" Event](#)

Step 5

Your Task Management system is now almost complete, except one possibly important feature- security.

Currently everyone can modify and delete any of the tasks. You may want to limit the access so that only the employee assigned to as task can update their tasks. There are many ways of accomplishing this, and we will explain several of them.

1. Click on the "tasks_maint" page in the Project Explorer.
2. Select "Events" tab in the Properties window.
3. Add "Custom Code" to the "After Initialize" event of the page. Once in the Code view, replace the generated comment:

```
/* Write your own code here. */
```

with the code below:

```
String currentTask =
e.getPage().getHttpGetParams().getParameter("task_id");
if (!StringUtils.isEmpty(currentTask))
{
    String uid = Utils.getUserId(e.getPage());
    Object task_id =
DBTools.dLookUp("user_id_assign_to","tasks","task_id="+DBTools
.toSql(currentTask,
                                JDBCConnection.INTEGER, "IntranetDB"),
                "IntranetDB");
    if (!uid.equals(String.valueOf(task_id)))
    {
        e.getPage().getRecord("tasks").setVisible(false);
        // e.getPage().setRedirect("tasks_list.jsp");
        // e.getPage().getRecord("tasks").setAllowUpdate(false);
        // e.getPage().getRecord("tasks").setAllowDelete(false);
    }
}
```

The above code allows you to test the following methods of implementing record security:

Do not show the Task (record form) on the page if the selected task doesn't belong to the current user. An unauthorized user should see a blank page.

You can hide any form on a page by assigning *false* value to the *Visible* property of the form with the *setVisible* method.

The code "*if (!StringUtils.isEmpty(currentTask))*" in the "if" condition specifies that the code should be executed only if a user tries to modify an existing task and he/she is not assigned to it. The "if" also assures that all users can create new tasks. You can test this functionality by inserting the above code into the event, then switching to Live Page mode and trying to modify a task that is not assigned to you, in which case you should see an empty page. Although such functionality may not be very useful, it shows how you can hide forms on a page. You may consider adding another record form to your page that is not updateable and has just the Label fields that show information. Once

you have two forms on the page, you can hide each form programmatically using opposite, mutually exclusive criteria.

Redirect unauthorized users to another page. Only users who are assigned to a task can view the page.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
String currentTask =
e.getPage().getHttpGetParams().getParameter("task_id");
if (!StringUtils.isEmpty(currentTask))
{
    String uid = Utils.getUserId(e.getPage());
    Object task_id =
DBTools.dLookUp("user_id_assign_to","tasks","task_id="+DBTools.toSql(c
urrentTask,
                                JDBCConnection.INTEGER, "IntranetDB"),
    "IntranetDB");
    if (!uid.equals(String.valueOf(task_id)))
    {
        // e.getRecord().setVisible(false);
        e.getPage().setRedirect("tasks_list.jsp");
        // e.getPage().getRecord("tasks").setAllowUpdate(false);
        // e.getPage().getRecord("tasks").setAllowDelete(false);
    }
}
```

The above code shows that you should comment out the previously active line, and uncomment the line that contains "setRedirect".

setRedirect is a method used by CodeCharge Studio to determine if the user should be redirected to another page, for example if a user is not logged in. This variable can be used only on pages that have restricted access and require users to login. You can simply assign the destination page using the *setRedirect* method and the user will be automatically redirected. Test this functionality by modifying the code as shown, then switch to **Live Page** mode and trying to modify a task that is not assigned to you.

Disallowed Update and Delete operations for unauthorized users. Only users who are assigned to a task can edit (delete) it.

You can implement and test this functionality by slightly modifying the above code as shown below:

```
String currentTask =
e.getPage().getHttpGetParams().getParameter("task_id");
if (!StringUtils.isEmpty(currentTask))
{
    String uid = Utils.getUserId(e.getPage());
    Object task_id =
DBTools.dLookUp("user_id_assign_to","tasks","task_id="+DBTools.toSql(c
urrentTask,
                                JDBCConnection.INTEGER, "IntranetDB"),
    "IntranetDB");
```

```
if (!uid.equals(String.valueOf(task_id)))
{
    // e.getRecord().setVisible(false);
    // e.getPage().setRedirect("tasks_list.jsp");
    e.getPage().getRecord("tasks").setAllowUpdate(false);
    e.getPage().getRecord("tasks").setAllowDelete(false);
}
}
```

This code shows how you can manipulate the *setAllowUpdate* and *setAllowDelete* properties of a record form. These properties control record update/delete operations execution. If set to *false*, the operation will not be executed. These properties also control the visibility of the **Update** and **Delete** buttons on the page.

Next: [Conclusion](#)

C# and VB.Net

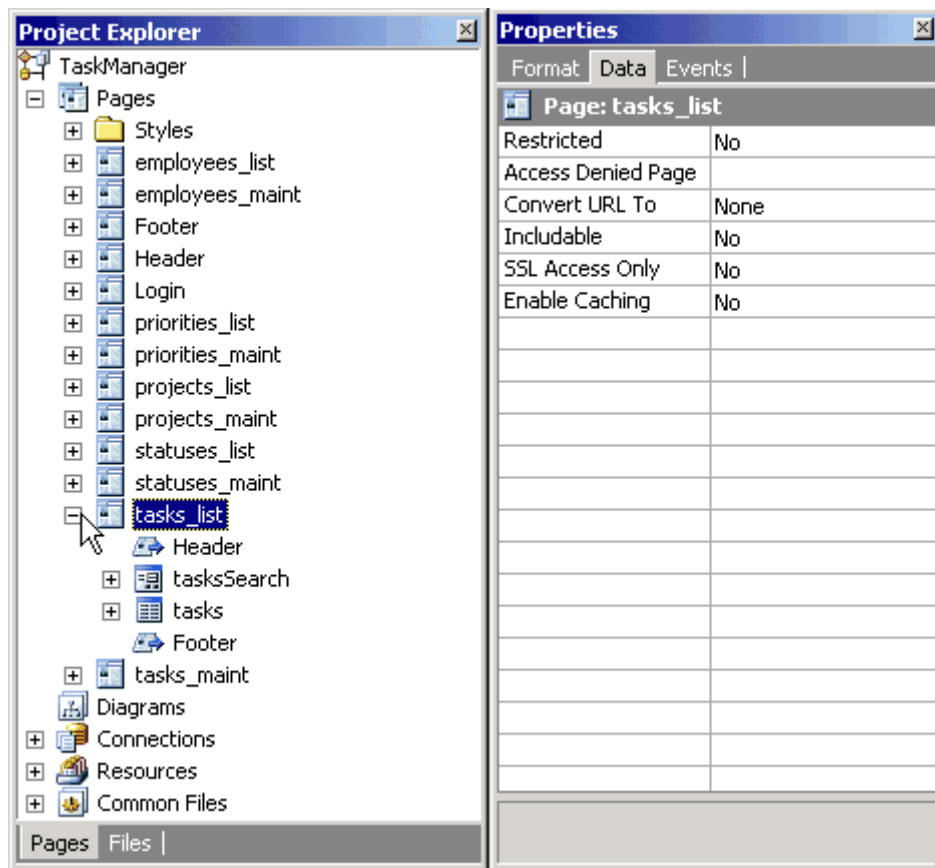
Step 1

Use the Before Show Event to Alter Text Color

The first task in the series of enhancements will be to alter the color of a grid field on our *Task List* page. To be more specific, we will mark the listed tasks assigned to the currently logged in user by showing them in blue text.

1. Open the *tasks_list* page in the **Project Explorer**.
2. Expand the *tasks* grid.
3. In the **Project Explorer** right-click on the *task_name* field and select **Properties**.
4. Under the **Data** tab, set the value of the **Content** property to *HTML*.
5. Select *task_name* label in the **Project Explorer**.
6. Select the **Events** tab in the **Properties** window.
7. Right-click on the **Before Show** event and select **Add Code...**

The **Before Show** event occurs in the program after the field values are assigned, but before being output as HTML. By adding code into this event, you can modify the field value before it is shown.



Next: [Programmatically Control Field's Value](#)

Programmatically Control Field's Value

Once you add Custom Code to the Event, you will see the code-editing window with the appropriate place to enter the new code.

1. Replace this line of code:

C#

```
// Write your own code here.
```

VB.Net

```
' Write your own code here.
```

with the following lines:

C#

```
if((DataItem.user_id_assign_to.Value).ToString() ==
DBUtility.UserLogin )
{
    taskstask_name.Text = "<b><font color='blue'>" +
(DataItem.task_name.Value).ToString() + "</b></font>";
}
```

VB.Net

```
If DataItem.user_id_assign_to.Value = DBUtility.UserLogin Then
    taskstask_name.Text = "<b><font color='blue'>" &
taskstask_name.Text & "</b></font>"
End if
```

The following is an explanation of how the code added above works:

The *if* condition is *true* only if the grid field *task_name* (containing the login name of the user to whom the task is assigned) is equal to the login name of the employee that is currently logged into the system. Therefore, once you login to the system, the program will recognize your tasks by comparing your name to the name of the person that each task is assigned to.

C#

```
taskstask_name.Text = "<b><font color='blue'>" +  
(DataItem.task_name.Value).ToString() + "</b></font>";
```

VB.Net

```
taskstask_name.Text = "<b><font color='blue'>" & taskstask_name.Text &  
"</b></font>"
```

This code is executed if the previous *if* condition is met. It modifies the value of the *task_name* Label. The field value is replaced with the login name value wrapped within HTML code that specifies the font color as blue, and adds HTML ** tag to make the font bold as well.

Next: [Preview the Tasks List Page](#)

Preview the Tasks List Page

1. Save your project.
2. Go to **Live Page** mode to view your working page. If the column "Name" doesn't have any task names highlighted, you are probably not logged in.
3. Since the menu doesn't contain a link to the Login page at this time, you can access it by trying to access one of the restricted pages, such as Task Maintenance.
4. Click on any of the project Ids and you should see the **Login** page.
5. Login as *george/george*, then click on the **Tasks** link on the menu to get back to the Task List page.

Now you should see some task names highlighted, which are the tasks of the user that you logged in as.

Search Tasks

Keyword

Project Select Value ▼

List of Tasks

Id	Project	Priority	Status	Name	Assigned To
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

[Add New](#) 1 of 1

Next: [Modify a Label Field on the Task Maintenance Page](#)

Step 2

Modify a Label Field on the Task Maintenance Page

Now let's make one necessary modification on the Task Maintenance page where you might've noticed that the Label field *Assigned By* doesn't display the employee name, but the ID, as shown below. This is because the *tasks* table contains only the user ID, while the *employees* table contains the actual user names.

There are several potential methods of dealing with the above issue as explained below:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, just like you did with the grid on the Task List page. Unfortunately, queries that contain multiple tables may not be updateable by their nature, and thus your whole record form may stop working. In other words, if you specified that you want to use a query containing *tasks* and *employees* table in your record form, then if you assigned a task to someone else, the program wouldn't know if you wanted to update the *tasks* table with the new *employee_id*, or if you wanted to update the *employees* table and change employee's name.

Thus if you used multiple tables as a data source for the record form, you would also need to specify Custom Insert, Custom Update and Custom Delete operations in record form's properties, to specify which database fields should be updated with corresponding values entered on the page. This approach looks like too much effort just for displaying one additional value on the page.

2. Use an Event Procedure to insert custom code where you can programmatically output the desired value. This method is very flexible, as it allows you to extend the generated code by adding your own. The next step describes this method in detail.

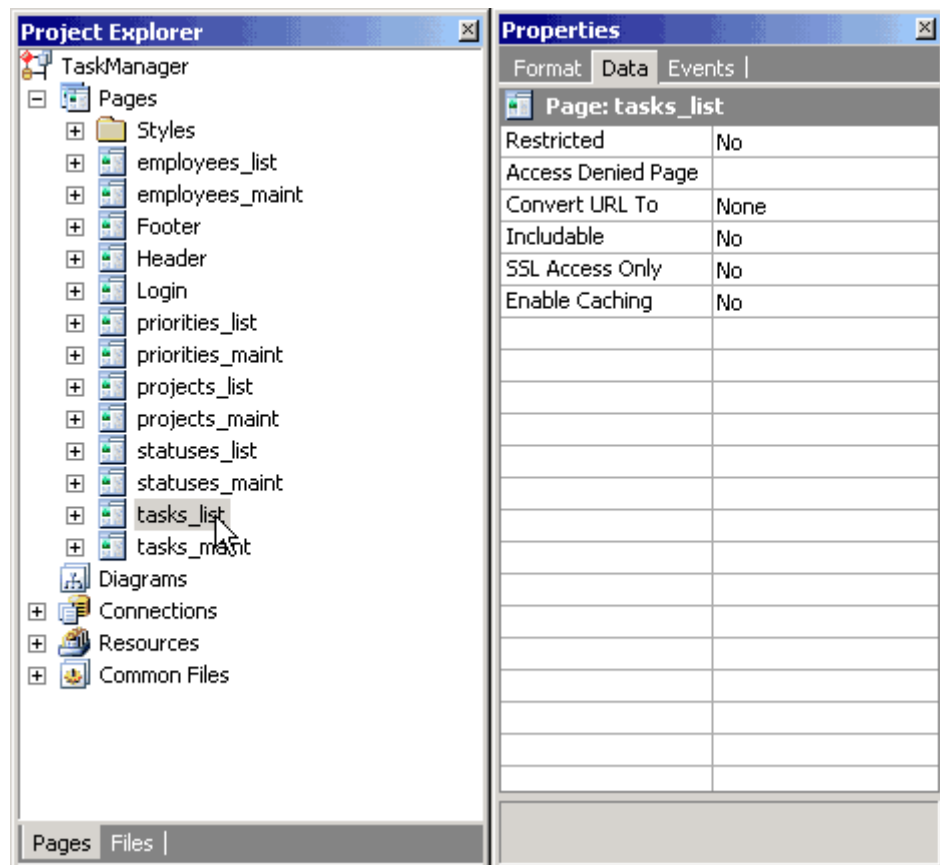
[Employees](#) [Priorities](#) [Projects](#) [Statuses](#) [Tasks](#)

Add/Edit Tasks	
Project	CodeCharge
Priority	Highest
Status	Closed
Type	Question
Name	Fix ALL bugs
Description	Staying up at night coding? Get CodeCharge, go home, get rest.
Assigned To	george
Finish Date	03.02.2003
Assigned By	3
Start Date	02.02.2003
<input type="button" value="Submit"/> <input type="button" value="Delete"/>	

Next: [Use the Before Show Event to Alter a Label's Value](#)

Use the Before Show Event to Alter a Label's Value

1. Select the Label *user_id_assign_by* in the **Project Explorer**.
2. In the **Properties** window click on the **Data** tab.
3. Select *Text* in the **Data Type** property.
4. Switch to the **Events** tab.
5. Right-click on **Before Show** event and select **Add Code....** .
CodeCharge Studio should then automatically switch to **Code** view.



6. Once in Code view, replace the following text:

C#

```
// Write your own code here.
```

VB.Net

```
` Write your own code here.
```

with the following:

C#

```
DataAccessObject NewDao = Settings.IntranetDBDataAccessObject;
if (IsInsertMode)
{
    SqlCommand userNameCmd = new SqlCommand("SELECT emp_name
FROM employees " +
                                           "WHERE emp_id=" +
NewDao.ToSql(DBUtility.UserId.ToString(), FieldType.Integer),
NewDao);
    tasksuser_id_assign_by.Text =
userNameCmd.ExecuteScalar().ToString();
}
else
{
    SqlCommand userNameCmd = new SqlCommand( "SELECT emp_name
FROM employees " +
                                           "WHERE emp_id=" +
NewDao.ToSql(item.user_id_assign_by.GetFormattedValue,
FieldType.Integer), NewDao);
```

```

tasksuser_id_assign_by.Text =
userNameCmd.ExecuteScalar().ToString();
}

```

VB.Net

```

Dim NewDao As DataAccessObject =
Settings.IntranetDBDataAccessObject
If IsInsertMode = True Then
    tasksuser_id_assign_by.Text =
Convert.ToString(NewDao.ExecuteScalar( _
                    "SELECT emp_name FROM
employees WHERE emp_id=" & _
                    DBUtility.UserId ))
Else
    tasksuser_id_assign_by.Text =
Convert.ToString(NewDao.ExecuteScalar( _
                    "SELECT emp_name FROM
employees WHERE emp_id=" & _
                    Convert.ToString(
tasksuser_id_assign_by.Text)))
End if

```

In the above code snippet, we have an *if* block which checks the value of the Boolean variable *IsInsertMode*. This variable is generated by CodeCharge Studio within the *Before Show* event of the record forms. Since the same CodeCharge Studio record form can be used to update records, delete records as well as insert new records, this variable is used to determine the state of the record form at runtime. If this variable is *true*, then the record form is inserting a new record, else it's either updating or deleting the record.

Both the *if* and *else* blocks define a *SqlCommand* object, which as discussed earlier helps you to execute commands against the database. The SQL query passed to both methods returns the name of the user stored in the *emp_name* field of the *employees* table. Once the name of the user is retrieved, it's set to the Label *tasksuser_id_assign_by*. The only difference between the two SQL queries is that if the user is inserting a new record i.e. the variable *IsInsertMode* is *true*, you pass the user ID of the currently logged user from the *DBUtility.UserId* variable. If the user is updating or deleting the record, then you pass the user ID of the user who was previously assigned the task. The *user_id_assign_by* property of the *item* object contains that value retrieved from the database.

The whole code reads approximately as follows:

- If a new record is being created:
 - Use the value of *DBUtility.UserId* to obtain the name of the currently logged in user and assign this name to the value of the *tasksuser_id_assign_by* Label.
- If a record is being edited:
 - Use the value of *tasksuser_id_assign_by* to obtain the name of the user to whom the task was assigned and assign this name to the value of the *tasksuser_id_assign_by* Label.

Next: [Add an Hidden "Assigned By" Field to Auto-Update New Tasks](#)

Step 3

Add an Hidden "Assigned By" Field to Auto-Update New Tasks

You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though the employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.

1. Add a Hidden field to your page from the **Forms** tab of the **Toolbox**.

This field type isn't visible in the browser, but will be used to store a value and update the database.

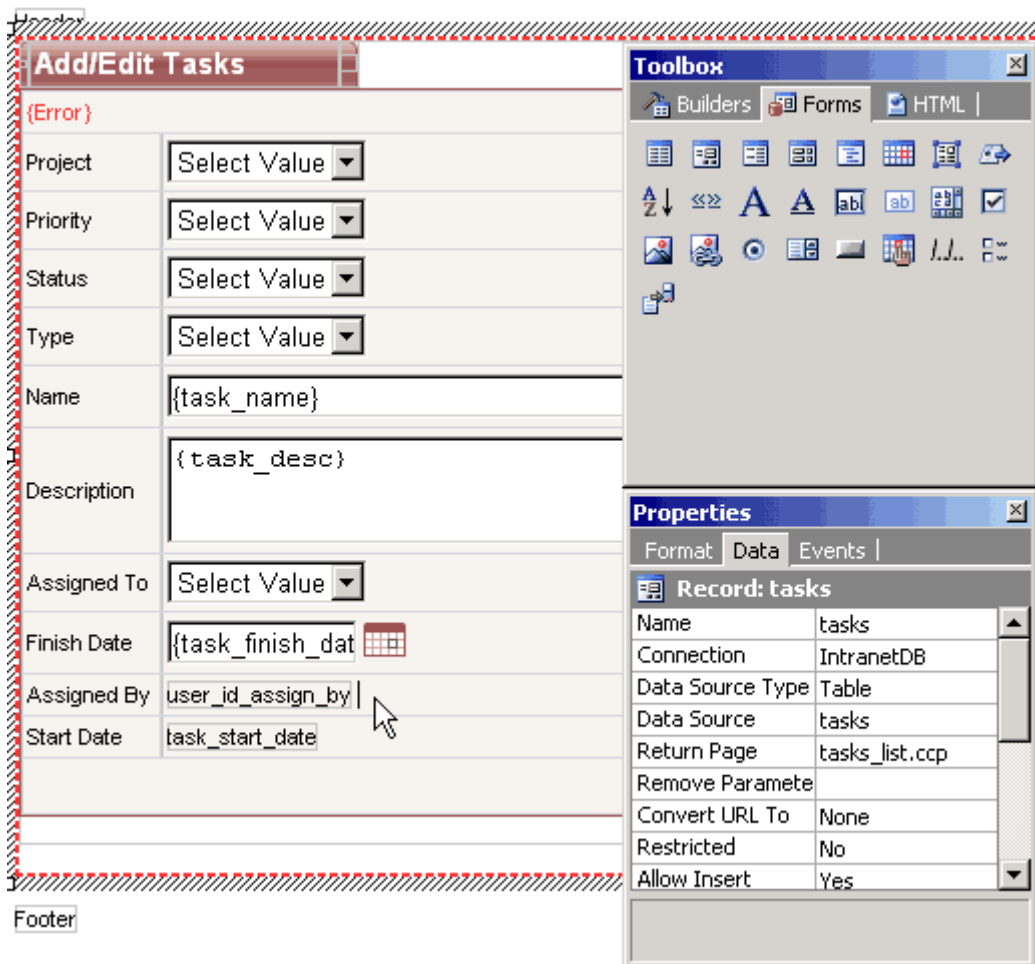
2. Configure the new field by setting its properties as follows:

- **Name:** *assigned_by* - the name of the newly added Hidden field.

This can be any name you choose.

- **Control Source:** *user_id_assign_by* - the database field/column that will be used to retrieve field's value and will be updated with the new value, if it changes.
- **Data Type:** *Integer* - the type of the value bound to the control source. Our user/employee id's are numeric.
- **Default:** *DBUtility.UserId* - default value for this field if it is empty.

DBUtility.UserId is a CodeCharge property that retrieves the user id of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.

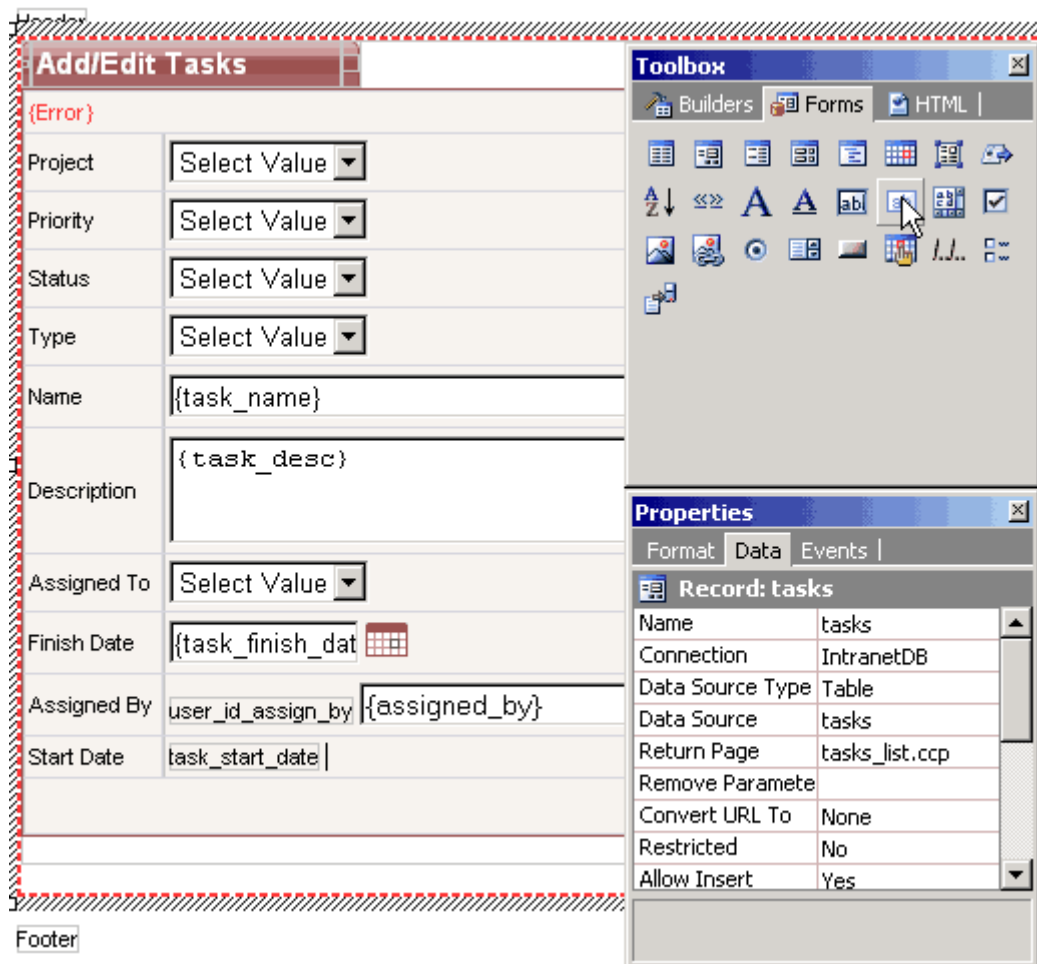


Next: [Add a Hidden "Date Created" Field to the Record Form](#)

Add a Hidden "Date Created" Field to the Record Form

Now add another Hidden field to your page, which will be used to submit the current date and time to the *date_assign* field in the database.

1. Configure the new field as follows:
 - **Name:** *date_created*
 - **Control Source:** *task_start_date*
 - **Data Type:** *Date*
 - **Default:** *CurrentDateTime* - The *CurrentDateTime* value allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.
2. Click on the *task_start_date* field.
3. In the **Properties** window, set its Default value to *CurrentDateTime*. This is so that the Label field can display the date since the hidden field will not be visible to the user.



Next: [Test the Label and Hidden Fields](#)


Test the Label and Hidden Fields

Finally,

1. Switch to Live Page mode.
2. Select or add a Task and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.

3. Don't forget to save it!

Add/Edit Tasks	
Project	My Project
Priority	Highest
Status	In progress
Type	Task
Name	Finish my project
Description	Implement "Assigned By" Label to show values from another table
Assigned To	alexander
Finish Date	<input type="text"/> 
Assigned By	George Pennington
Start Date	13.09.2005 8:07:17
<input type="button" value="Add"/>	

Next: [Programming the Record Form](#)

Step 4

Programming the Record Form

Now you've created a simple task management application, but how do you extend it to be more practical and useful? In this section, you will get a glimpse of how to implement practical and sophisticated applications by adding programming code and actions that enhance the application's functionality.

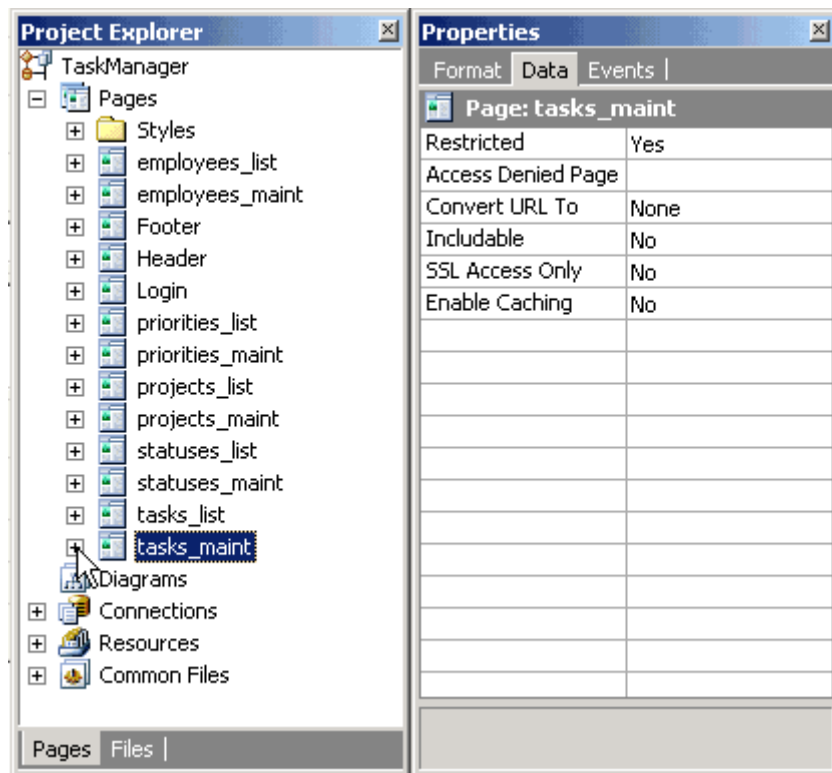
You will learn how to:

- Send email notifications to the person that the task is being assigned to
- Allow only the person assigned to the task to modify it

Next: [Add Code in the After Insert Event to Send Emails](#)

Add Code in the After Insert Event to Send Emails

1. Select the *tasks* form by selecting it in the **Project Explorer**, or clicking anywhere within the form's caption.
2. In the **Properties** window click on the **Events** tab.
3. Select the **After Insert** event.
4. Click on the **[+]** button, then select **Add Code...**



5. Once you are in the Code view, replace the generated comment:

C#

```
// Write your own code here.
```

VB.Net

```
` Write your own code here.
```

with the code below:

C#

```
Dim NewDao As DataAccessObject =
Settings.IntranetDBDataAccessObject
if(ExecuteFlag&&!ErrorFlag){
    SqlCommand userEmail = new SqlCommand( "SELECT email FROM
" +
    "employees WHERE emp_id=" + DBUtility.UserId,
    Settings.IntranetDBDataAccessObject );

    SqlCommand assignedUserEmail = new SqlCommand( "SELECT email
FROM " +
    "employees WHERE emp_id=" +
    item.user_id_assign_to.Value.ToString(),
    Settings.IntranetDBDataAccessObject );

    SqlCommand taskId = new SqlCommand( "SELECT max(task_id) FROM
" +
    "tasks WHERE user_id_assign_by=" + DBUtility.UserId,
    Settings.IntranetDBDataAccessObject );
```

```
System.Web.Mail.MailMessage newMessage = new
System.Web.Mail.MailMessage();
```

```
newMessage.From = userEmail.ExecuteScalar().ToString();
newMessage.To = assignedUserEmail.ExecuteScalar().ToString();
newMessage.Subject = "New task for you!";
newMessage.BodyFormat = System.Web.Mail.MailFormat.Html;
newMessage.Body = "The following task was submitted:<br><br>"
+
  "Task ID :"+ taskId.ExecuteScalar().ToString() +
  "<br><br>" + item.task_desc.Value;

System.Web.Mail.SmtpMail. SmtpServer = "localhost";
System.Web.Mail.SmtpMail.Send(newMessage);
}
```

VB.Net

```
If ExecuteFlag And (Not ErrorFlag) Then
  Dim userEmail, assignedUserEmail, taskId As String

  userEmail =
  Convert.ToString(Settings.IntranetDBDataAccessObject.ExecuteSc
  alar( _
    "SELECT email FROM employees WHERE emp_id=" &
  DBUtility.UserId))

  assignedUserEmail =
  Convert.ToString(Settings.IntranetDBDataAccessObject.ExecuteSc
  alar( _
    "SELECT email FROM employees WHERE emp_id=" &
  Convert.ToString(item.user_id_assign_to.Value)))

  taskId =
  Convert.ToString(Settings.IntranetDBDataAccessObject.ExecuteSc
  alar( _
    "SELECT max(task_id) FROM tasks WHERE
  user_id_assign_by=" & DBUtility.UserId))

  Dim newMessage As New System.Web.Mail.MailMessage()

  newMessage.From = userEmail
  newMessage.To = assignedUserEmail
  newMessage.Subject = "New task for you!"
  newMessage.BodyFormat = System.Web.Mail.MailFormat.Html
  newMessage.Body = "The following task was submitted:<br><br>
  Task ID :" & taskId & _
    "<br><br>" & item.task_desc.Value

  System.Web.Mail.SmtpMail. SmtpServer = "localhost"
  System.Web.Mail.SmtpMail.Send(newMessage)
```

```
End If
```

As you may have realized by now, the above code sends emails to users to whom the new tasks are assigned. Here is additional information you should be aware of:

1. The above code use the classes provided by the .NET Framework to send e-mails, so you do not need to install extra components.
2. The .NET Framework classes rely on the CDO component to send e-mails; hence you should need an SMTP service installed on the server hosting this application.

The following is an explanation of the above code.

```
if(ExecuteFlag&&!ErrorFlag){
SqlCommand userEmail = new SqlCommand( "SELECT email FROM " +
"employees WHERE emp_id=" + DBUtility.UserId,
Settings.IntranetDBDataAccessObject );

SqlCommand assignedUserEmail = new SqlCommand( "SELECT email FROM " +
"employees WHERE emp_id=" + item.user_id_assign_to.Value.ToString(),
Settings.IntranetDBDataAccessObject );

SqlCommand taskId = new SqlCommand( "SELECT max(task_id) FROM " +
"tasks WHERE user_id_assign_by=" + DBUtility.UserId,
Settings.IntranetDBDataAccessObject );
```

In the above code snippet three *SqlCommand* objects are defined. As you would remember, these objects are used to execute queries against the database. The first SQL query is used to retrieve the email address of the currently logged-in user. The second SQL query retrieves the email address of the user to whom the task is assigned. The *user_id_assign_to* property of the *item* object is used to get the user id of the user to whom the task is assigned. The last SQL query is used to get the task id. The last inserted task id can be obtained using different methods with different databases. Unfortunately, MS Access doesn't support the retrieval of the last inserted record; therefore you will need to use the *SqlCommand* object to lookup the largest task id submitted by the current user (assuming that task ids are created incrementally).

```
System.Web.Mail.MailMessage newMessage = new
System.Web.Mail.MailMessage();
```

Creates a *MailMessage* object, which is under the namespace *System.Web.Mail*.

```
newMessage.From = userEmail
newMessage.To = assignedUserEmail
newMessage.Subject = "New task for you!"
newMessage.BodyFormat = System.Web.Mail.MailFormat.Html
newMessage.Body = "The following task was submitted:<br><br> Task ID
:" & taskId & _
"<br><br>" & item.task_desc.Value
```

The above code executes the query against the database and sets the various properties of the *MailMessage* object. Set the "BodyFormat" property of the *MailMessage* object to *Html* so HTML content can be sent (as opposed to plain text). The body of the email consists of the task description and the task id.

```
System.Web.Mail.SmtpMail.SmtpServer = "localhost"
System.Web.Mail.SmtpMail.Send(newMessage)
```

Set the address of the SMTP server which will be used to send the email then use the static *Send* method of the class *SmtpMail* to send the email. If you don't have an SMTP server on the local machine, you can enter the address of an external SMTP server which you have access to e.g. mail.yourdomain.com.

Next: [Use the After Update Event to Send Emails](#)

Use the After Update Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. We shall now implement similar functionality in **After Update** Event to notify the assignee when an existing task is updated and reassigned to someone.

1. Click on the *tasks* form in the **Project Explorer**.
2. In the **Properties** window, select the **Events** tab.
3. Add the following **Custom Code** in **After Update** event:

C#

```
if (ExecuteFlag && !ErrorFlag && DBUtility.UserId.ToString() !=
item.user_id_assign_to.Value.ToString())
{
    SqlCommand userEmail = new SqlCommand("SELECT email FROM " +
"employees WHERE emp_id=" + DBUtility.UserId,
Settings.IntranetDBDataAccessObject);
    SqlCommand assignedUserEmail = new SqlCommand("SELECT email
FROM " +
"employees WHERE emp_id=" +
item.user_id_assign_to.Value.ToString(),
Settings.IntranetDBDataAccessObject);
    System.Web.Mail.MailMessage newMessage = new
System.Web.Mail.MailMessage();
    newMessage.From = userEmail.ExecuteScalar().ToString();
    newMessage.To =
assignedUserEmail.ExecuteScalar().ToString();
    newMessage.Subject = " A task was assigned to you";
    newMessage.BodyFormat = System.Web.Mail.MailFormat.Html;
    newMessage.Body = "The following task was assigned to
you:<br><br>" +
"Task ID :" + Request.QueryString["task_id"].ToString() +
"<br><br>" + item.task_desc.Value;
    System.Web.Mail.SmtpMail.SmtpServer = "localhost";
    System.Web.Mail.SmtpMail.Send(newMessage );
}
```

VB.Net

```
If ExecuteFlag AndAlso (Not ErrorFlag) AndAlso
Convert.ToString (DBUtility.UserId) <>
Convert.ToString(item.user_id_assign_to.Value) Then
```

```

Dim userEmail, assignedUserEmail As String

userEmail =
Convert.ToString(Settings.IntranetDBDataAccessObject.ExecuteSc
alar( _

    "SELECT email FROM employees WHERE emp_id=" &
DBUtility.UserId))
    assignedUserEmail =
Convert.ToString(Settings.IntranetDBDataAccessObject.ExecuteSc
alar( _

    "SELECT email FROM employees WHERE emp_id=" &
Convert.ToString(item.user_id_assign_to.Value)))

Dim newMessage As New System.Web.Mail.MailMessage()

newMessage.From = userEmail
newMessage.To = assignedUserEmail
newMessage.Subject = "A task was assigned to you"
newMessage.BodyFormat = System.Web.Mail.MailFormat.Html
newMessage.Body = "The following task was assigned to you
:<br><br> Task ID :" & _
    Convert.ToString(Request.QueryString("task_id")) &
"<br><br>" & item.task_desc.Value

System.Web.Mail.SmtpMail.SmtpServer = "localhost"
System.Web.Mail.SmtpMail.Send(newMessage)

End if

```

The main differences between the above code and the one you used in *After Insert* event are as follows:

1. An "if" condition was added to send an email only if a user assigns a task to someone other than himself/herself
2. *task_id* is retrieved from the URL using the Request.QueryString indexer. You can use this method because tasks can be updated only if the user arrived at the current page via a URL that contains the task id to be updated. A such a URL would look like
this:http://localhost/TaskManager/tasks_maint.aspx?task_id=9

Next: [Test Email Delivery](#)

Test Email Delivery

Before testing the system:

1. You should add new users to your database with correct email addresses, or modify the existing users by changing their email address.
 - a. You can do this by opening the Intranet.mdb database that is located in your project directory.

- b. Alternatively, you may use the Task Manager itself. Go to the Employees page to view and modify user emails there.
2. Once you have users configured with test emails, save your project and switch to **Live Page** mode to test your system.

Note: You will need MS Access 2000 or higher to manually edit the database file. If your email code worked correctly, you should end up back at the Task List page after adding or modifying a task, and an email should be delivered to the person to whom the task was assigned.

Next: [Implement Record Security in After Initialize Event](#)

Step 5

Implement Record Security in After Initialize Event

Your Task Management system is now almost complete, except one possibly important feature- security.

Currently everyone can modify and delete any of the tasks. You may want to limit the access so that only the employee assigned to as task can update their tasks. There are many ways of accomplishing this, and we will examine several of them.

1. Click on the *tasks_maint* page in the **Project Explorer**.
2. Select **Events** tab in the **Properties** window.
3. Add **Custom Code** to the **After Initialize** event of the page as follows. Once in the **Code** mode, replace the generated comment:

C#

```
// Write your own code here.
```

VB.Net

```
' Write your own code here.
```

with the code below:

C#

```
if(Request.QueryString["task_id"] != null)
{
    IntegerField task_id = new IntegerField("",
Request.QueryString["task_id"]);
    SqlCommand taskCmd = new SqlCommand("SELECT
user_id_assign_to FROM " + "tasks WHERE task_id=" +
task_id.GetFormattedValue(),
Settings.IntranetDBDataAccessObject);
    int assignedUserId = (int)taskCmd.ExecuteScalar();

    if(int.Parse(DBUtility.UserId.ToString()) != assignedUserId)
    {
        tasksHolder.Visible = false;
        // Response.Redirect("tasks_list.aspx");
        // tasksOperations.AllowUpdate = false;
        // tasksOperations.AllowDelete = false;
    }
}
```

```
}  
}
```

VB.Net

```
If Request.QueryString("task_id") <> " " Then  
    Dim task_id As IntegerField = new IntegerField("",  
Request.QueryString("task_id"))  
    Dim taskCmd As SqlCommand = New SqlCommand("SELECT  
user_id_assign_to FROM " & "tasks WHERE task_id=" &  
task_id.GetFormattedValue(),  
Settings.IntranetDBDataAccessObject)  
    Dim assignedUserId As Integer =  
CInt(taskCmd.ExecuteScalar())  
  
    If Convert.ToInt32(DBUtility.UserId) <> assignedUserId Then  
        tasksHolder.Visible = false  
        ' Response.Redirect( "tasks_list.aspx" )  
        ' tasksOperations.AllowUpdate = false  
        ' tasksOperations.AllowDelete = false  
    End If  
End if
```

The above code allows you to test the following methods of implementing record security:

Do not show the Task (record form) on the page if the selected task doesn't belong to the current user. An unauthorized user should see a blank page.

You can hide any form on a page by assigning *false* value to the *Visible* property of the table holding the record form. First, check for the presence of the query string variable *task_id*, which indicates the record form is in update or delete mode, since you want to restrict users to only view/modify tasks assigned to them. The *if* block also ensures that all users can create new tasks. You can test this functionality by inserting the above code into the event, then switching to **Live Page** mode and trying to modify a task that is not assigned to you, in which case you should see an empty page (with just the header). Although such functionality may not be very useful, it shows how you can hide forms on a page. You may consider adding another record form to your page that is not updateable and has just the Label fields that show task information. Once you have two forms on the page, you can hide each form programmatically using mutually exclusive criteria.

Redirect unauthorized users to another page. Only users, who are assigned to a task can view the page.

You can implement and test this functionality by slightly modifying the above code as shown below:

C#

```
if(Request.QueryString["task_id"] != null)  
{  
    IntegerField task_id = new IntegerField("",  
Request.QueryString["task_id"]);
```

```

SqlCommand taskCmd = new SqlCommand("SELECT user_id_assign_to FROM "
+ "tasks WHERE task_id=" +
                                task_id.GetFormattedValue(),
Settings.IntranetDBDataAccessObject);

int assignedUserId = (int)taskCmd.ExecuteScalar();

if(int.Parse(DBUtility.UserId.ToString()) != assignedUserId)
{
    // tasksHolder.Visible = false;
    Response.Redirect("tasks_list.aspx");
    // tasksOperations.AllowUpdate = false;
    // tasksOperations.AllowDelete = false;
}
}

```

VB.Net

```

If Request.QueryString("task_id") <> " " Then
    Dim task_id As IntegerField = new IntegerField("",
Request.QueryString("task_id"))
    Dim taskCmd As SqlCommand = New SqlCommand("SELECT user_id_assign_to
FROM " & "tasks WHERE task_id=" &_
                                task_id.GetFormattedValue(),
Settings.IntranetDBDataAccessObject)
    Dim assignedUserId As Integer = CInt(taskCmd.ExecuteScalar())

    If Convert.ToInt32(DBUtility.UserId) <> assignedUserId Then
        ' tasksHolder.Visible = false
        Response.Redirect( "tasks_list.aspx" )
        ' tasksOperations.AllowUpdate = false
        ' tasksOperations.AllowDelete = false
    End If
End if

```

The above code shows that you should comment out the previously active line, and uncomment the line that starts with *Response.Redirect*. The *Redirect* method of the *Response* object is used to redirect the user to a new page. You can simply assign the destination page to the *Redirect* method and the page will be automatically redirected. Test this functionality by modifying the code as shown, and then switch to **Live Page** mode and try to modify a task that is not assigned to you.

Disallowed Update and Delete operations for unauthorized users. Only users who are assigned to a task, can edit (delete) it.

You can implement and test this functionality by slightly modifying the above code as shown below:

C#

```

if(Request.QueryString["task_id"] != null)
{

```



```

IntegerField task_id = new IntegerField("",
Request.QueryString["task_id"]);

SqlCommand taskCmd = new SqlCommand("SELECT user_id_assign_to FROM "
+ "tasks WHERE task_id=" +

                                task_id.GetFormattedValue(),
Settings.IntranetDBDataAccessObject);

int assignedUserId = (int)taskCmd.ExecuteScalar();

if(int.Parse(DBUtility.UserId.ToString()) != assignedUserId)
{
    //tasksHolder.Visible = false;
    //Response.Redirect("tasks_list.aspx");
    tasksOperations.AllowUpdate = false;
    tasksOperations.AllowDelete = false;
}
}

```

VB.Net

```

If Request.QueryString("task_id") <> " " Then
    Dim task_id As IntegerField = new IntegerField("",
Request.QueryString("task_id"))
    Dim taskCmd As SqlCommand = New SqlCommand("SELECT user_id_assign_to
FROM " & "tasks WHERE task_id=" &_
                                task_id.GetFormattedValue(),
Settings.IntranetDBDataAccessObject)
    Dim assignedUserId As Integer = CInt(taskCmd.ExecuteScalar())

    If Convert.ToInt32(DBUtility.UserId) <> assignedUserId Then
        ' tasksHolder.Visible = false
        ' Response.Redirect( "tasks_list.aspx" )
        tasksOperations.AllowUpdate = false
        tasksOperations.AllowDelete = false
    End If
End if

```

This code shows how you can manipulate the *UpdateAllowed* and *DeleteAllowed* properties of a record form. These properties control record update/delete operations execution. If set to *false*, the operation will not be executed.

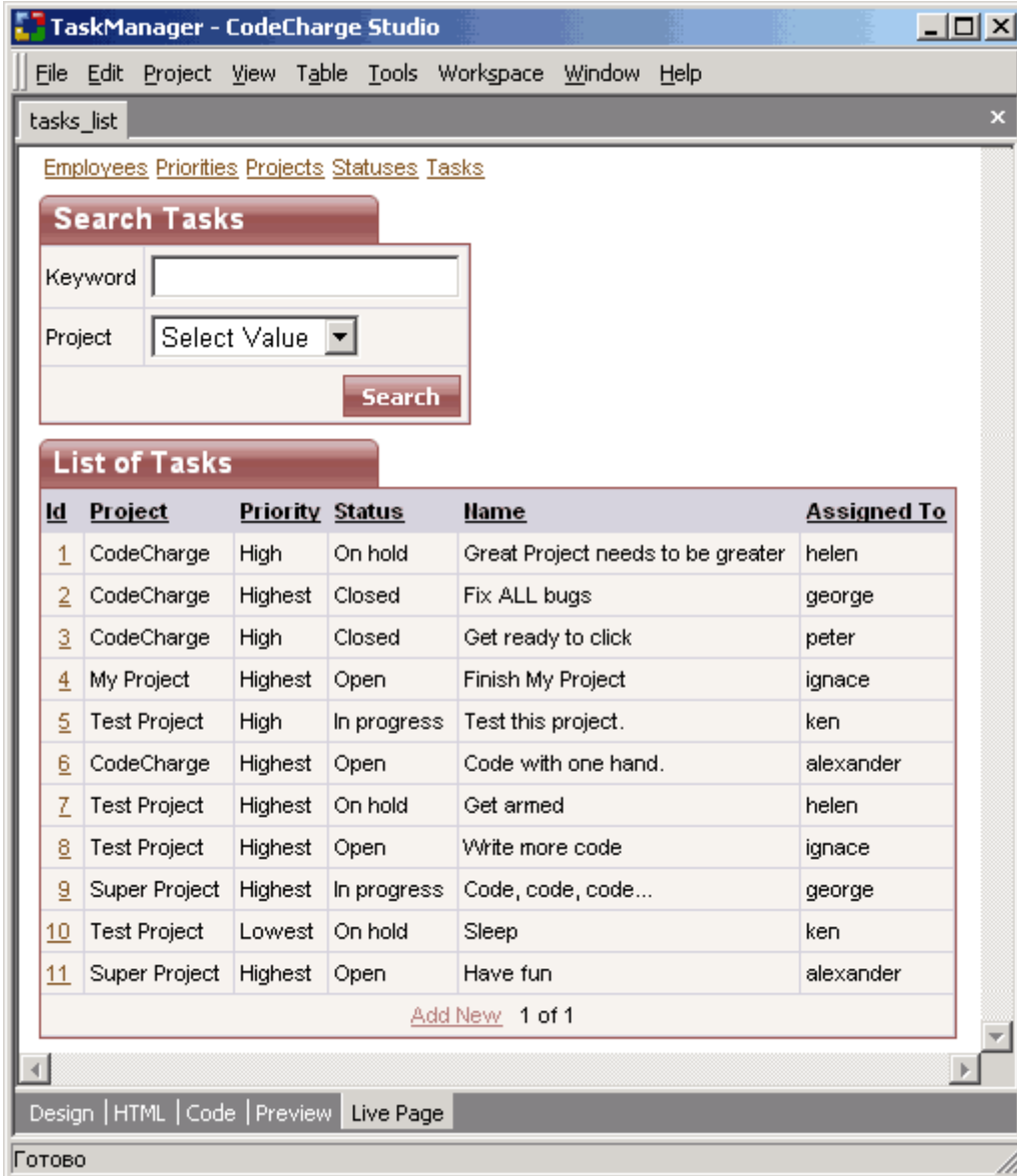
These properties also control the visibility of the **Update** and **Delete** buttons on the page.

Next: [Conclusion](#)

Conclusion

During the course this chapter, you've used the Application Builder to create a simple Task Management application. Although many additional features and improvements can be implemented, you should now be familiar with CodeCharge Studio's interface and many of its features. Refer to the rest of the chapters in the User Manual for more information about specific features.

Enjoy!



The screenshot shows the 'TaskManager - CodeCharge Studio' window. It features a menu bar with 'File', 'Edit', 'Project', 'View', 'Table', 'Tools', 'Workspace', 'Window', and 'Help'. Below the menu is a tab labeled 'tasks_list'. The main content area has navigation links for 'Employees', 'Priorities', 'Projects', 'Statuses', and 'Tasks'. A 'Search Tasks' section contains a 'Keyword' text input, a 'Project' dropdown menu with 'Select Value' selected, and a 'Search' button. Below this is a 'List of Tasks' section with a table containing 11 rows of task data. At the bottom of the table is an 'Add New' link and the text '1 of 1'. The window footer includes a mode selector with 'Design', 'HTML', 'Code', 'Preview', and 'Live Page' options, and a status indicator 'Готово'.

<u>Id</u>	<u>Project</u>	<u>Priority</u>	<u>Status</u>	<u>Name</u>	<u>Assigned To</u>
1	CodeCharge	High	On hold	Great Project needs to be greater	helen
2	CodeCharge	Highest	Closed	Fix ALL bugs	george
3	CodeCharge	High	Closed	Get ready to click	peter
4	My Project	Highest	Open	Finish My Project	ignace
5	Test Project	High	In progress	Test this project.	ken
6	CodeCharge	Highest	Open	Code with one hand.	alexander
7	Test Project	Highest	On hold	Get armed	helen
8	Test Project	Highest	Open	Write more code	ignace
9	Super Project	Highest	In progress	Code, code, code...	george
10	Test Project	Lowest	On hold	Sleep	ken
11	Super Project	Highest	Open	Have fun	alexander

Back to: [Tutorials](#)