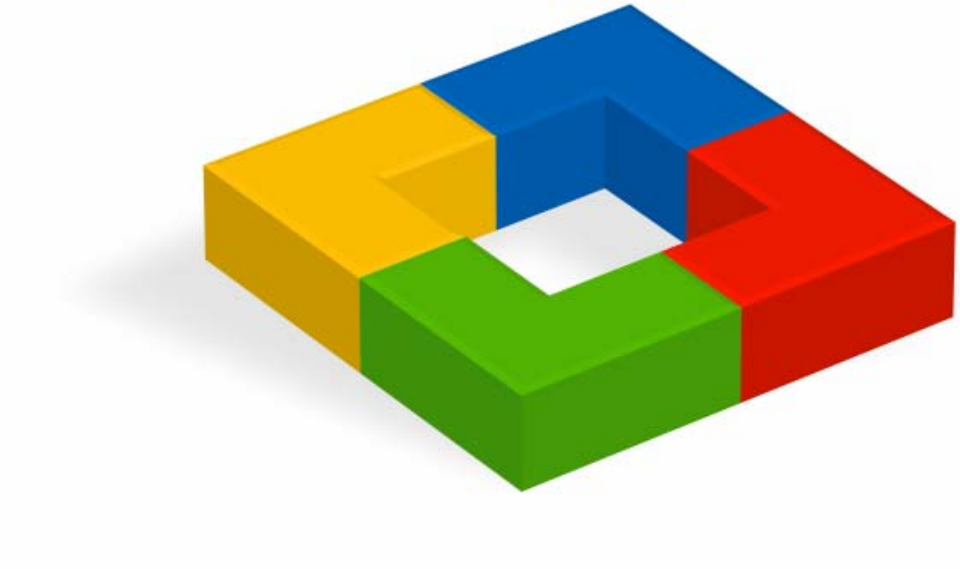
CodeCharge Studio Tutorial



CodeCharge Studio™

Release Date: August 5, 2002

Contents

Introduction

Welcome to the Tutorial on rapidly creating web applications with CodeCharge Studio. You will find that CodeCharge Studio is not only easy to use but is a vital and powerful code generator for visually creating professional web database solutions.

This tutorial consists of two chapters; each designed to take you through different approaches to creating web applications from scratch.

It is assumed that you have some knowledge of databases and web servers and it is required that you have a working web server such as IIS or PWS for Windows, or Apache for Linux already installed and configured. For the purpose of this tutorial, we recommend using IIS or PWS server for Windows. CodeCharge Studio also comes with sample MS Access databases that can be used during the course of this tutorial.

If you run into technical difficulties, refer to the Appendix section for help and possible solutions.

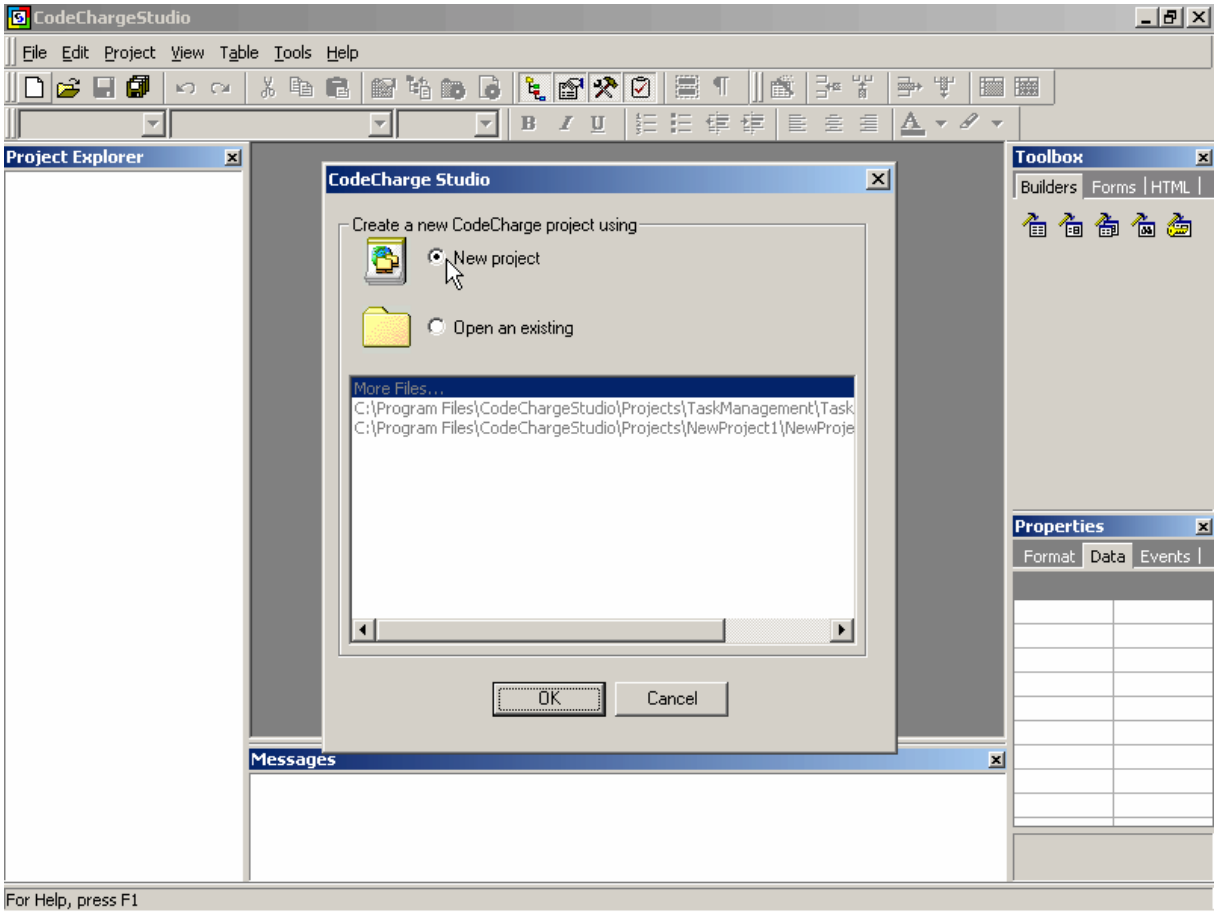
Chapter 1: Creating a Task Management System with the Application Builder

The Application Builder is a powerful feature of CodeCharge Studio that can automatically convert a database into a working web application. By following a few short steps, you can create administrative web pages for your SQL database, or create a skeleton web application that you can then extend into a full-featured solution. In this chapter, we will take you through the steps of creating a ready-to-use Task Management System created mainly using the Application Builder then finalized by utilizing other features of CodeCharge Studio.

Using the Application Builder

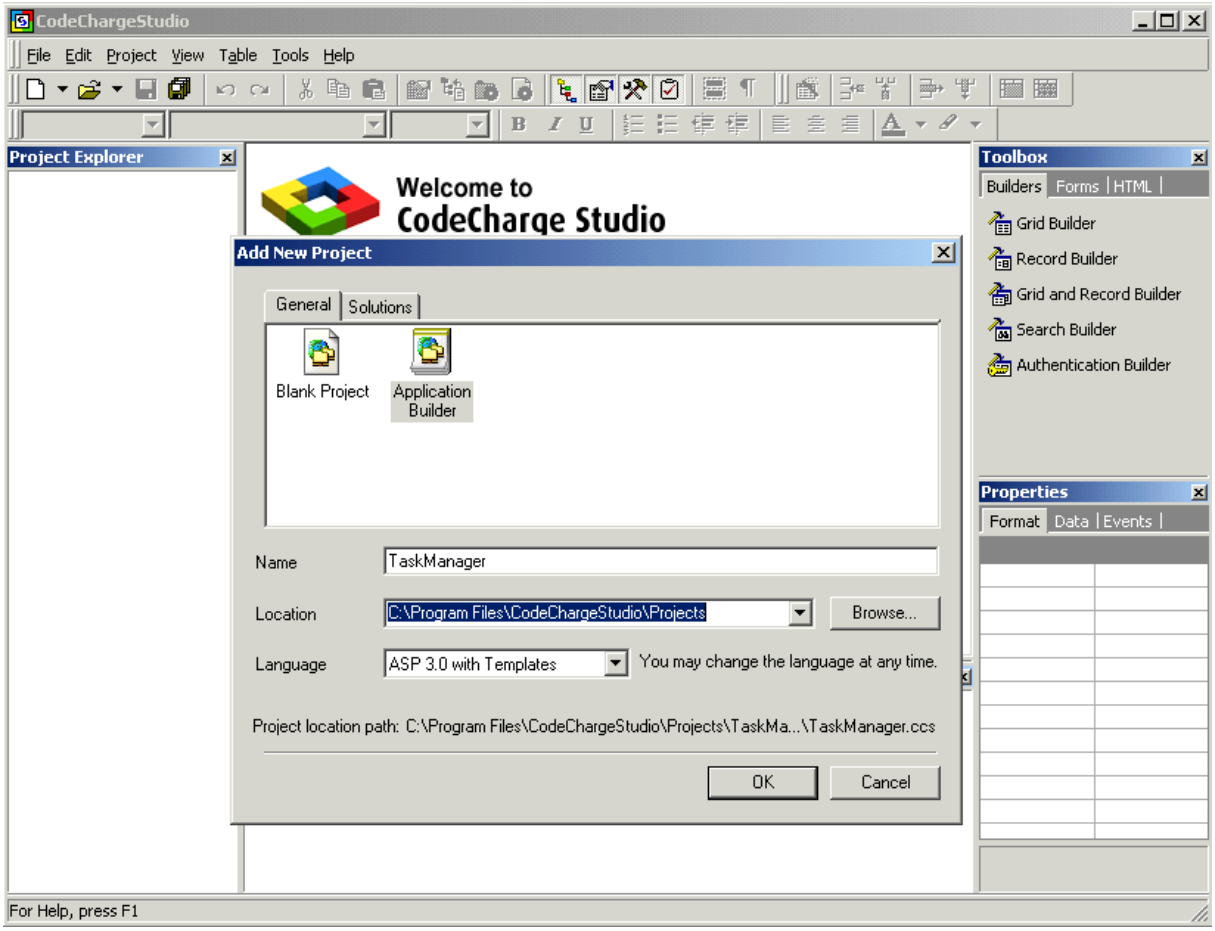
Create a New Project

Start CodeCharge Studio and select “New project” in the initial screen.



Launch the Application Builder

To start the Application Builder, specify the name of the new project (TaskManager), the location on the disk where the project will be published, and the programming language then double-click on the “Application Builder” icon. You can also select the “Application Builder” icon then click the [OK] button to launch it.



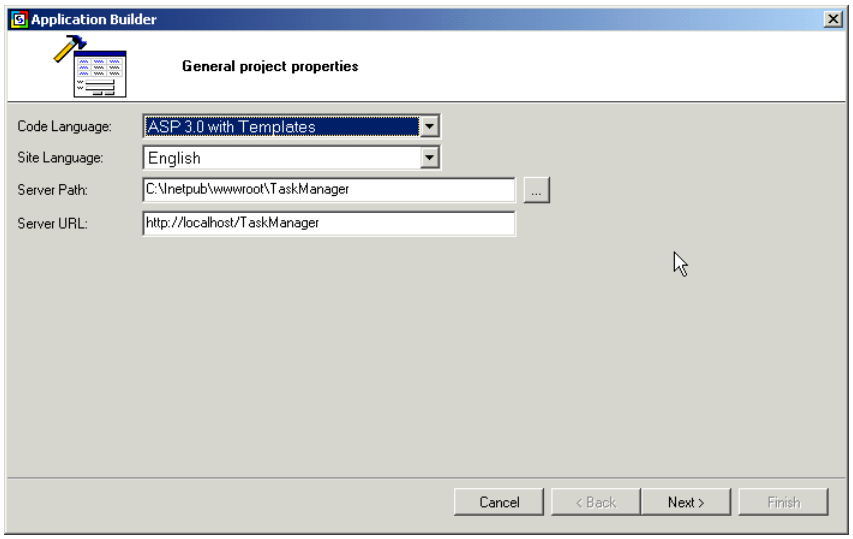
Specify Project Properties

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

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

- ***ASP 3.0 with Templates*** – generates ASP 3.0 files that uses separate .html files as templates during run-time.
- ***ASP.Net 1.0 C#*** - generates .aspx files with C# 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 1.2***. – generates JSP 1.1 code.
- ***PHP 4.0 with Templates*** – generates PHP 4.0 code (.php) and separate .html template files.
- ***Servlets 2.2 JDK 1.2 with Templates*** – generates Java code that utilizes .html templates.

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.

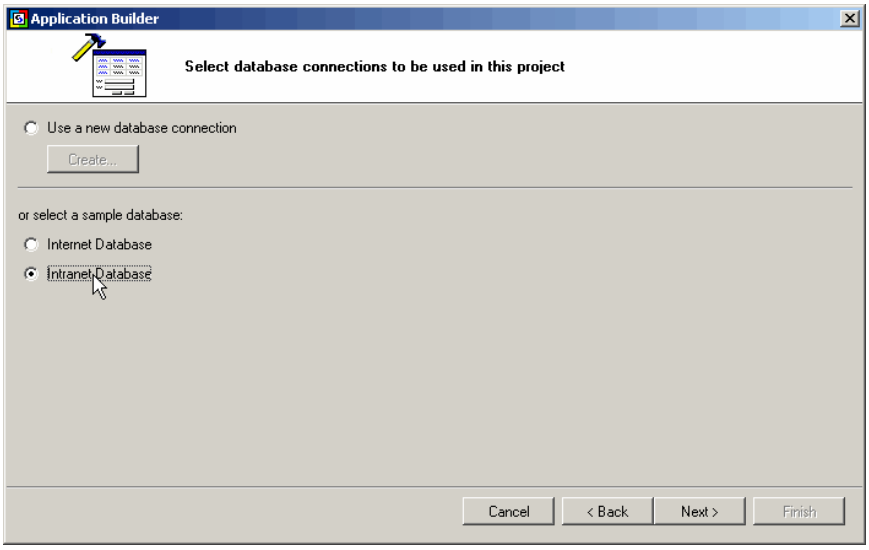


Server Path:
The full path where generated files should be published (locally). This path is usually preset by the Application Builder and can be left without changes.

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

Select Database Connection

Specify the database that you want to connect to. You can create a new database connection by selecting “Use a new database connection” and following the steps described in chapter 2: Create Database section.
For the purpose of this tutorial select “Intranet Database”, which is one of the two sample databases included with CodeCharge Studio and contains tables such as *tasks*, *priorities* and *employees*.



Configure the Application Builder

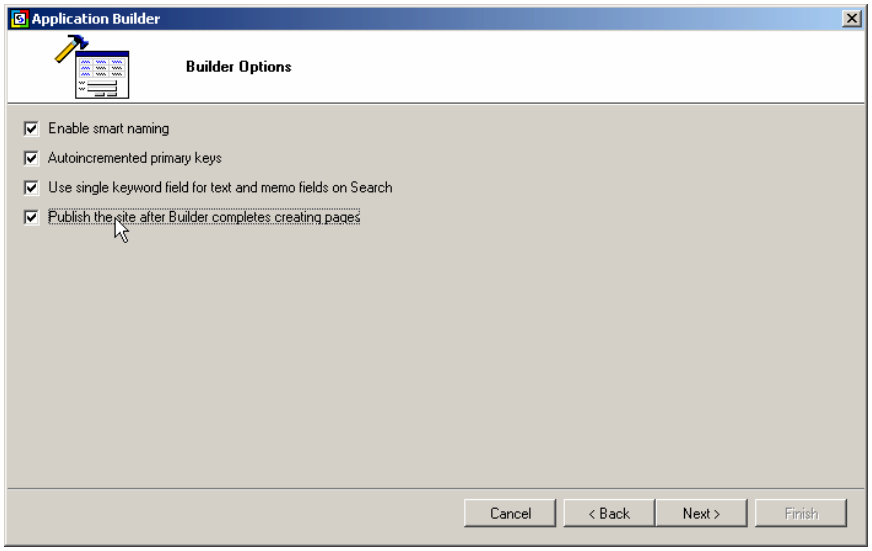
Configure the Application Builder by specifying configuration options as follows:

- **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.

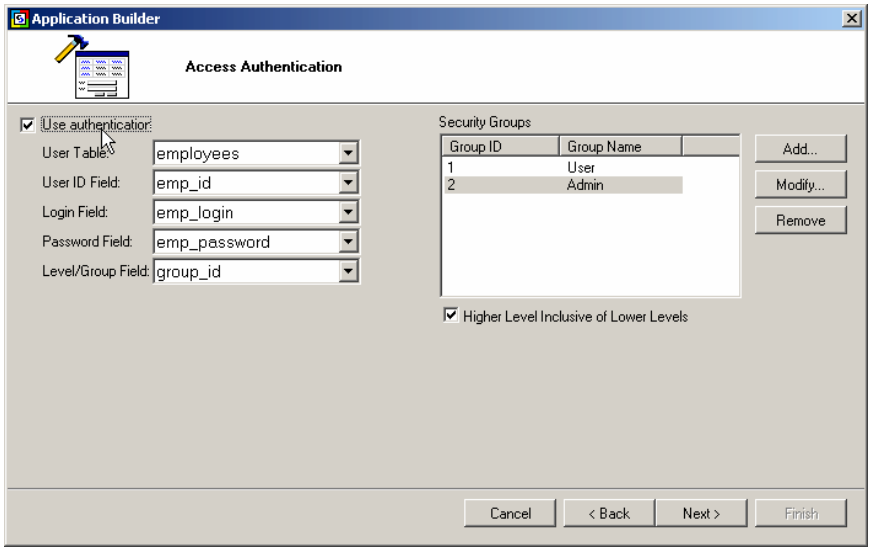
- **Use single keyword field for text and memo fields on search forms**
  Select this option to generate a single search field that searches against all the fields on the tables/grids. If this option is deselected, Application Builder creates a search section with multiple search fields – one search 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.



Setup Site Security and Authentication

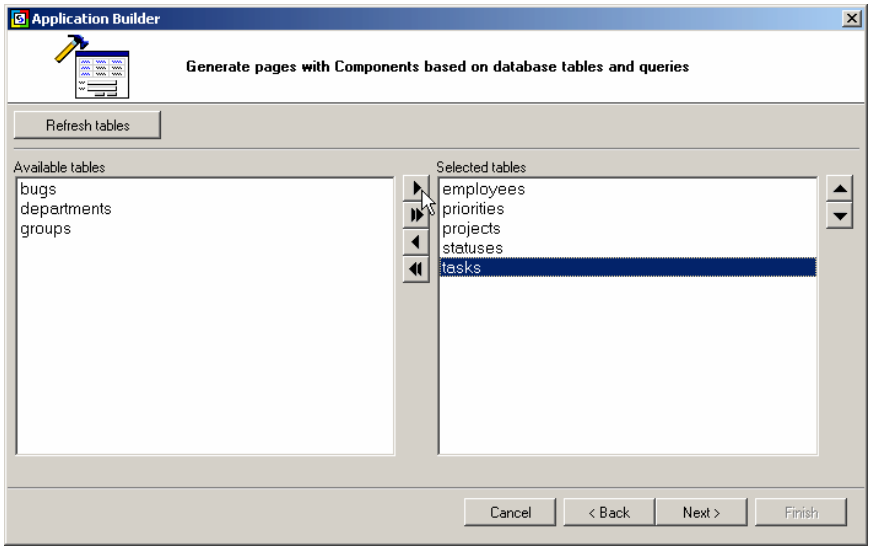
In this step, you can specify if you want to use authentication and check users' access privileges before allowing them to access certain pages.
Select "Use authentication" and leave all default options. Application Builder will then create the Login page and will allow you to specify a security level for each of the pages created.
You can also add additional security groups/levels or specify a different user table for authenticating users, as described in chapter 2: Setup Security Settings for the Project.



Select Database Tables

Select the following database tables to be converted to a web application:

employees
priorities
projects
statuses
tasks



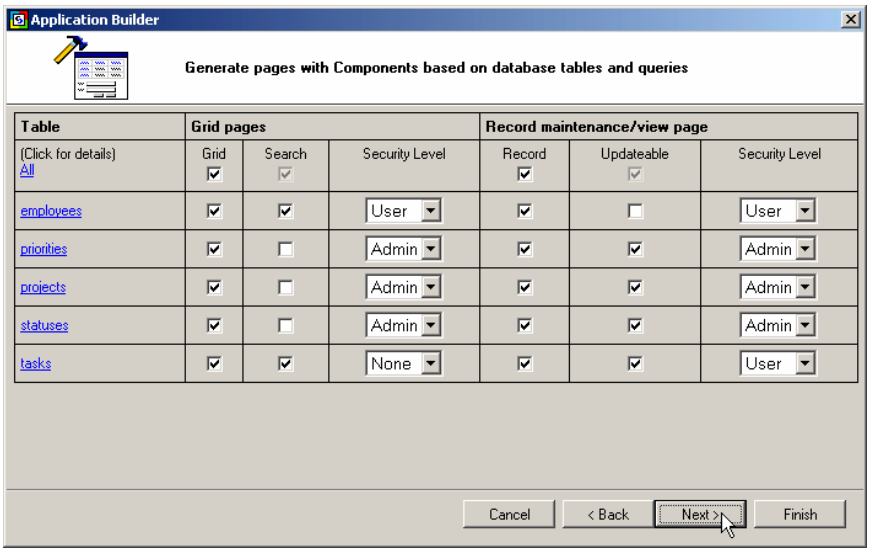
Configure Site Pages

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

Search and Grid page
Record Maintenance page

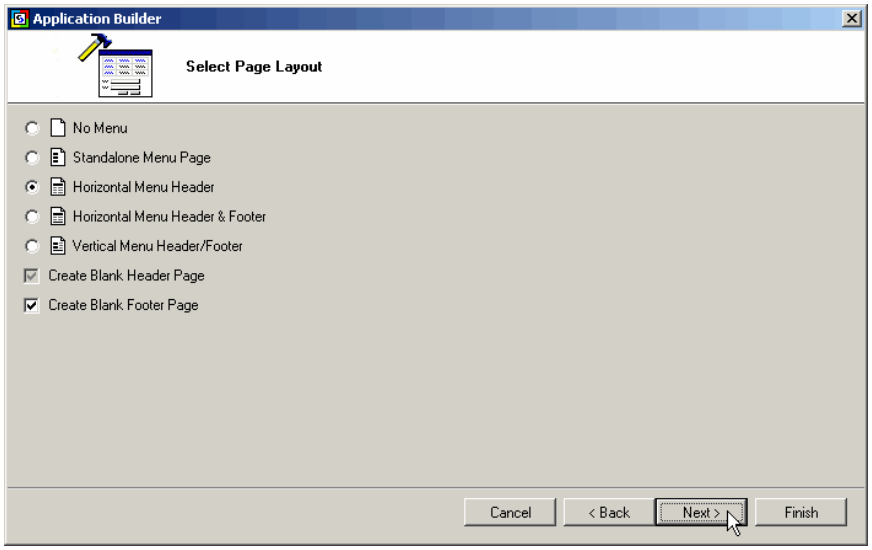
By clicking on a name of any of the tables, it is also possible to specify that the Search, Grid and Record forms should all be on the same page for each of the tables.
For this tutorial, configure pages as shown below. Application Builder will then convert the tables to web pages as follows:

- *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



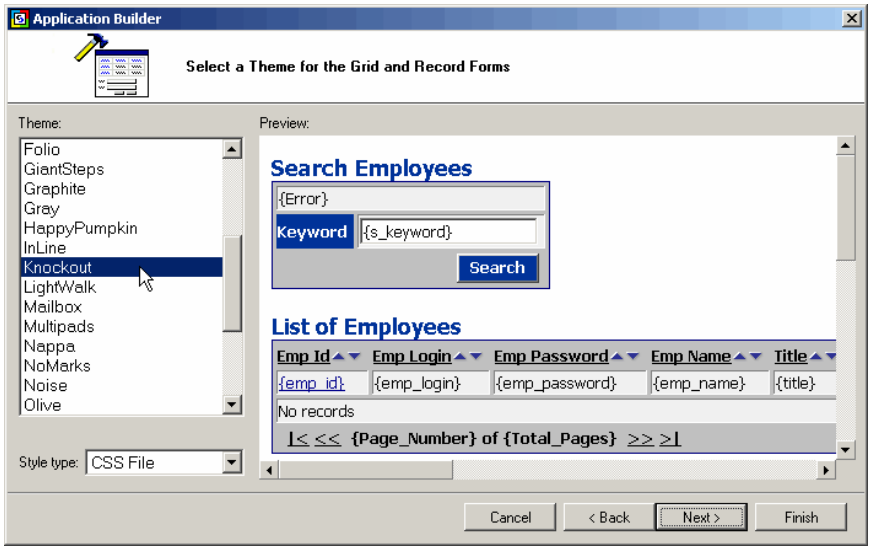
Specify Site Layout and Menu

The Application Builder automatically creates a header page with a menu, which is then placed in all other pages for easy navigation. Click “Next” to leave the default horizontal position of the menu for all pages.



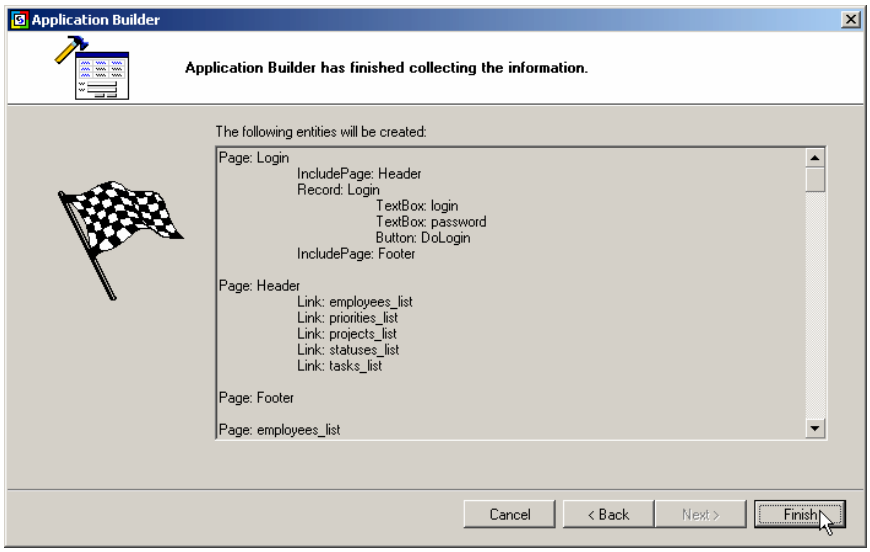
Select Site Theme

Finally, select the “Knockout” theme to apply to the site. The Application Builder will then utilize the theme to apply specific fonts and colors to each page.



Review Pages and Create the Site

Click [Finish] to let the Application Builder create all pages and publish the site.

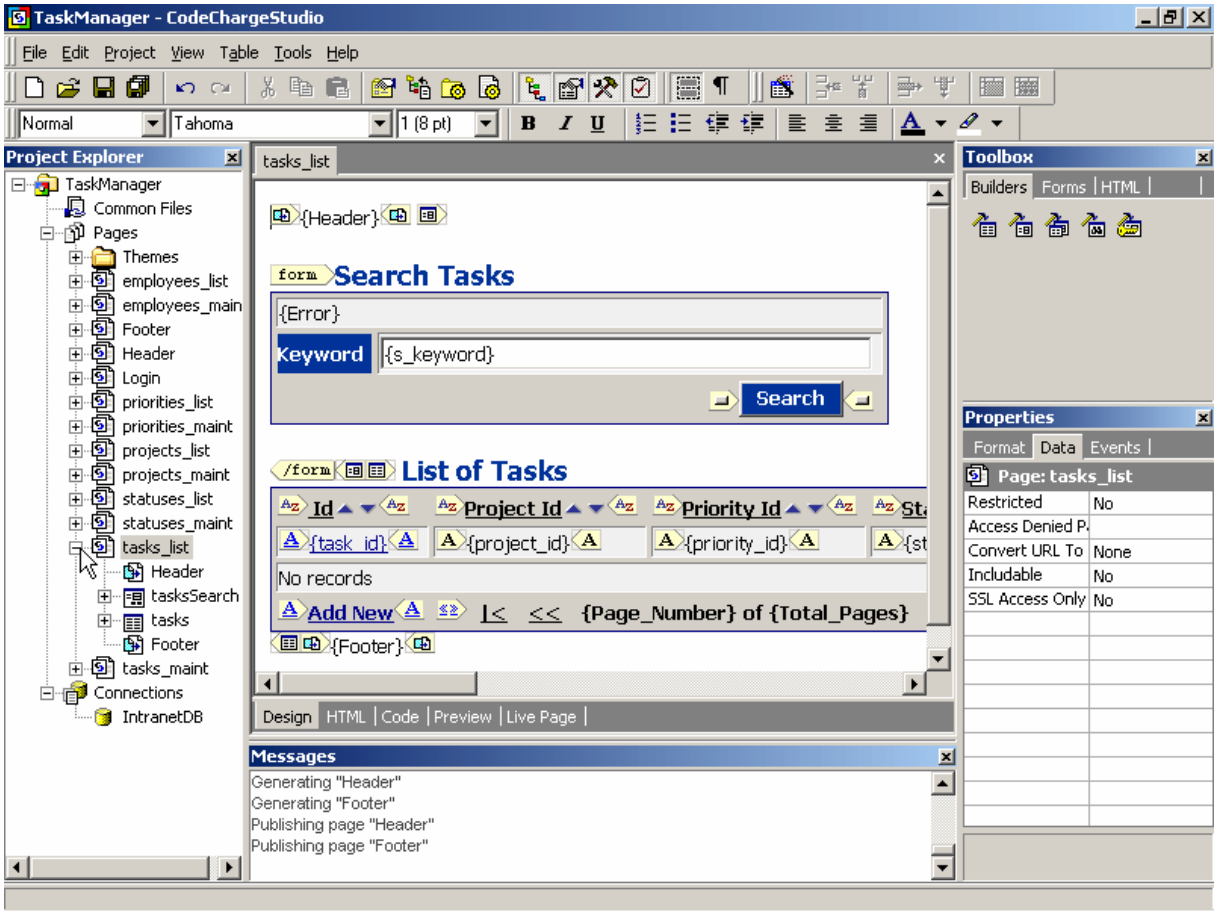


Finalizing the Task List Page

Although the Application Builder can create simple web applications, usually you will need to modify and extend the functionality created by it. For example, you may want to change text and captions, remove some of the grid columns, add additional search fields and listboxes, etc. In this section, you will learn how to do just that.

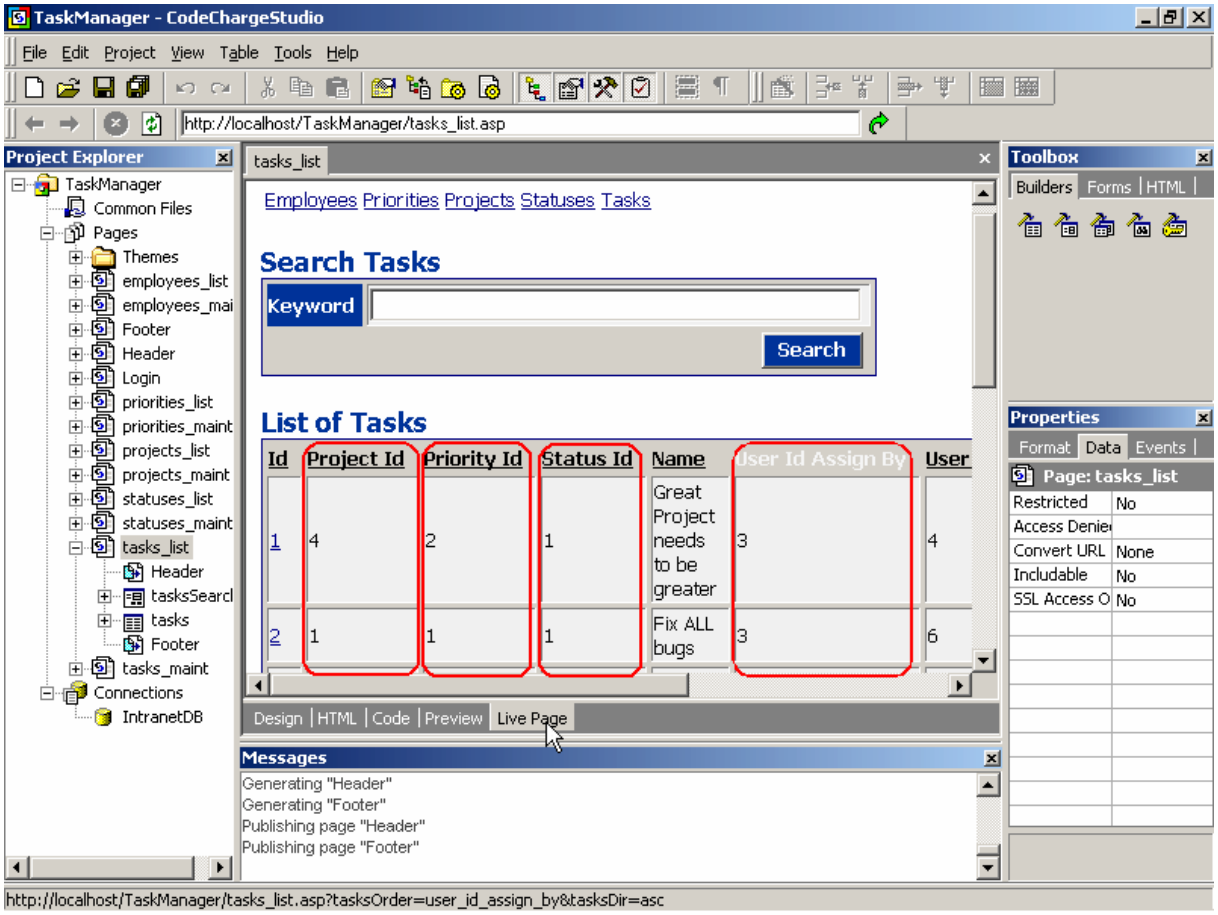
Open the Task List Page

Click on the "+" sign next to the tasks_list page name to open it for modification.



Test the Page

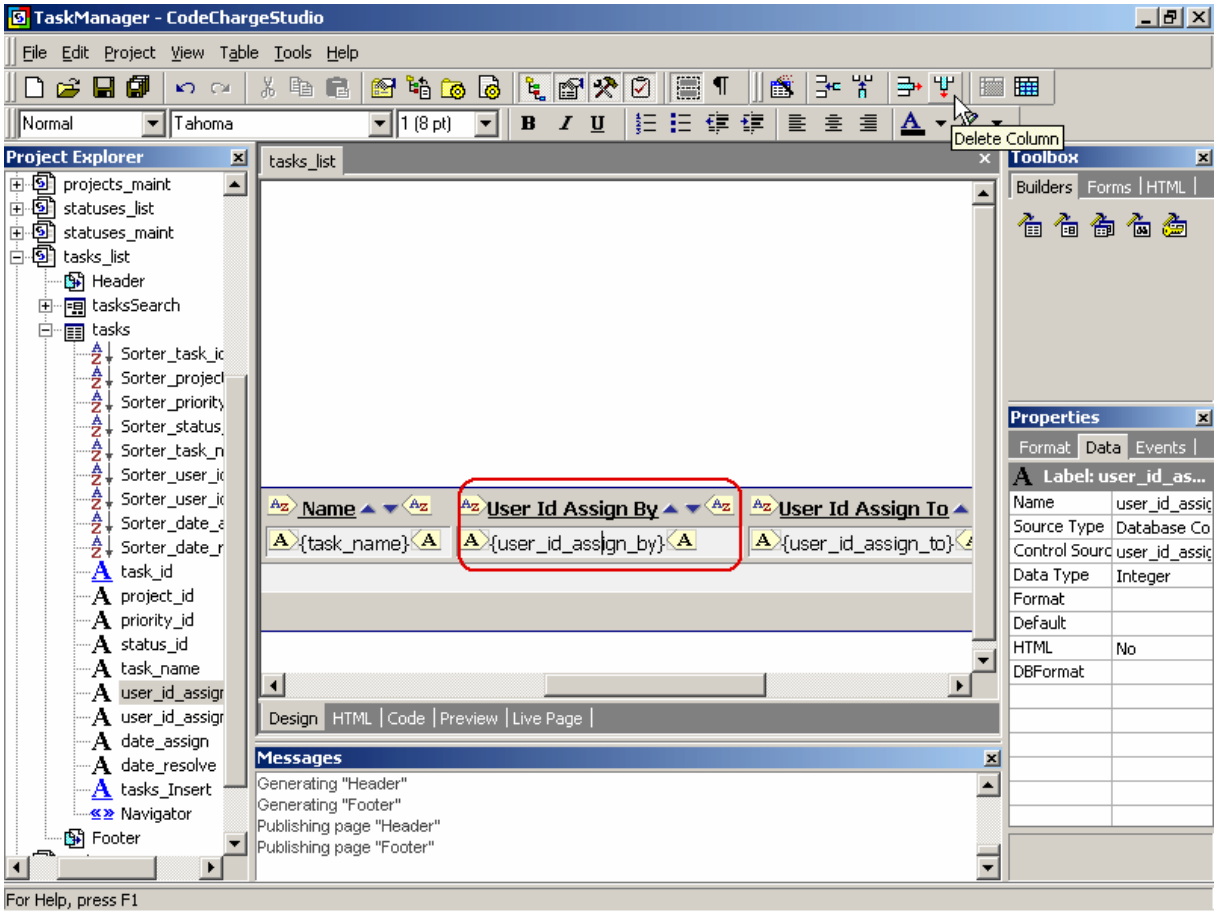
Click on “Live Page” tab to view and test the working page.
Notice that some of the grid columns contain numeric IDs of entities such as projects, priorities and statuses while other columns such as “User Id Assign By” simply may not be needed.



Delete Unneeded Columns

Revert to “Design” mode and select the unneeded column by clicking and positioning the cursor somewhere within the column. Next, click on the “Delete Column” icon in the toolbar to remove the column.
Use this method to delete the following three columns:

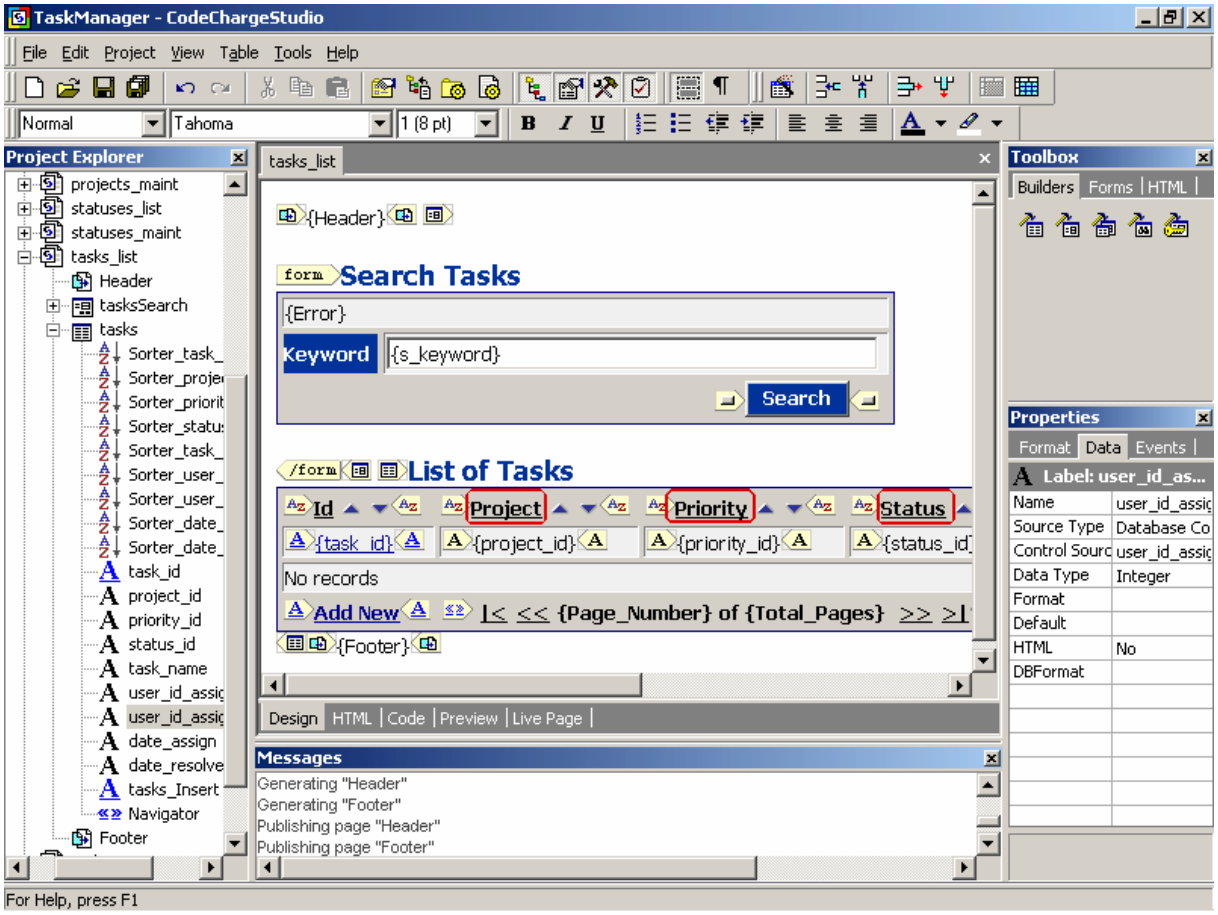
User Id Assign By
Date Assign
Date Resolve



Change Text and Captions

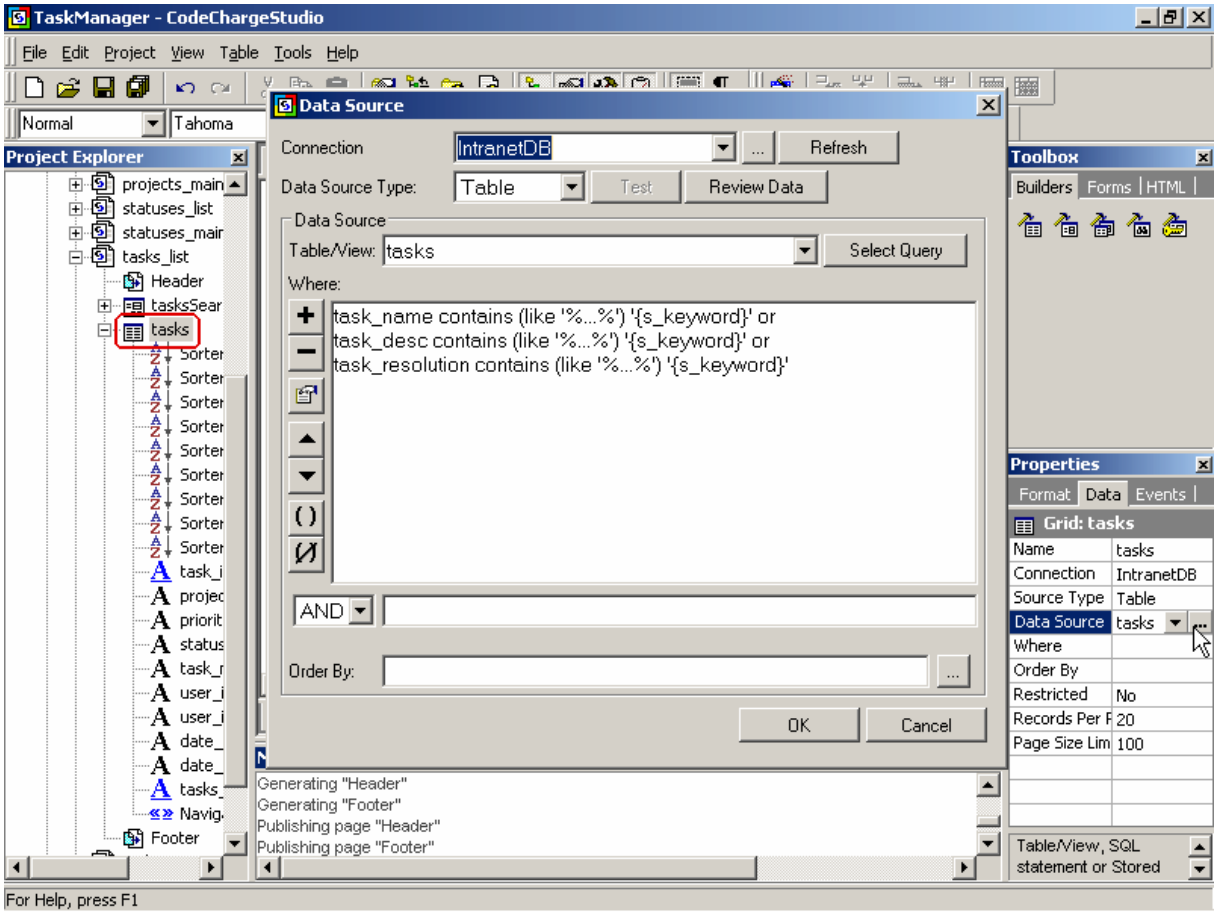
Use the design editor to modify some of the text appearing on the page.
Change:

Project Id to ***Project***
Priority Id to ***Priority***
Status Id to ***Status***
User Id Assign To to ***Assigned To***



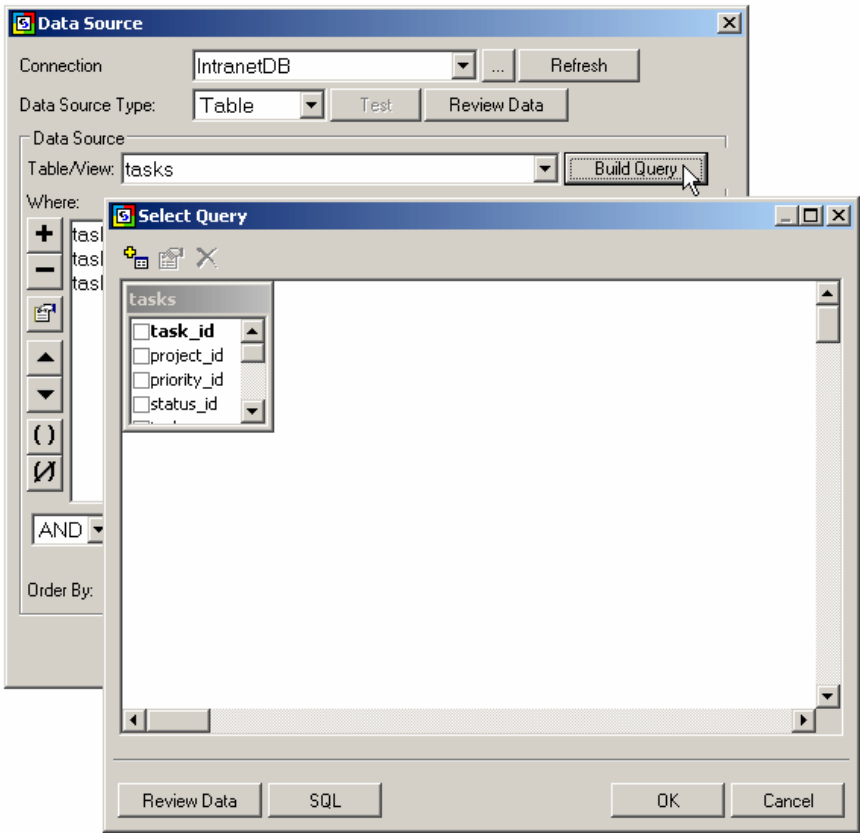
Launch Data Source Editor

Now use the Data Source editor to specify additional tables to be used in the grid. First, select the “tasks” grid in the Project Explorer then click on the […] button next to the Data Source property to open the Data Source window.
Here you can specify additional tables to use in the grid, such as the priorities table containing priority names and the statuses table that contains status names.



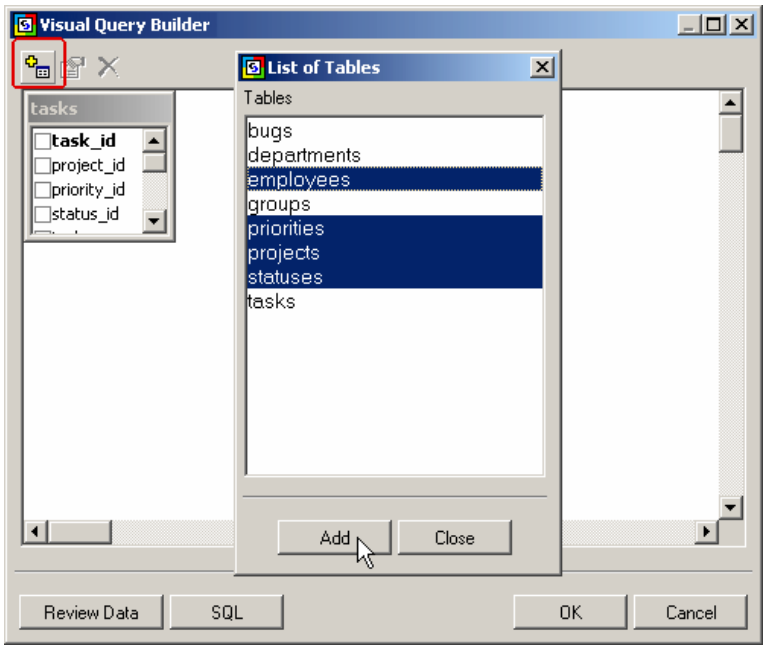
Open the Visual Query Builder

Now open the Visual Query Builder by clicking on the “Build Query” button.
A new window will open up that shows the *tasks* table that is currently used in the Grid.



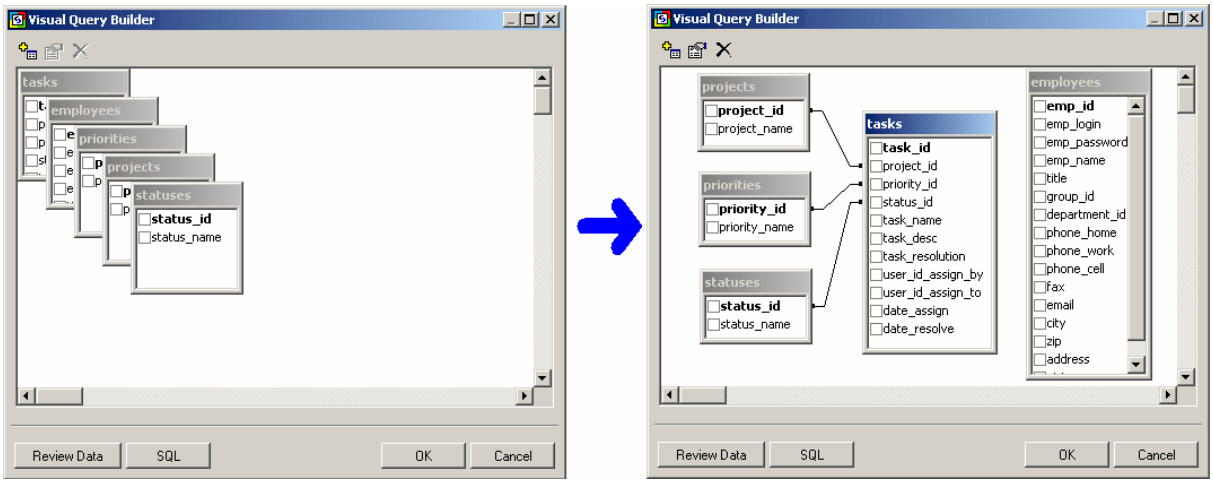
Select Additional Tables

Select additional tables that are related to the tasks table. Hold down the *Ctrl* key when clicking on table names to select multiple tables at once. Click [Add] to add the tables to the Select Query window.



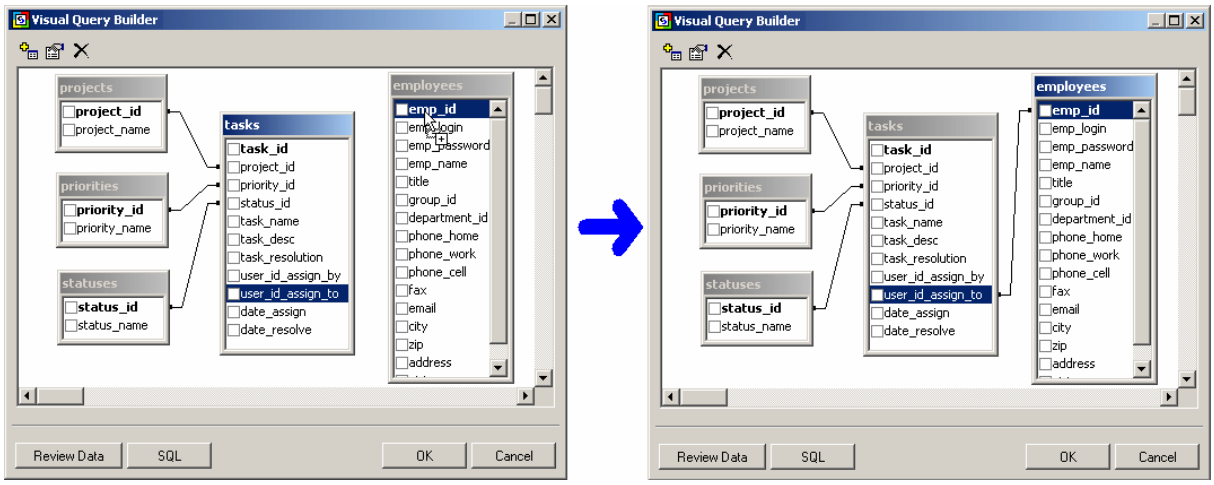
Arrange Tables in the Visual Query Builder

Once multiple tables are in the Visual Query Builder, drag the tables and arrange them on the screen to see all the information. You will notice that some of the tables are connected. This is because the Query Builder recognizes relations between tables that have fields with the same names. In this case both *tasks* and *statuses* tables have the same field name: *status_id*, which makes it logical to assume that they are related.



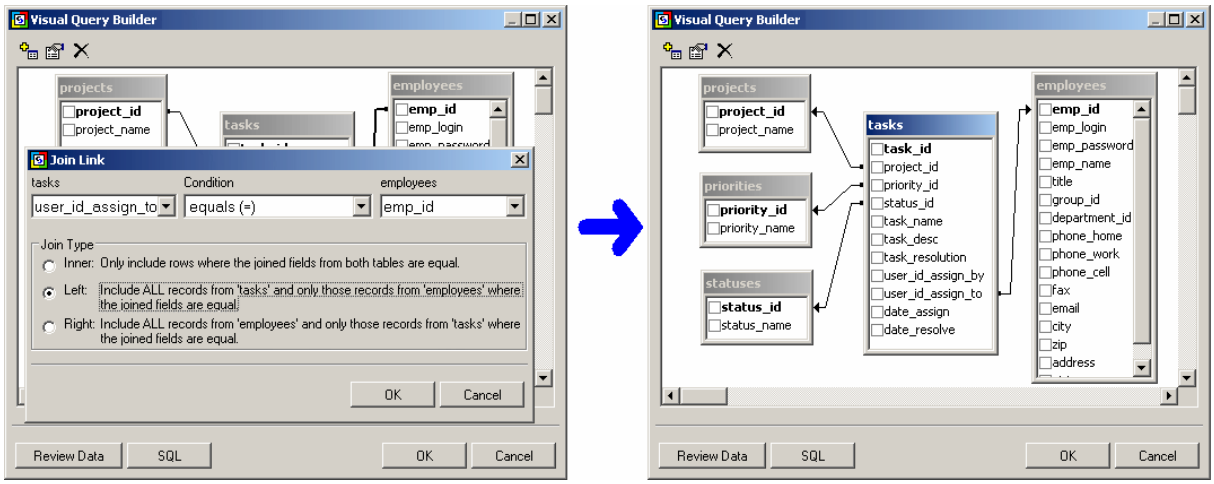
Define Table Relations

Now define relations between the remaining tables, in this case *employees* and *tasks*, by dragging the foreign key field *user_id_assign_to* from the *tasks* table to the *emp_id* field in the *employees* table. All the tables should then be connected to the tasks table.



Define Field Joins

Although not necessary in our case, you can also define joins between fields by double-clicking on each of the lines that connect the tables, then specifying that you want to display all records from *tasks* table and matching records from related tables. If you do not do this, the page may not display all records, for example if you have a task without any priority or status assigned to it.

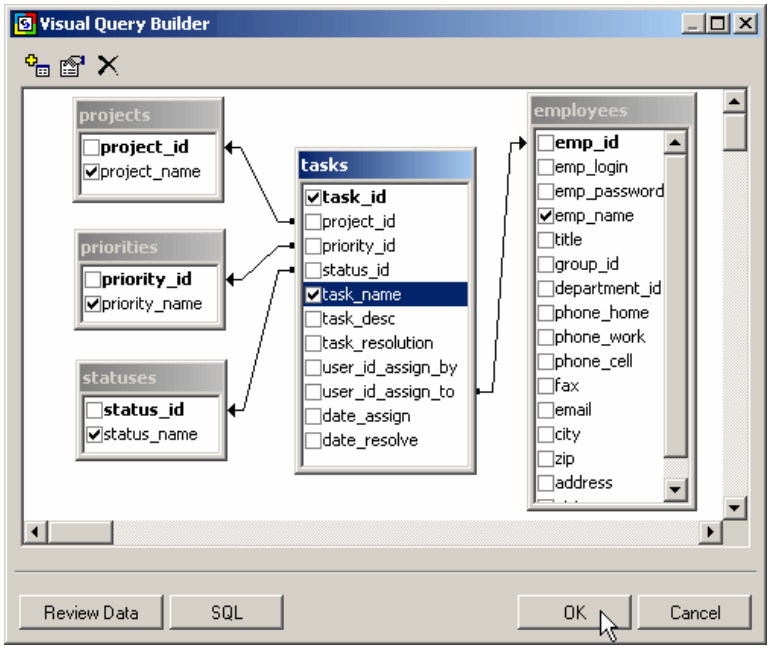


Set Fields for Inclusion in the Grid

Now mark the fields to be included in the grid by clicking on their corresponding checkboxes. Include the following fields:

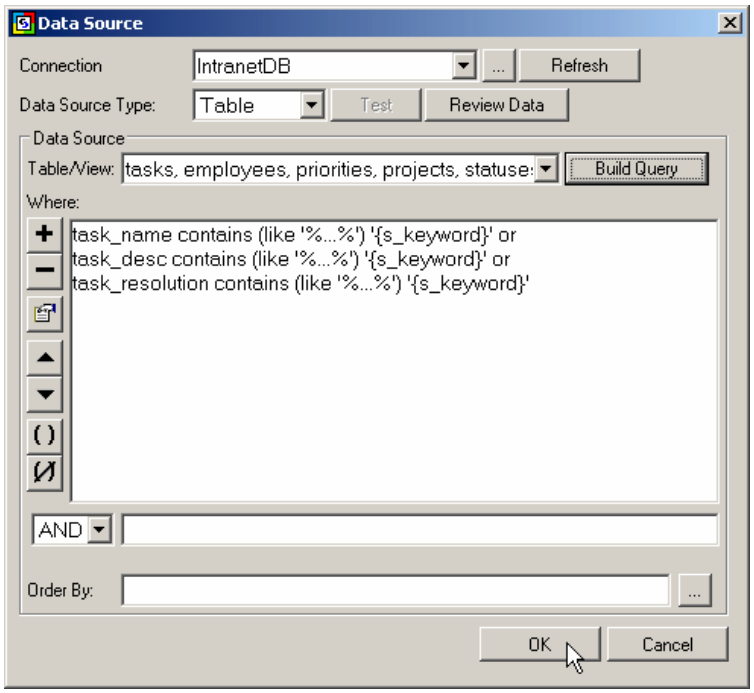
task_id
task_name
project_name
priority_name
status_name
emp_name

Click [OK] when done.



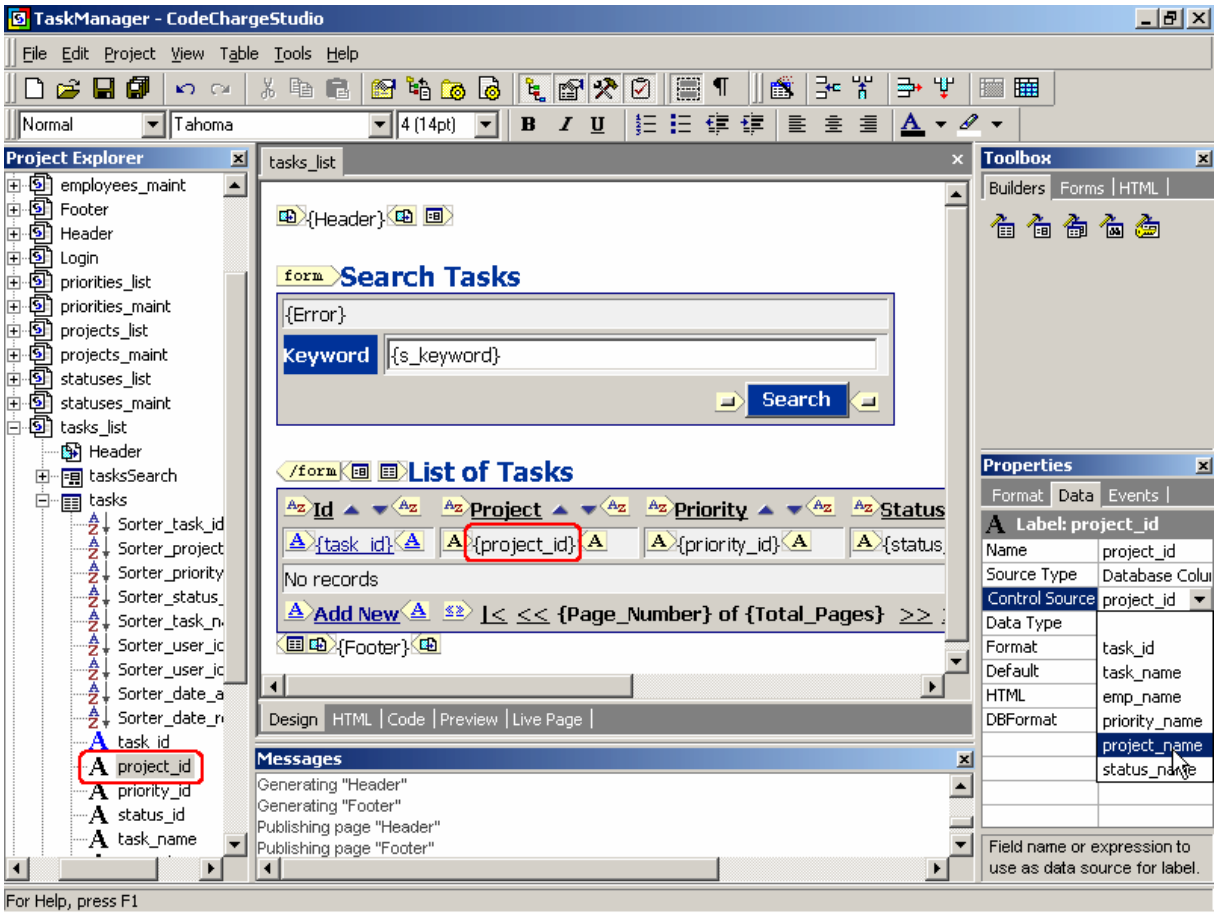
Return to the Grid

The Data Source window now lists several tables previously selected in the Visual Query Builder. Click OK again in the Data Source window to return to page design mode.

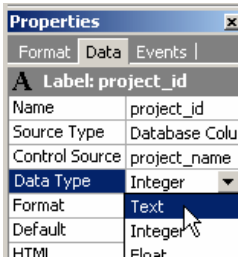


Update Control Sources

Select the *project_id* field by selecting it in the Project Explorer or by clicking on it within the page design area. Change the value of the “Control Source” property from *project_id* to *project_name*.



Also, change the value of the “Data Type” property from Integer to Text.
Now your page should display names of projects instead of their numeric IDs.

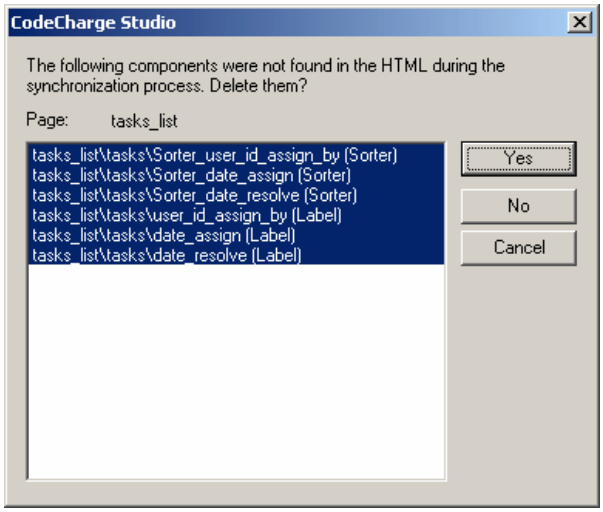


Repeat the above actions for the following fields:

priority_id – change to *priority_name*
status_id – change to *status_name*
user_id_assign_to – change to *emp_name*

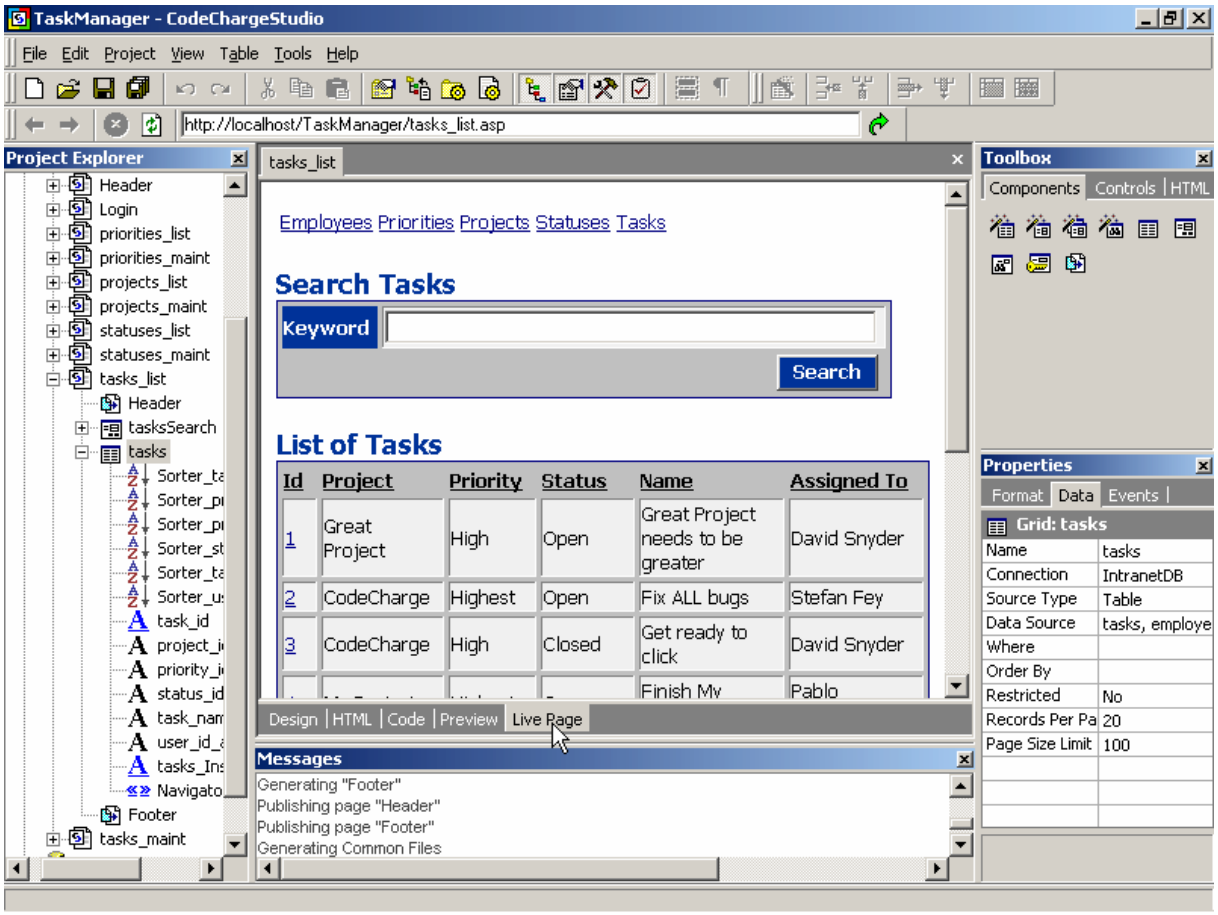
Synchronize HTML with the Project

Click “Live Page” to view and test the working page.
Sometimes you will see the message window shown below alerting you that some of the components or controls were not found in the HTML. This is because you previously removed some of the grid columns and CodeCharge Studio wants you to confirm that this is OK since those controls are still defined in the project.
Click “Yes” to confirm the removal of the columns and continue.



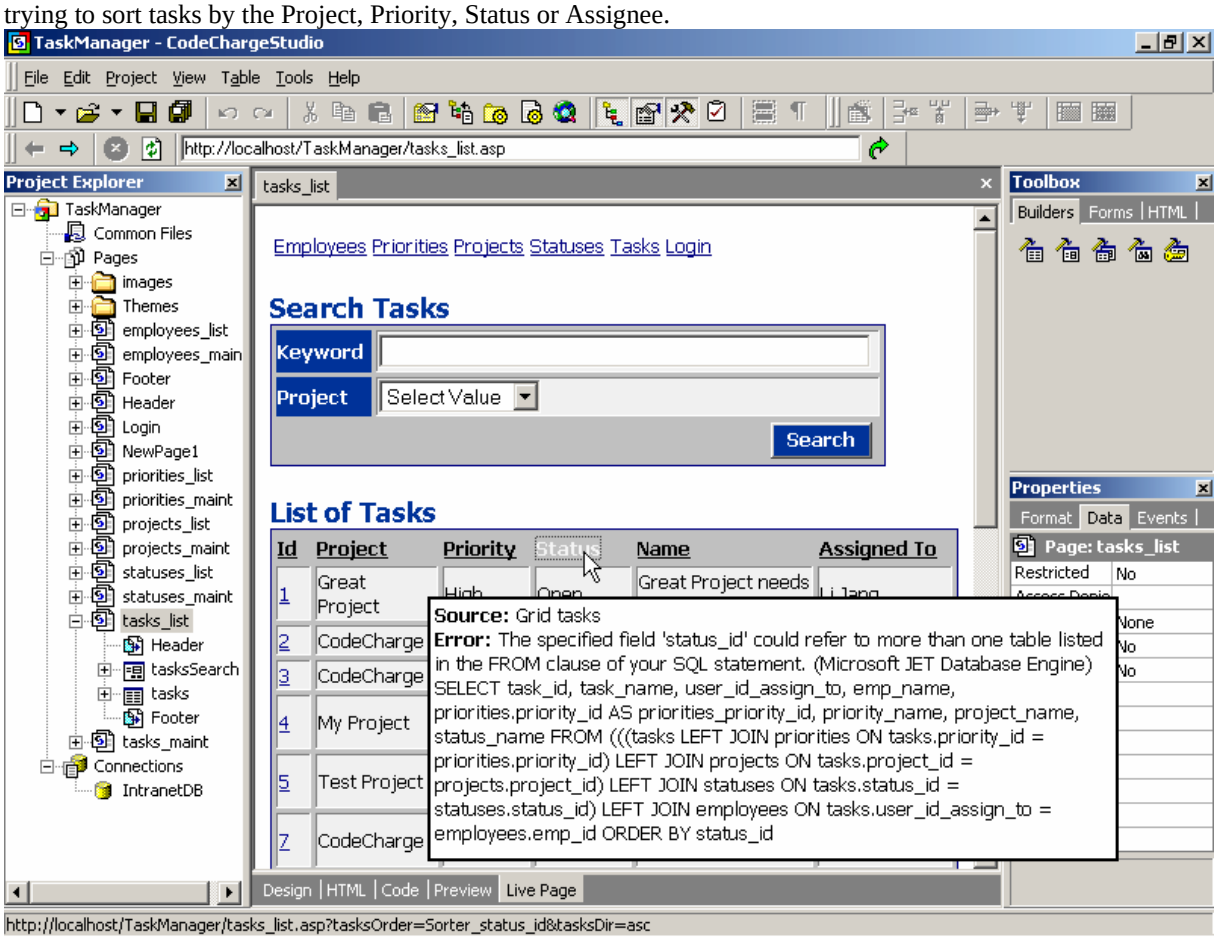
View and Test the Live Page

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.



Correct Sorting Errors

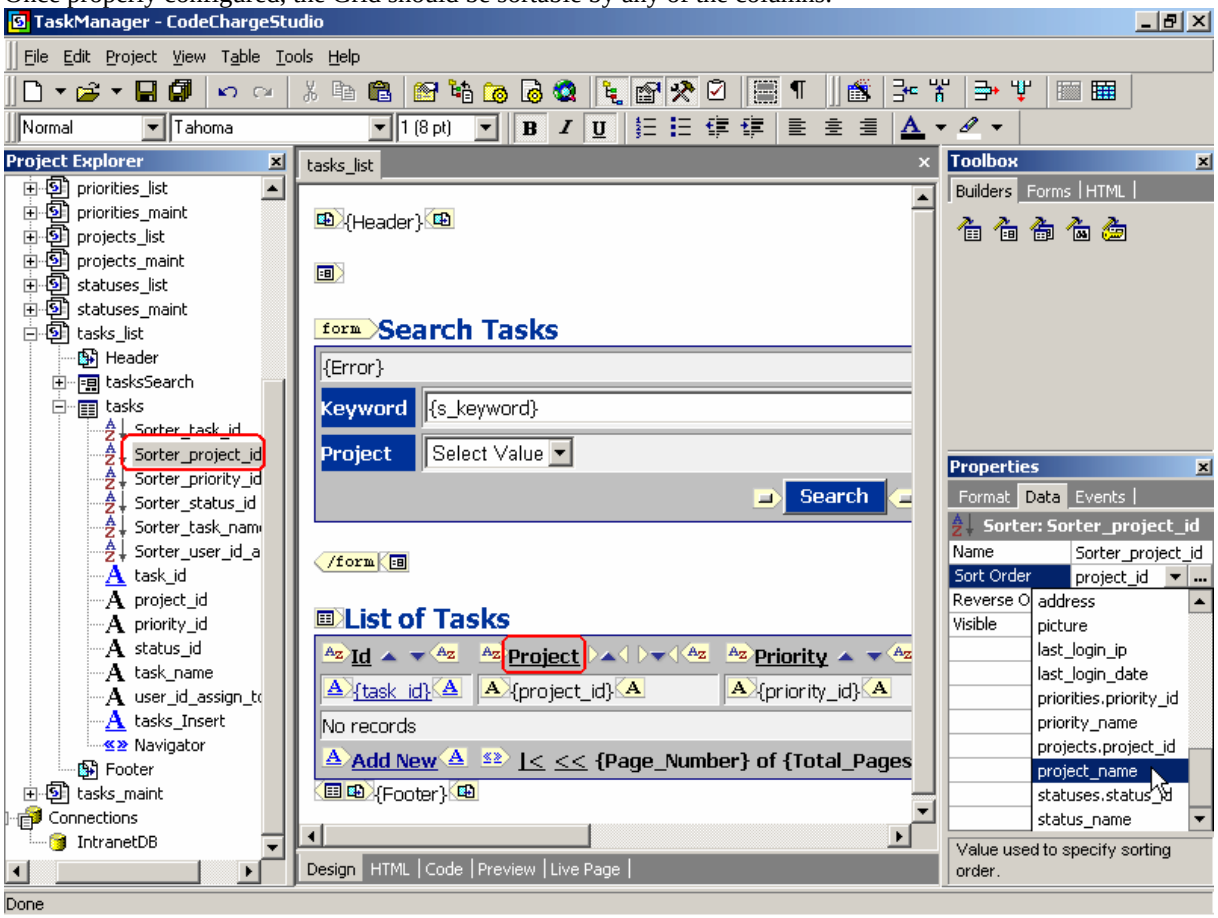
Although the page should generally work OK, you may see sort errors, similar to the one pictured below, when trying to sort tasks by the Project, Priority, Status or Assignee.



This happens because the Application Builder creates the Grid based on single database table, however adding additional tables to the query can cause a conflict when two or more tables have fields with the same name. In such case the program doesn't know which of the identical fields to perform the search on. In the above case, both *tasks* and *statuses* tables contain the *status_id* field.

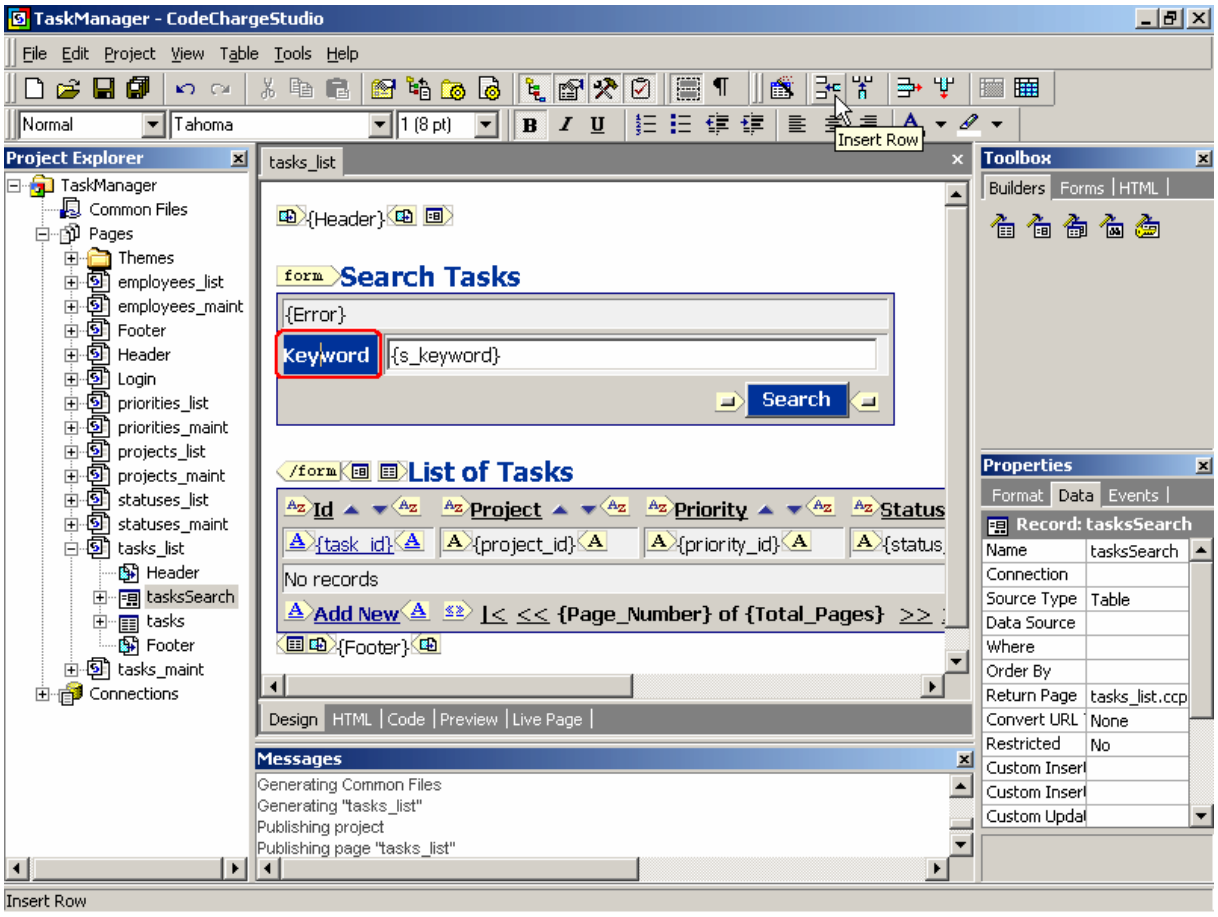
Configure Grid Sorters

Switch back to the “Design” view and click on the “Project” heading within the Grid, or on the “Sorter_project_id” object in Project Explorer. This will select your sorter, and you should see its properties in the Property browser window. Change the Sort Order from "project_id" to "project_name".
Repeat the same steps for the remaining sorters on the page, especially: Priority, Status and Assigned To.
Once properly configured, the Grid should be sortable by any of the columns.



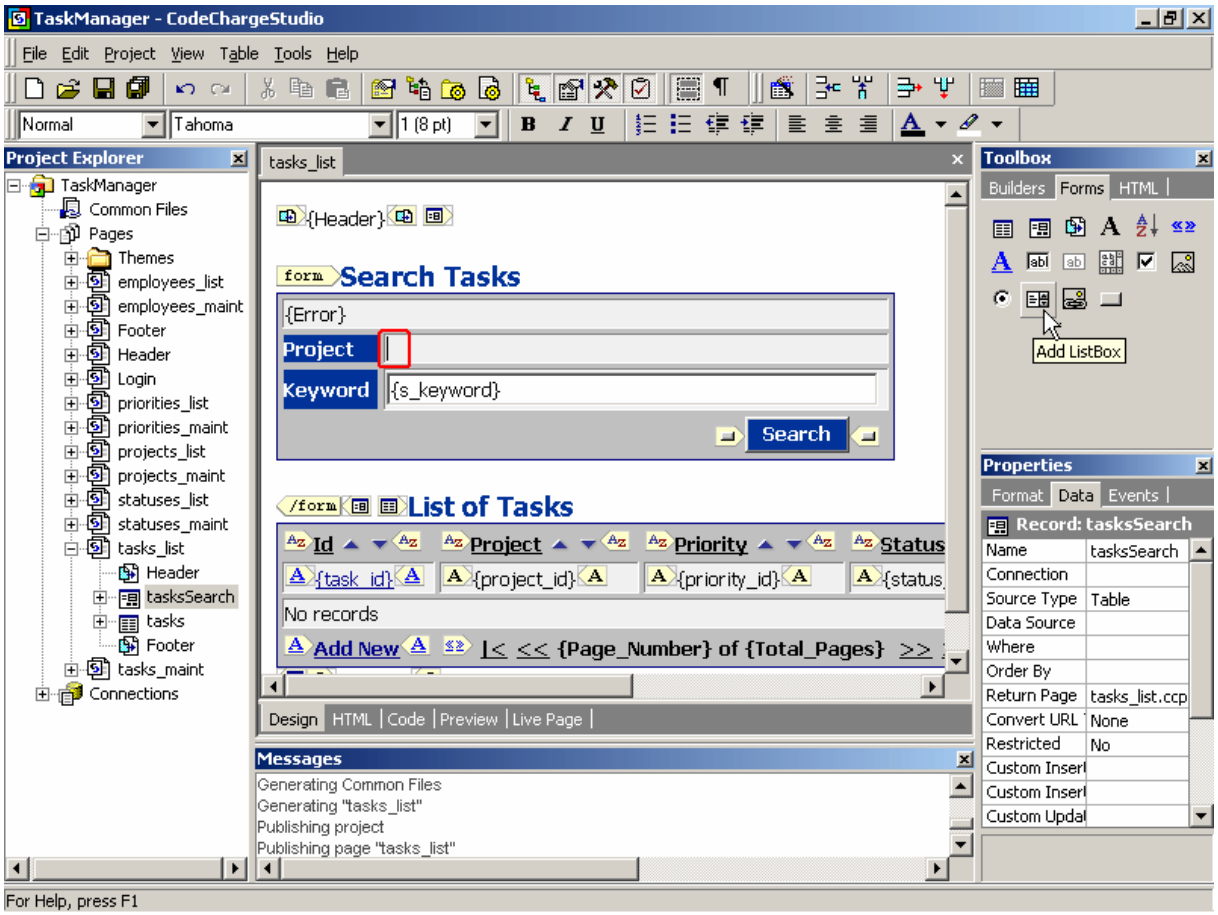
Add ListBox Search – Create New Table Row

Now let's add an additional search option to the grid – a ListBox with project names so that users can view tasks for selected project.
Position the cursor somewhere within the "Keyword" text by clicking on it, then select the "Insert Row" icon to add a new table row at the top of the search section.



Add ListBox Search – Insert ListBox Control

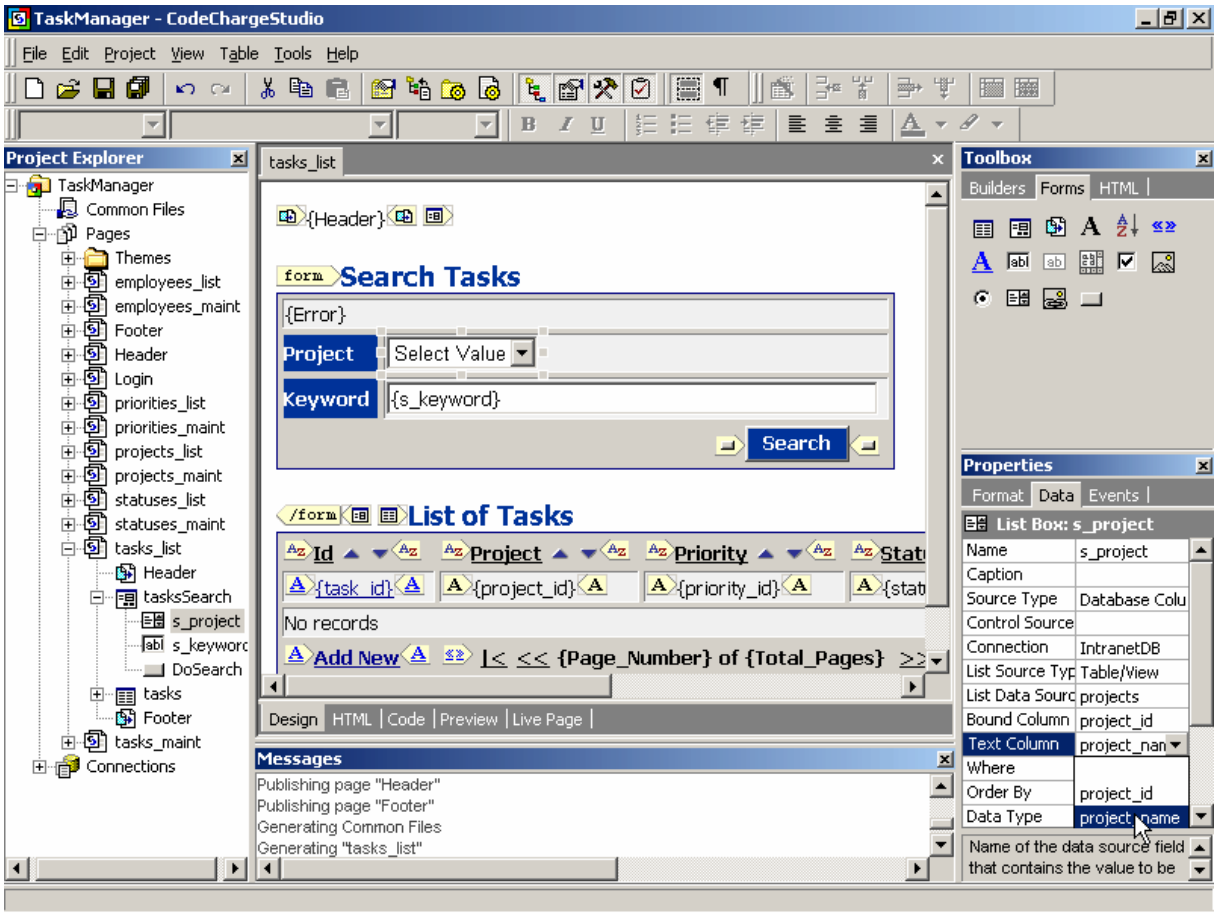
Type the text “Project” within the newly created left table cell, then position the cursor in the right cell as shown, and then click on the “Add ListBox” icon in the Toolbox to add it to the page.



Add ListBox Search – Set ListBox Properties

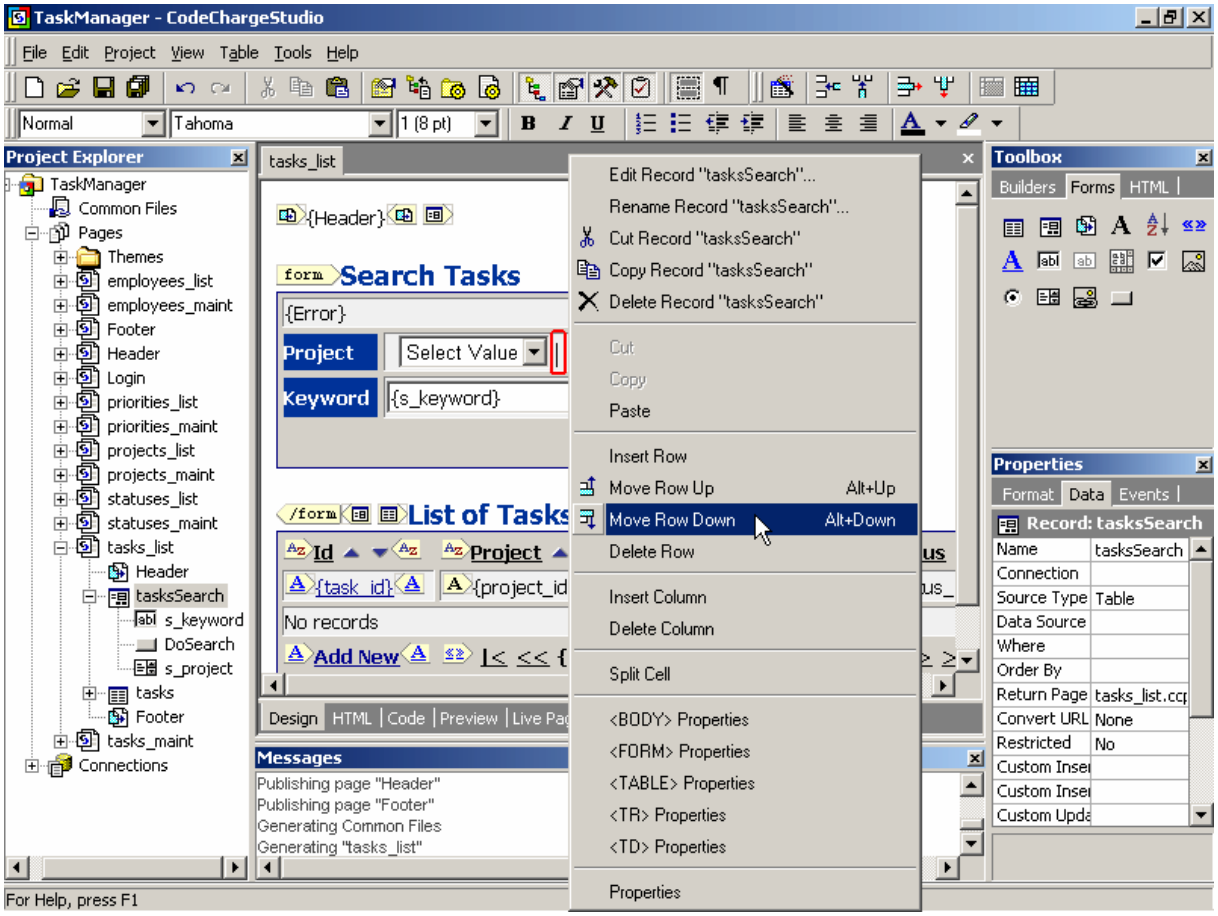
Configure the ListBox properties by clicking on it and specifying required values in the property editor as follows:

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
List 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)



Add ListBox Search – Move Table Row

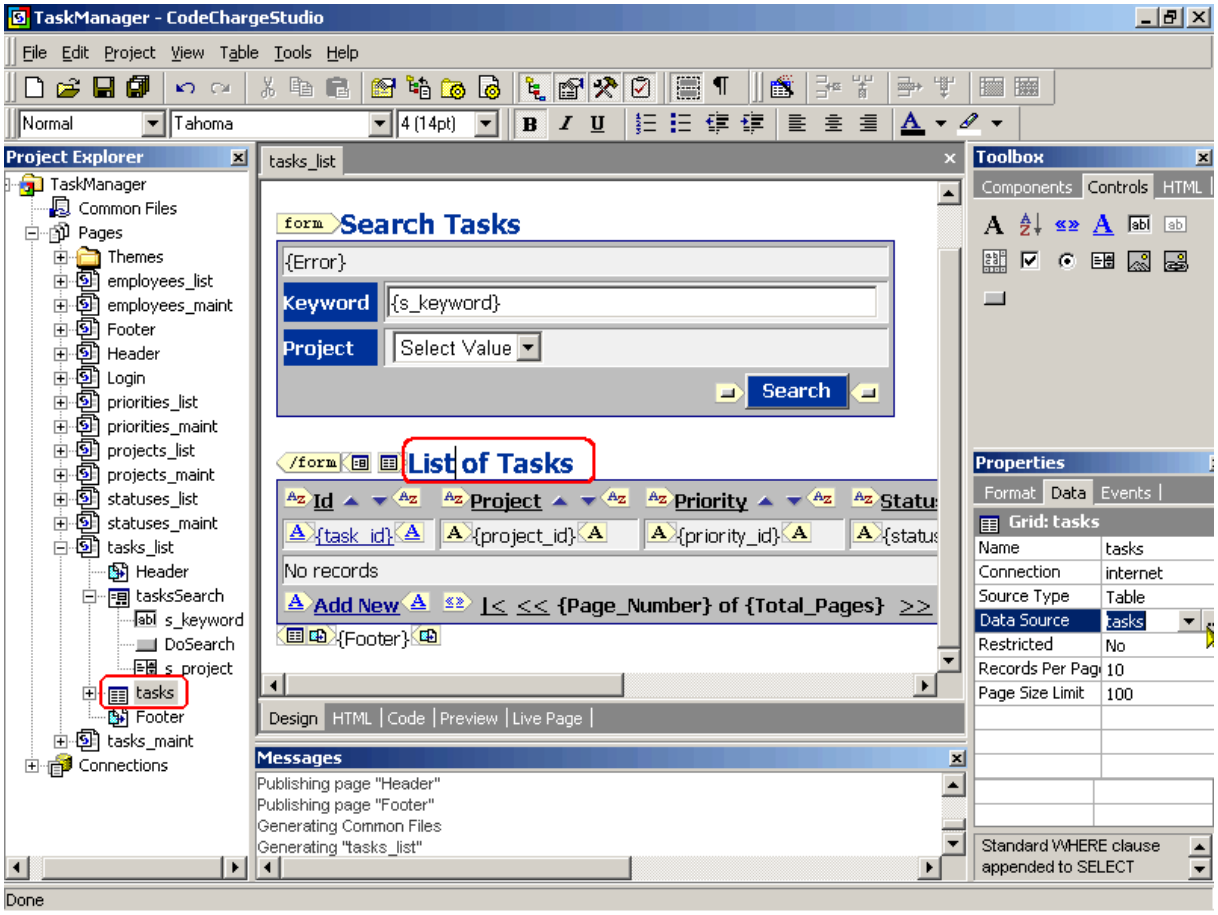
Finally, move down the table row containing the ListBox by right clicking near the ListBox and selecting “Move Row Down”.
You can also do this by positioning the cursor next to the ListBox and using the *Alt + Down Arrow* keyboard keys.



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.

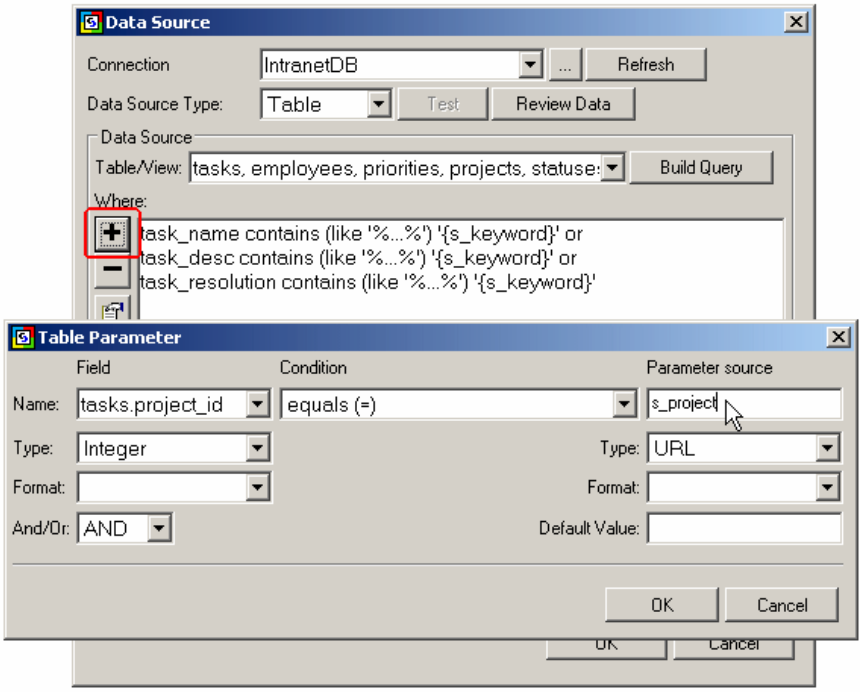
Select the grid by clicking anywhere within the grid’s caption on the page or by selecting it in the Project Explorer. Then click on the […] button of the “Data Source” property.



Filter Grid Records – Add Search Parameter

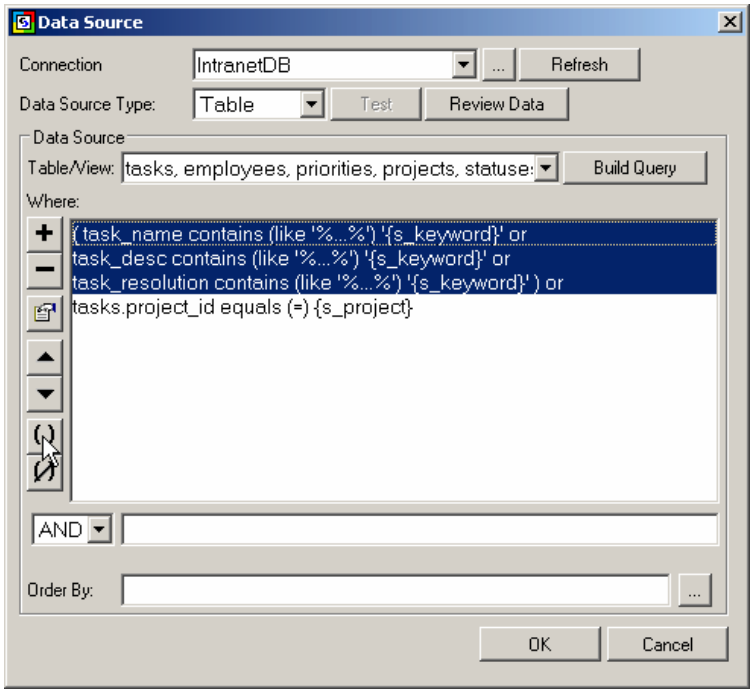
Add a new search parameter to the grid by clicking on the [+] button and then specifying the *tasks.project_id* field to be matched against the *s_project* parameter, which is the name of the previously created ListBox. This will cause the grid to receive the parameter via the URL then show only matching results.

Click OK when finished entering the information.



Filter Grid Records – Group "Where" Parameters

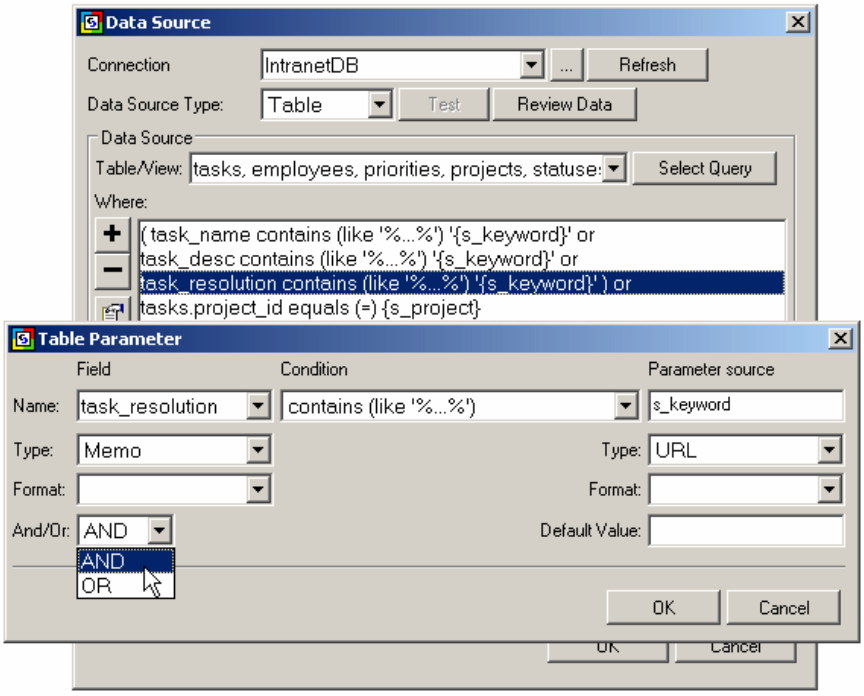
Back in the Data Source window, you will see four parameters, three previously configured by the Application Builder for the search keyword, and one you just added for the project listbox. Now the grid will return results if any of the criteria are met, however we want results to be returned only if both the keyword and listbox parameters are matched. For example if a user enters keyword "bug" and selects project "CodeCharge", the grid shouldn't return records that are within the selected project **or** that contain word "bug". Instead, we'd want the grid to show results that are within the project "CodeCharge" **and** contain the word "bug".
To configure the grid in such a way, first group together all parameters matched against "s_keyword". Select the first three parameters by dragging the mouse over them or by holding down the Ctrl key and clicking on each parameter. Then click the [( )] button to add parentheses around the selected parameters, which will group them together.



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:

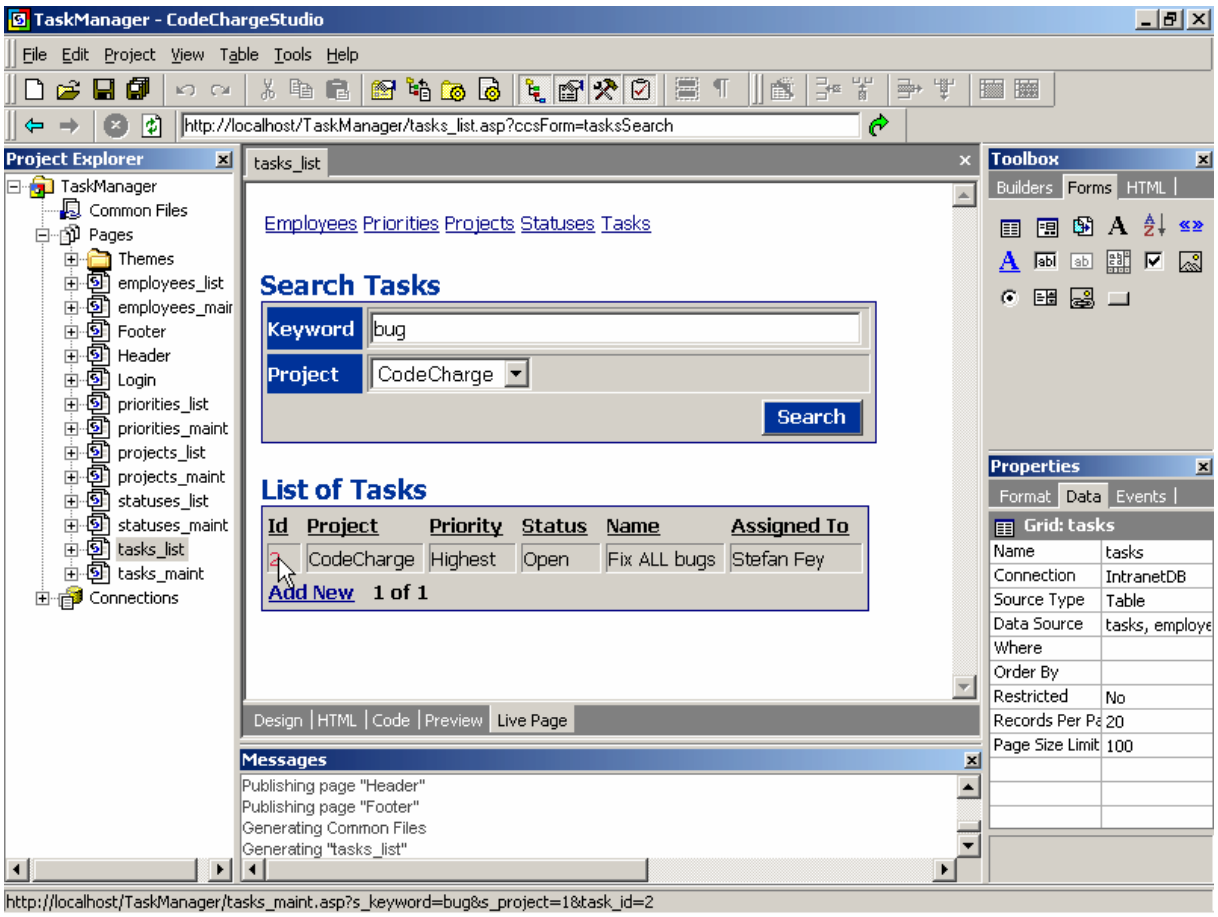
(task_name contains (like '%…%') '{s_keyword}' or
task_desc contains (like '%…%') '{s_keyword}' or
task_resolution contains (like '%…%') '{s_keyword}') ***and***
tasks.project_id equals (=) {s_project}



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.

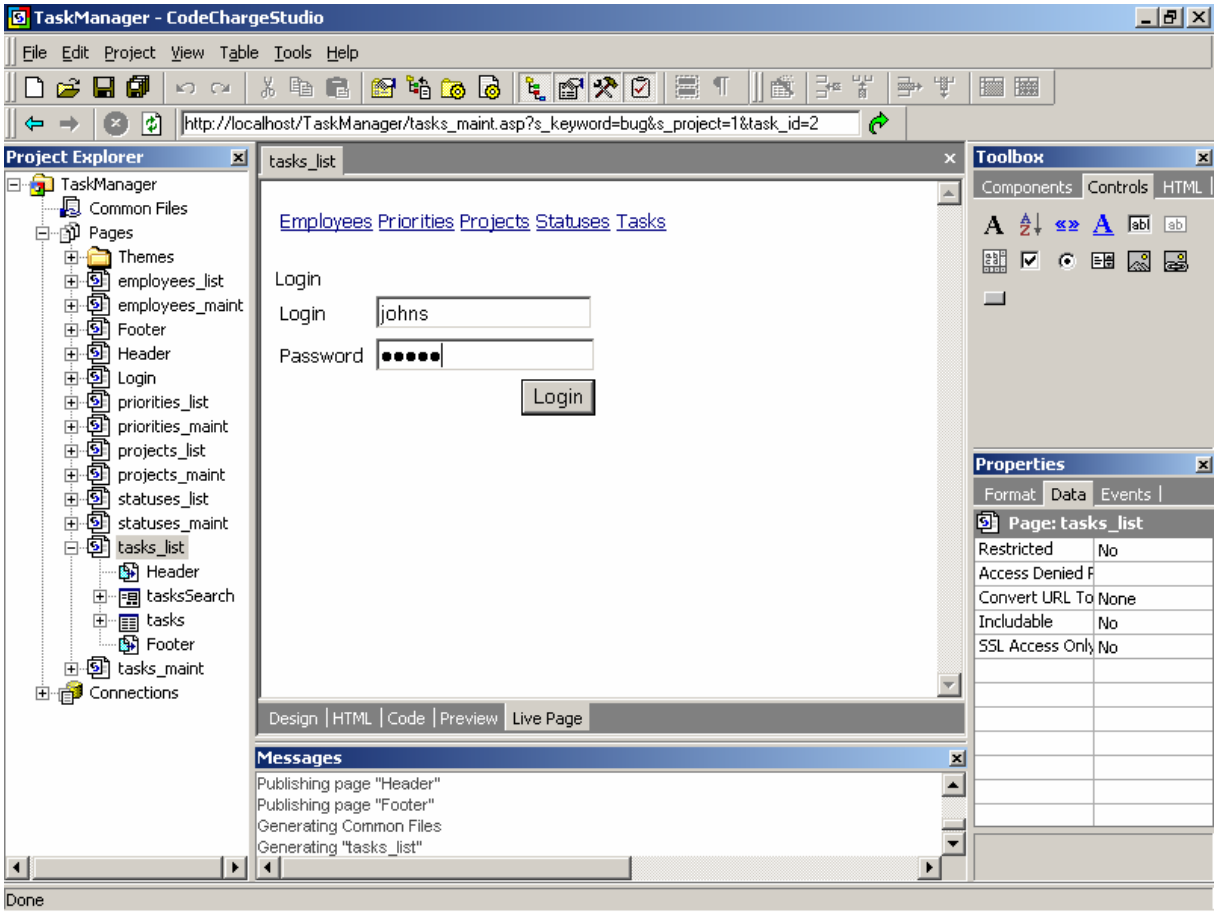
When finished viewing the page, click on a Task Id for any of the tasks to test the Task Maintenance page.



Login to the System

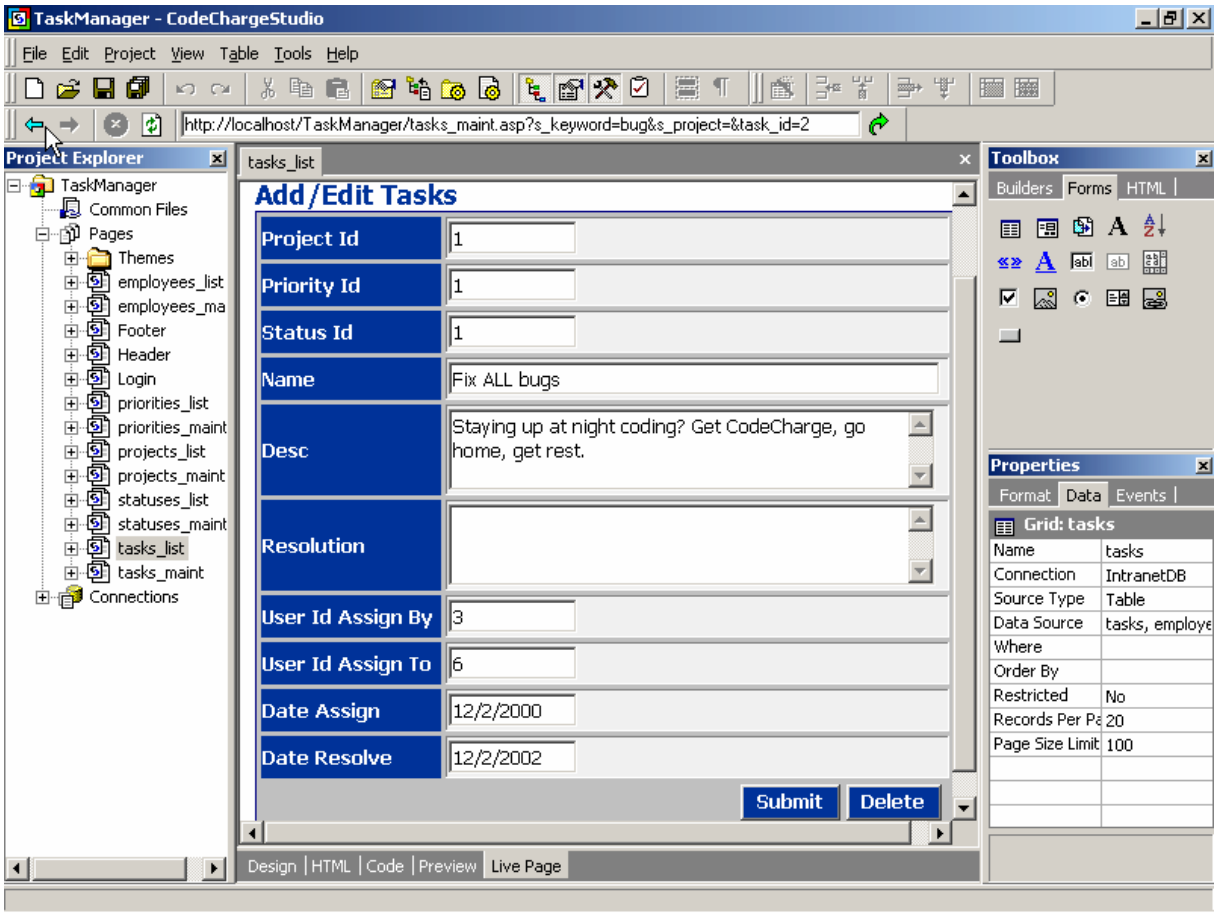
When you click on any of the tasks ids on the task list page, you will be directed to the Login page where you can enter your login and password.
Enter *johns* / *johns* to login as John Smith.

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



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.



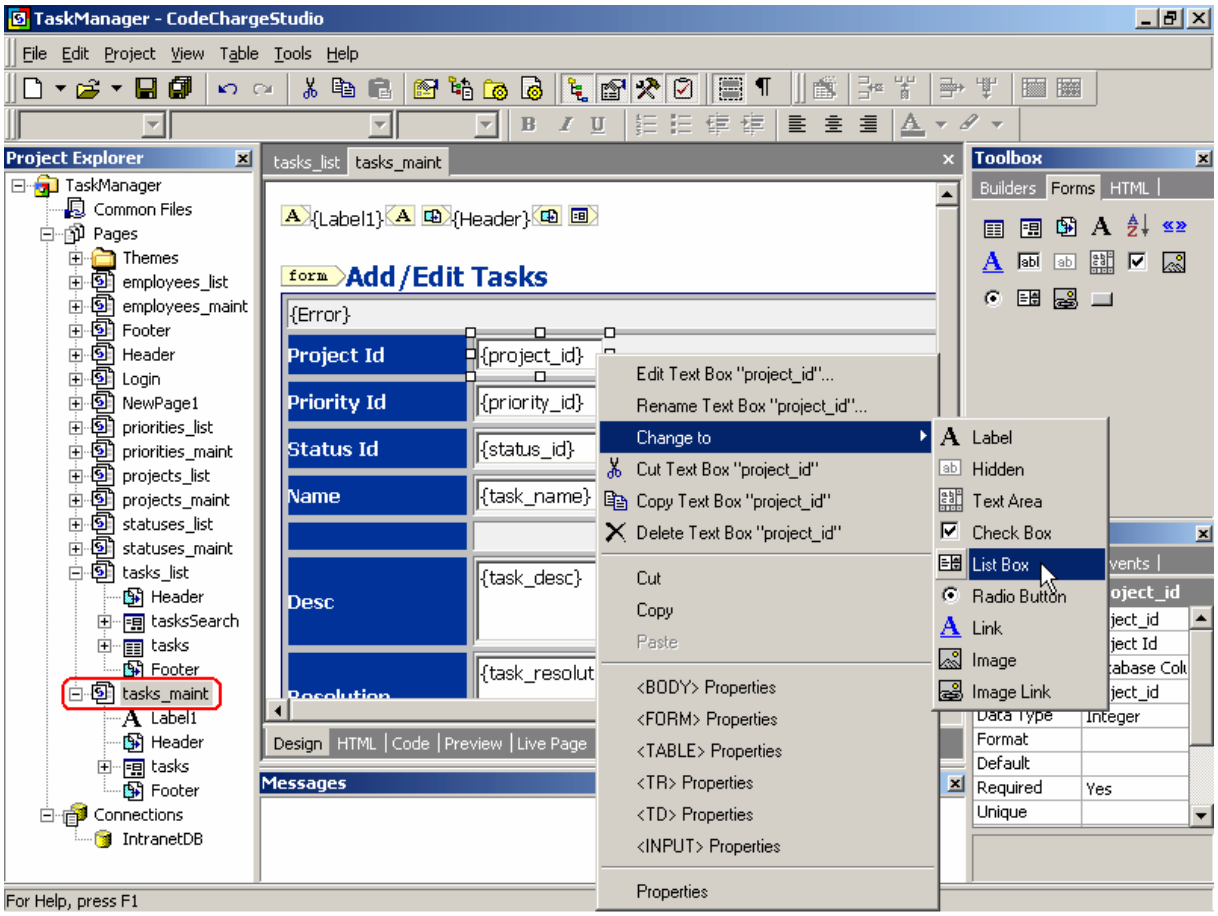
Finalizing the Task Maintenance Page

After using the Application Builder to generate the draft application and then adjusting the Task List page, we shall now see how to configure the Record form to update data.

Convert TextBox to a ListBox

Switch to “Design” mode and click on the [+] next to the *tasks_maint* page in Project Explorer.
This will open the Task Maintenance page in the document window. Notice that you are required to enter Project Id, Priority Id and Status Id as numeric values. It would be more convenient to present the user with the available options using Listboxes. In addition, some of the other fields may not need to be editable.

First, let’s change the Project Id TextBox field type to a ListBox.
Right-click on the {project_id} textbox in your design editor, and then select *Change to -> ListBox*.

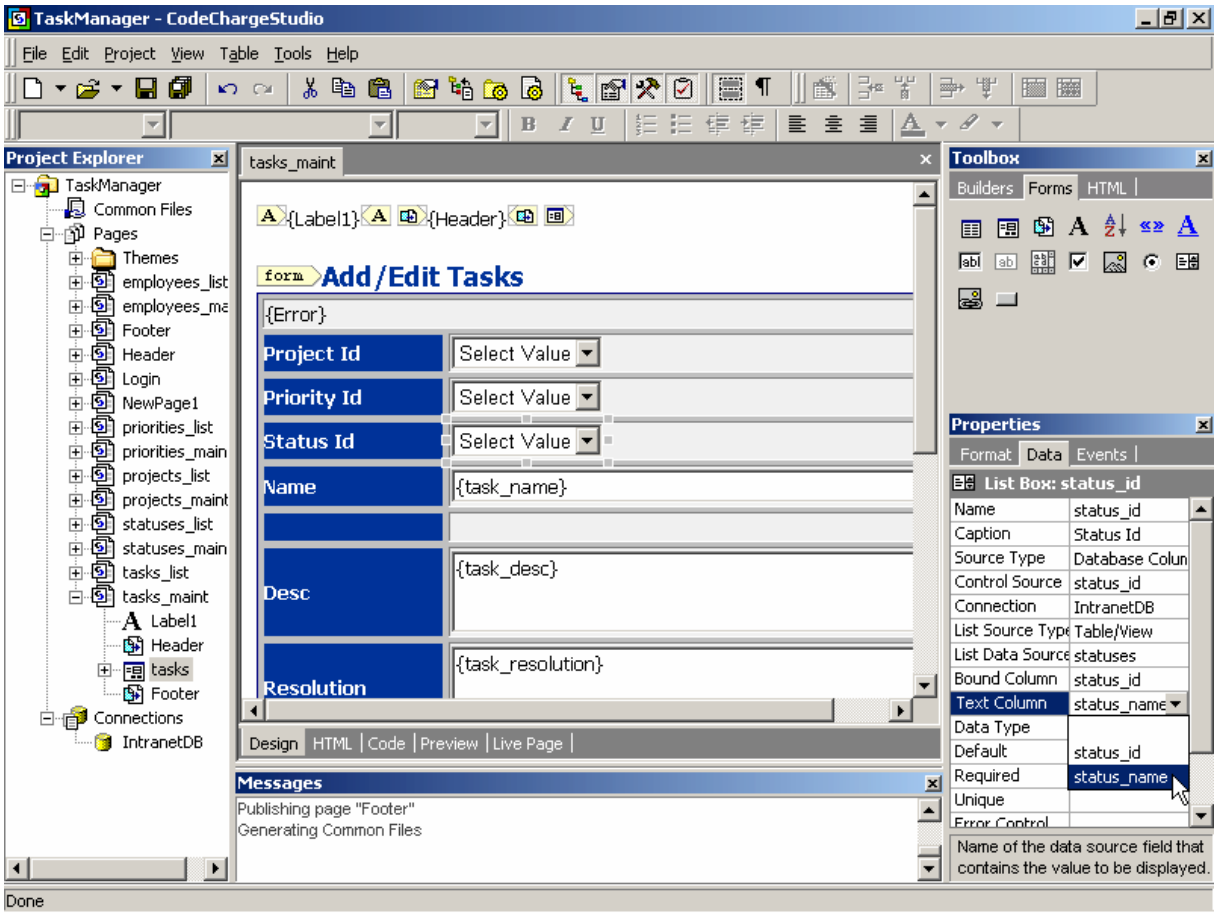


Configure ListBox Fields

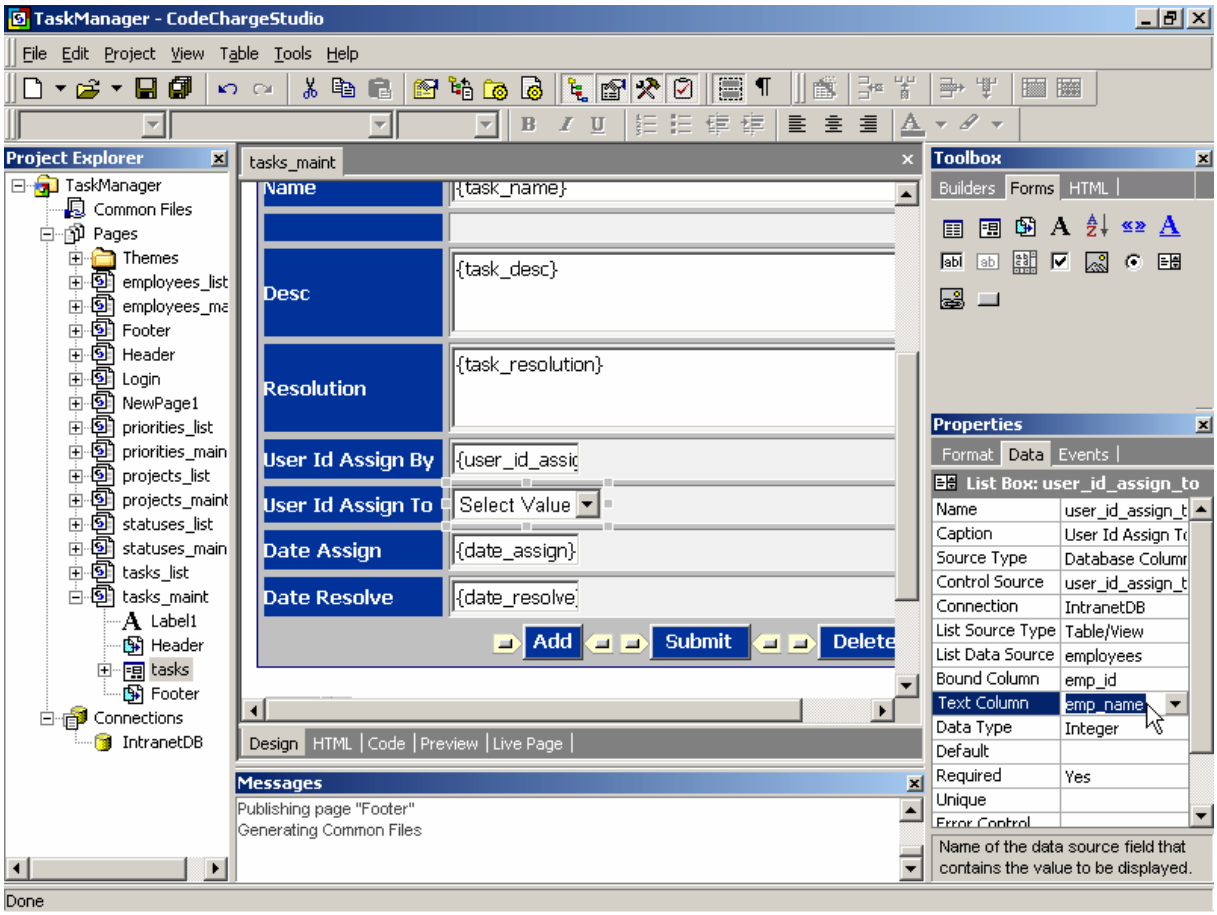
After converting the Project Id TextBox to a ListBox, you now need to configure the ListBox to display the list of Projects. Do so by specifying the property values for the ListBox as follows:

Connection: *IntranetDB* - database connection to use for retrieving ListBox values
List Data Source: *projects* - table containing ListBox values
Bound Column: *project_id* - table column whose values will be submitted by the ListBox
Text Column: *project_name* - table column whose values will be displayed in the ListBox
Data Type: *Integer* – data type of the value that will be submitted by the ListBox. (*project_id* is numeric)

Use the same approach to change *{priority_id}* and *{status_id}* TextBox fields to ListBox type and configure their properties accordingly. Your final page should contain three ListBox fields as shown below.

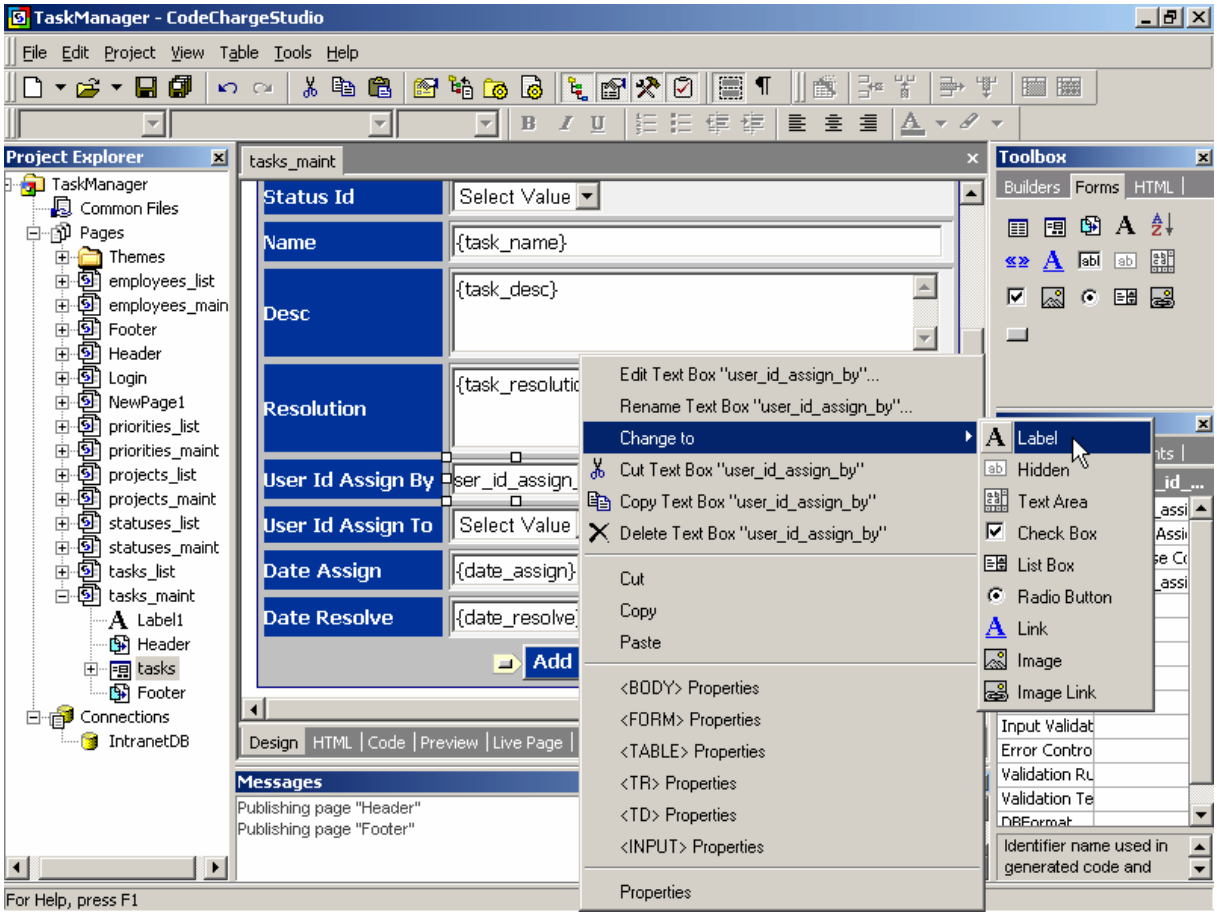


Scroll down the page and also convert the remaining field, *user_id_assign_to* field to a ListBox, so that users may assign a task to someone by selecting an employee name.



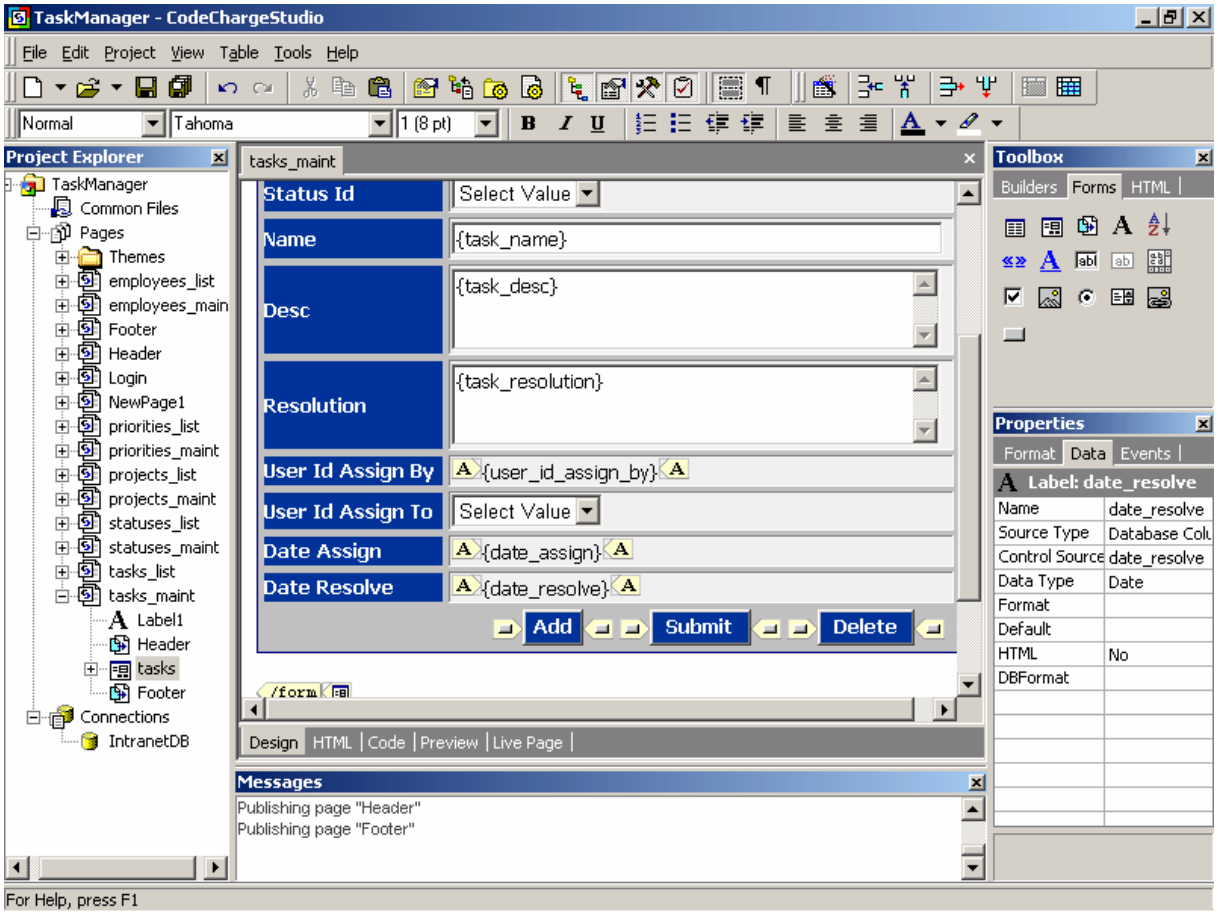
Create Label Fields

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 from TextBox type to Label.
Right-click on the *user_id_assign_by* field and select: Change To -> Label



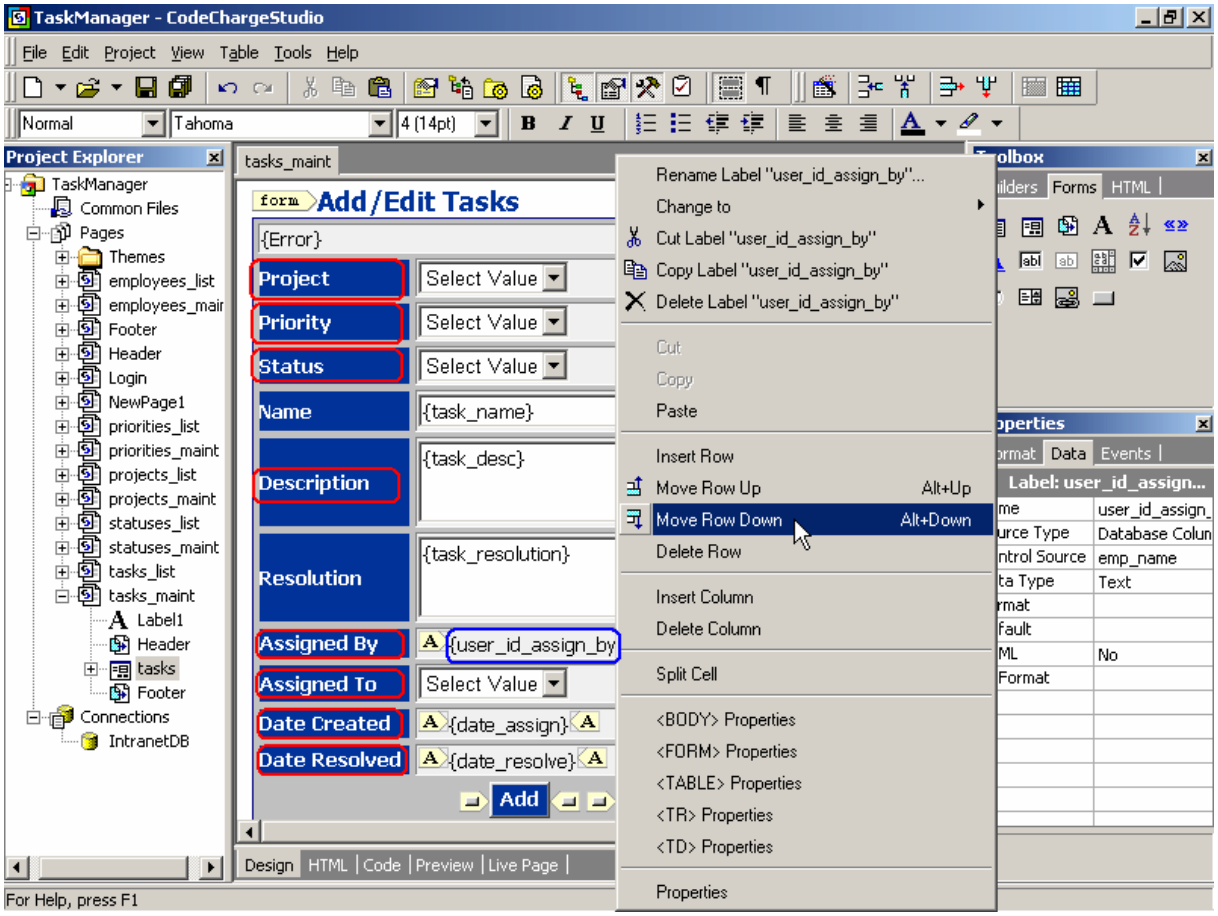
Repeat the same actions to convert the following fields to Labels:

date_assign
date_resolve



Rearrange and cleanup fields

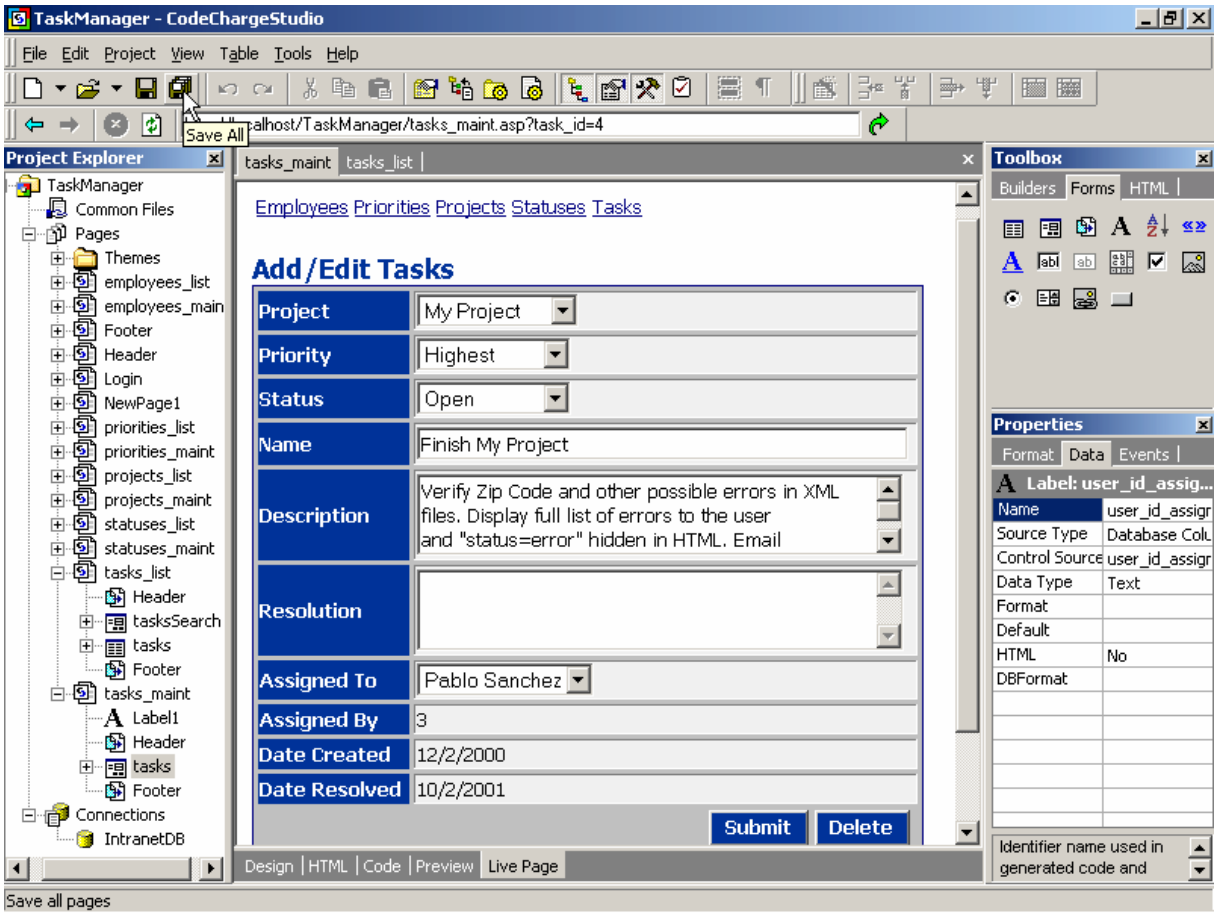
Now you can refine and finish the page by editing the text to be more descriptive, as well as moving all Label fields to the bottom of the form.
Change the field captions as shown, then right-click anywhere within the *user_id_assign_by* label, and select "Move Row Down".



Preview the Task Maintenance Page

Congratulations!
The basic version of your Task Management system is now ready. Preview the working Task Maintenance page by selecting the *tasks_list* page in Project Explorer, then selecting one of the existing tasks. You can also add a new task by clicking on “Add New” below the *tasks* grid on the main tasks_list page.

You may notice that the value of the “Assigned By” field is not shown correctly and several fields may be blank if you are creating a new task. You can improve your system by adding actions and programming code to the application. This will be addressed in the next section.



Enhancing Application Functionality with Programming Events

ATTENTION:
This section refers specifically to VBScript/ASP implementation of the task management example. Other technology implementations are described in the Appendix A, as follows:

ASP.NET (C#)
ColdFusion
JSP
PHP

You've probably noticed that until now you've built your Task Management application without having to deal with the programming code. CodeCharge Studio can help you build fairly functional systems without any programming; however creating more sophisticated systems will require a certain amount of programming code. Fortunately, CodeCharge Studio makes programming easy by providing you with a first-rate code editor, in addition to Events and Actions that aid you in inserting pre-programmed snippets of code into the right place within the program.

Here are the definitions of Action and Event:

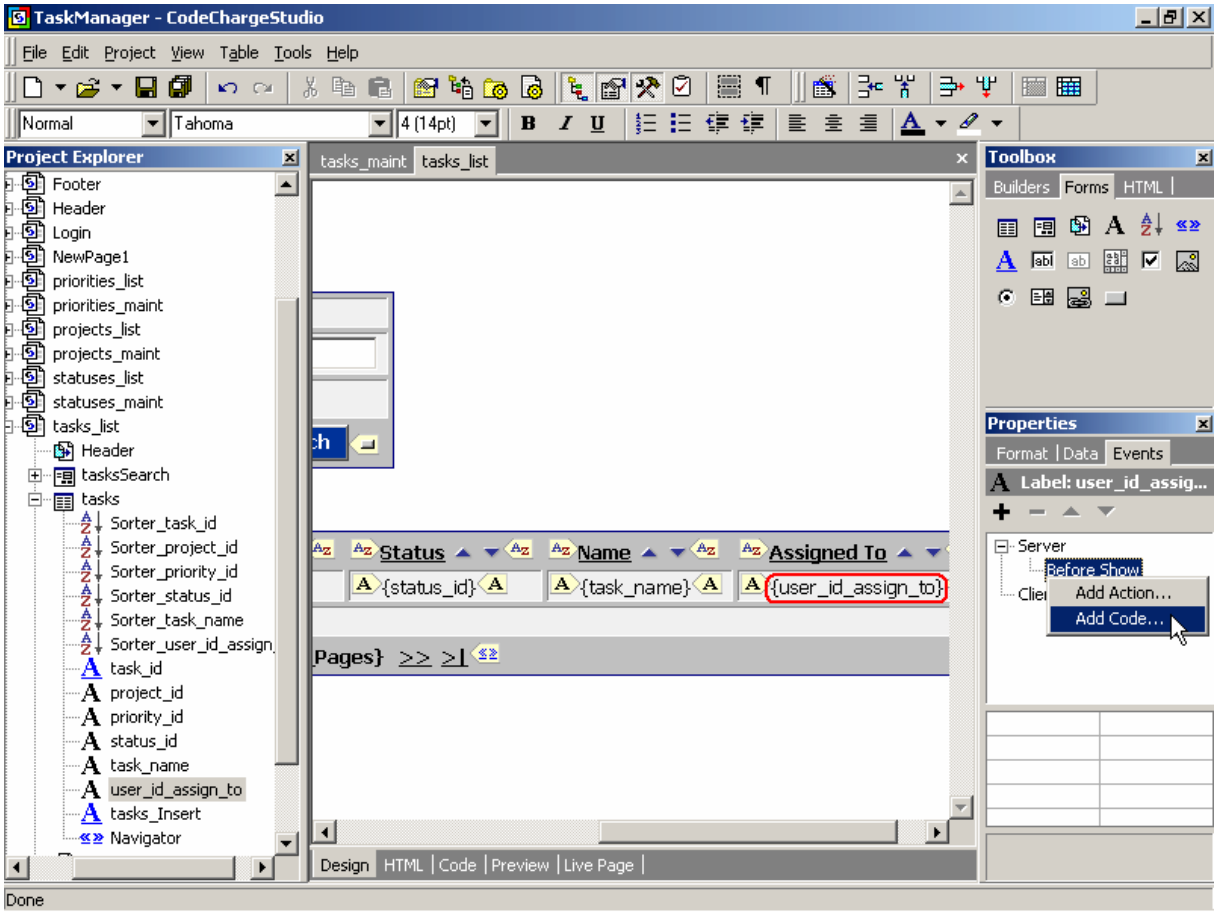
Action
User-definable code generation component, which inserts block of code into an event procedure. CodeCharge Studio comes with several predefined Actions, which are installed into the following folder:
(CCS folder)\Components\Actions
Internally, actions consist of XML and XSL code that can be easily customized.
For example an action can be set on a TextBox to validate the e-mail address.

Event Procedure
A procedure automatically executed in response to an event initiated by the program code during execution. Events are the best place for putting custom programming code.

Use "Before Show" Event to Alter Grid's Output

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.

Open the *tasks_list* page in the "Project Explorer", expand the *tasks* grid, and right-click on the *user_id_assign_to* field and select "Properties". Under the "Data" tab, set the value of the "Content" property to "HTML". Next, select the "Events" tab in the Properties window, then 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.



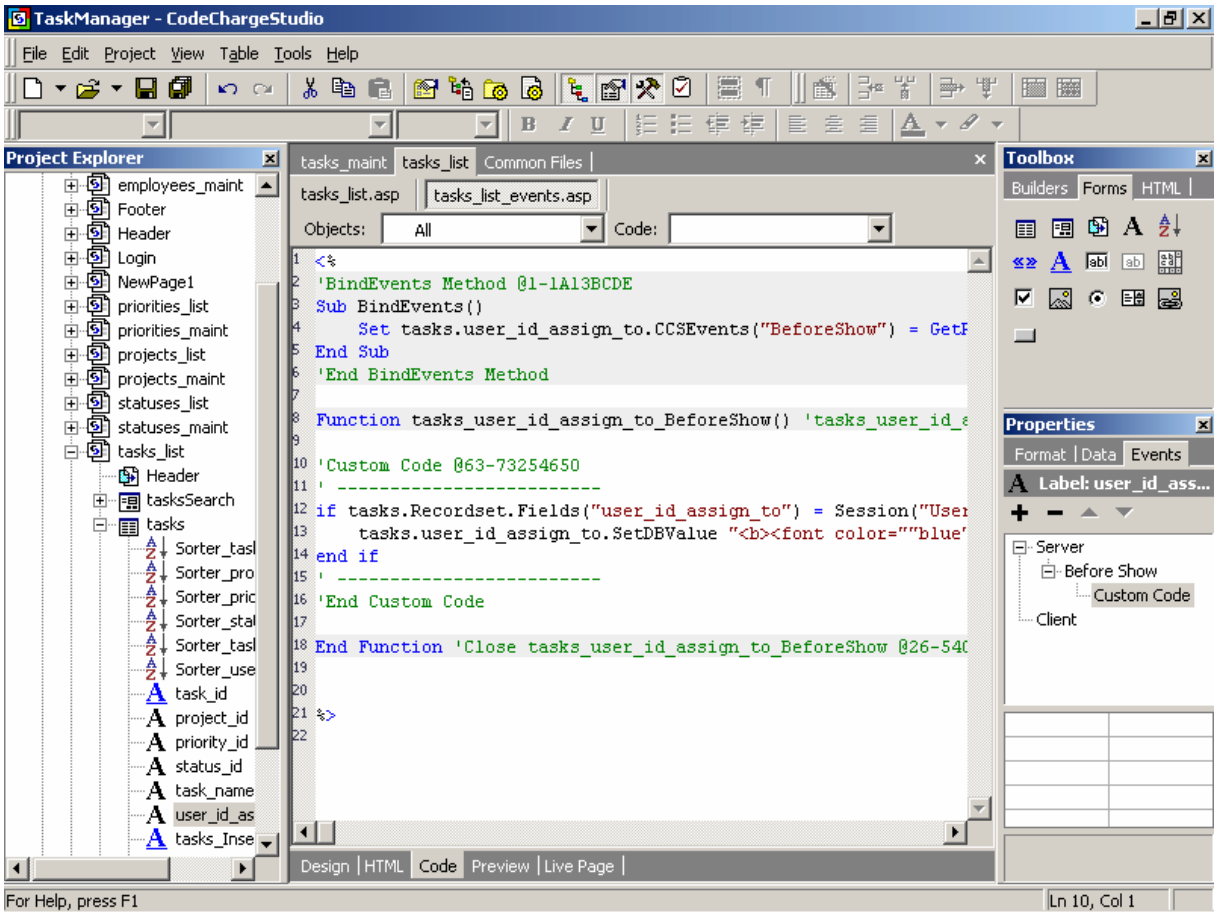
Programmatically Control Field's Value

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

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

with the following lines (ASP/VBScript):

```
if tasks.Recordset.Fields("user_id_assign_to") = Session("UserID") then
   tasks.user_id_assign_to.Value = "<b><font color=""blue""">" & tasks.user_id_assign_to.Value &
      "</font></b>"
end if
```



Let's explain how the above VBScript code works:

```
if tasks.Recordset.Fields("user_id_assign_to") = Session("UserID") then
```

This is an "if" condition that is true only if the database field *user_id_assign_to* is equal to the id 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 id to the id of the person that a task is assigned to. "UserID" is one of session variables used by CodeCharge-generated programs, and it holds the ID of the currently logged in user until the session expires. 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.user_id_assign_to.Value = "<b><font color=""blue"">" & tasks.user_id_assign_to.Value &
"</font></b>"
```

This code is executed if the previous "if" condition is met. It modifies the value of the *user_id_assign_to* field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML <*b*> 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 *user_id_assign_to* field in the *tasks* grid. *Value* is a property of that field, which you can both read and modify.

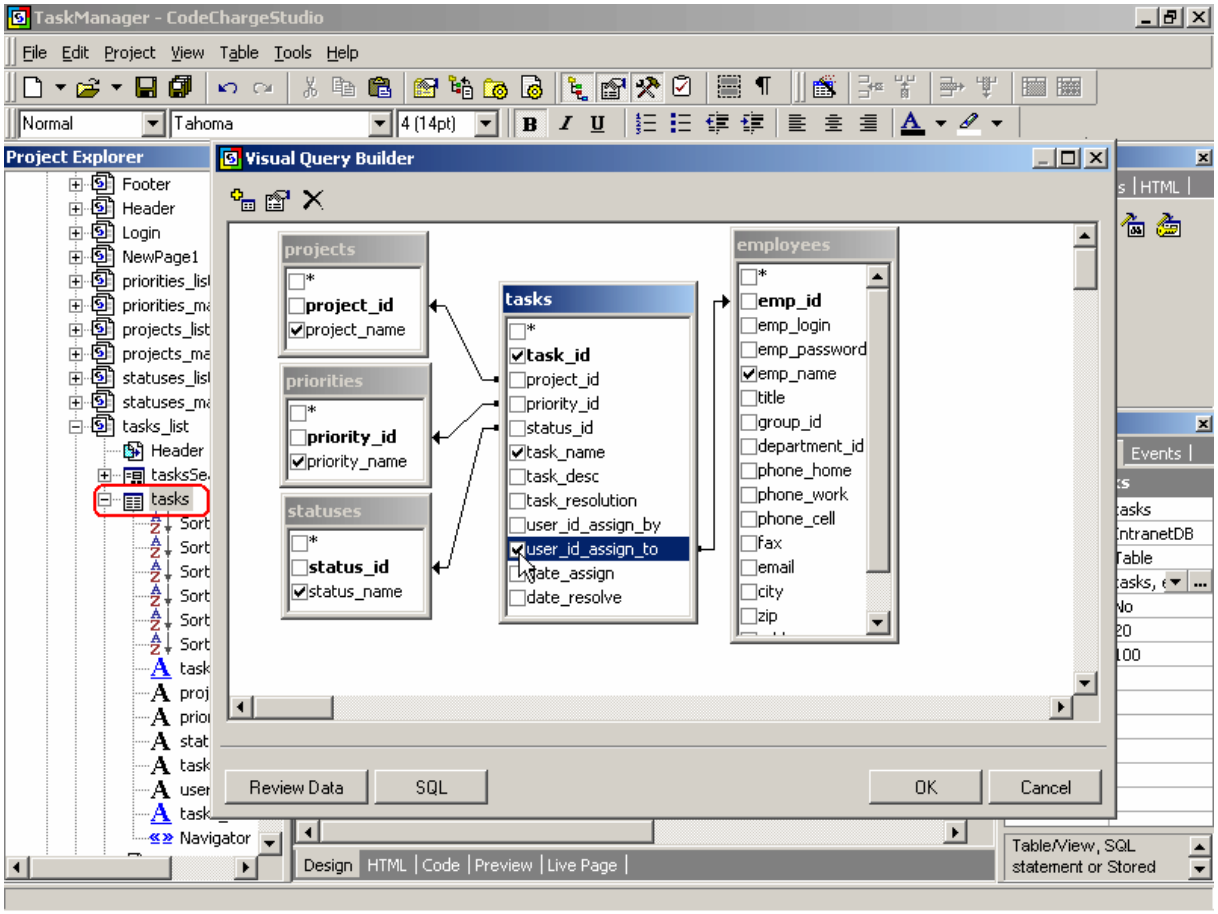
```
end if
```

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

Although the code you've inserted is complete, it may not work if the data source of your grid doesn't contain the value of the *user_id_assign_to*, which is compared against user's id. The next step is to add this database field to the grid's data source.

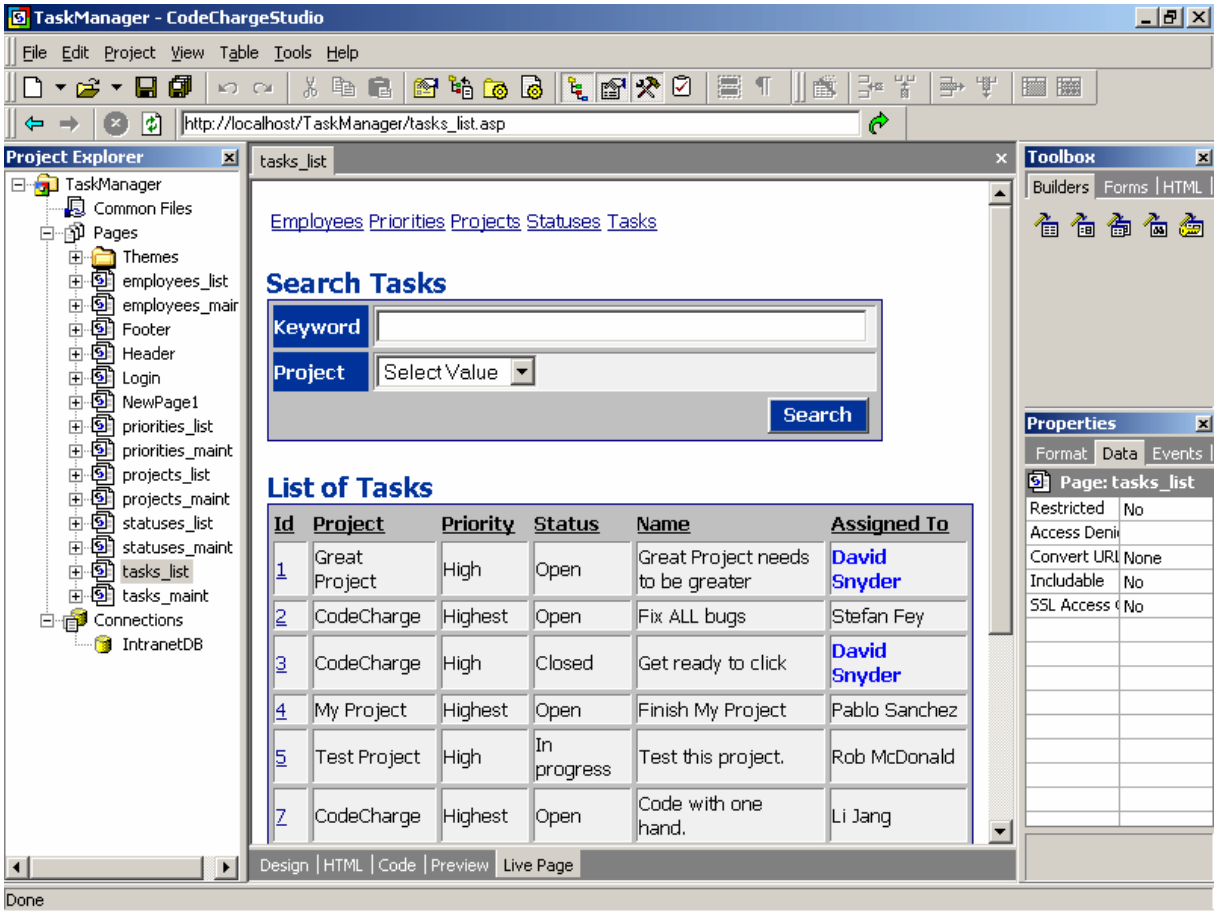
Specify Additional Database Fields for the Grid

Switch to Design mode, select the *tasks* grid in Project Explorer (or click on grid's caption on the screen), then click on the […] button next to grid's "Data Source" property, then click "Build Query" to open the Visual Query Builder window.
Once in Visual Query Builder, add the *user_id_assign_to* field to the data source by selecting its checkbox. Notice that the *tasks* table is connected to *employees* table because both *user_id_assign_to* and *emp_id* fields are related. You will be ale to assign a task to someone by specifying an employee and his id.



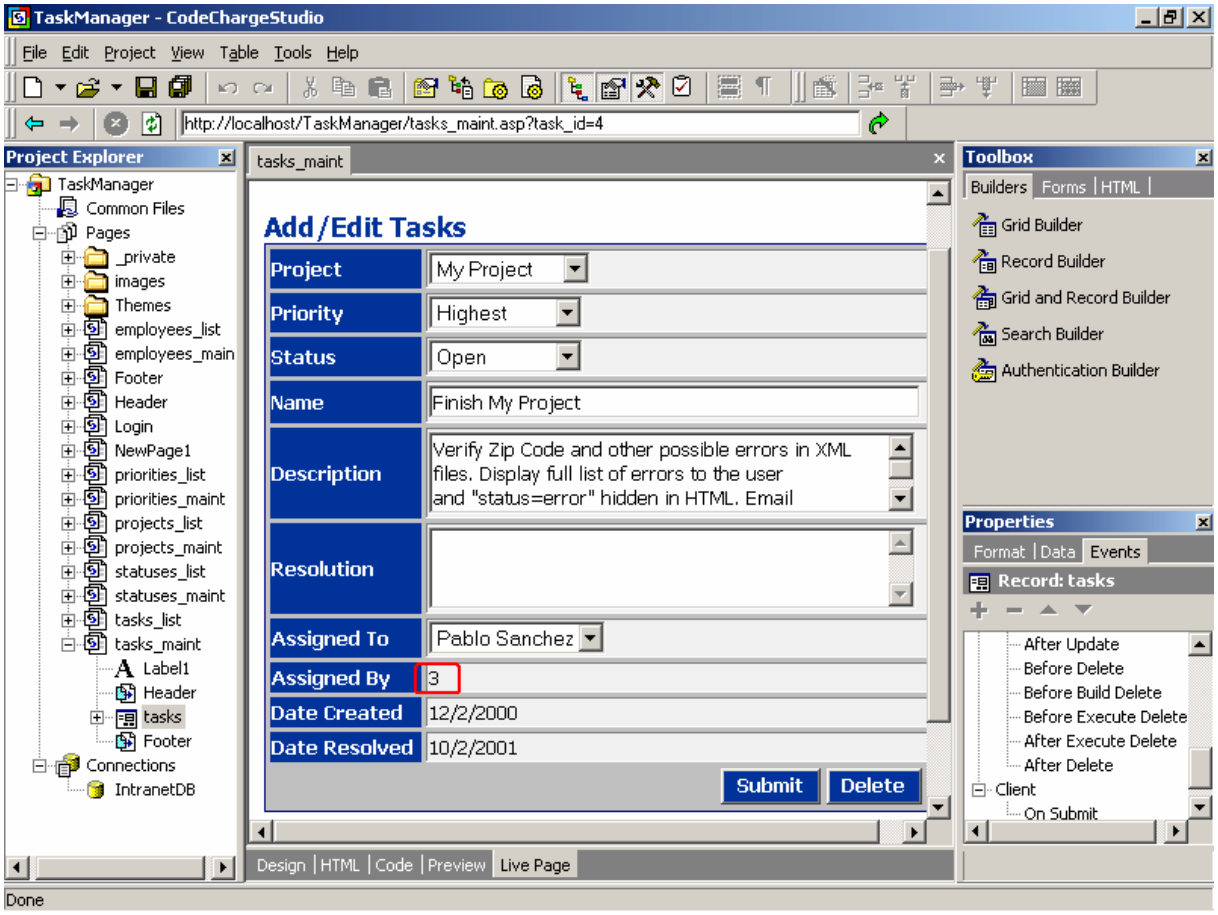
Preview Tasks List Page

Save your project and 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.
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. Click on any of the project Ids and you should see the login page.
Login as davids / davids , then 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.



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 *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.



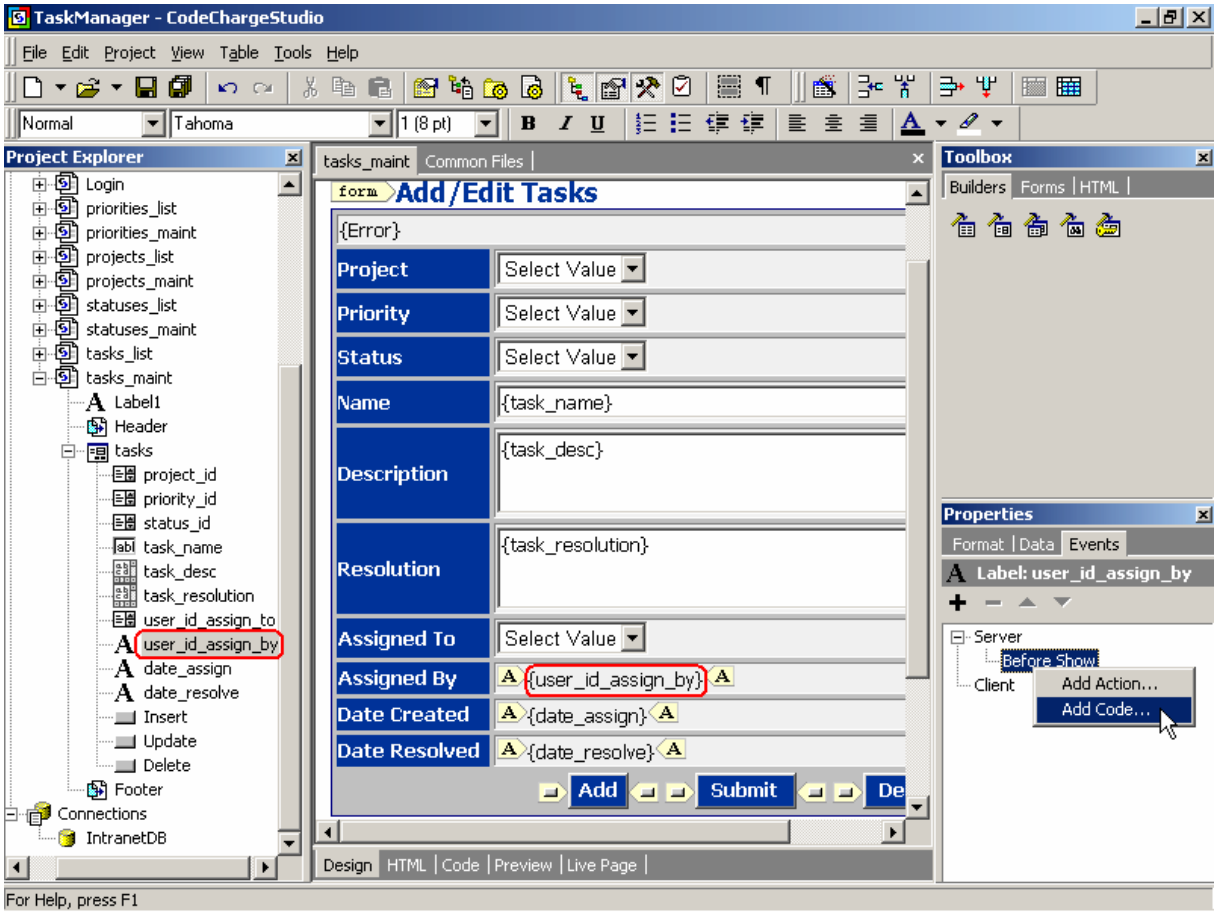
There are several potential methods of dealing with the above issue and let's explain each one in detail:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, 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.

Create “Before Show” Event to Alter Label’s Value

Select the Label *user_id_assign_by* in the Design mode, then in the Properties window click on “Events” tab.
Right-click on “Before Show” event and select “Add Code…”.
CodeCharge Studio should automatically switch to Code view at that time.
“Before Show” event is a place in the program that is executed after a value is assigned to the component, but before such value is displayed.



Use "Before Show" Event to Alter Label's Value

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

```
'Custom Code @28-73254650
' -----------------------
' Write your own code here.
' -----------------------
'End Custom Code
```

Replace the text:

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

with the following:

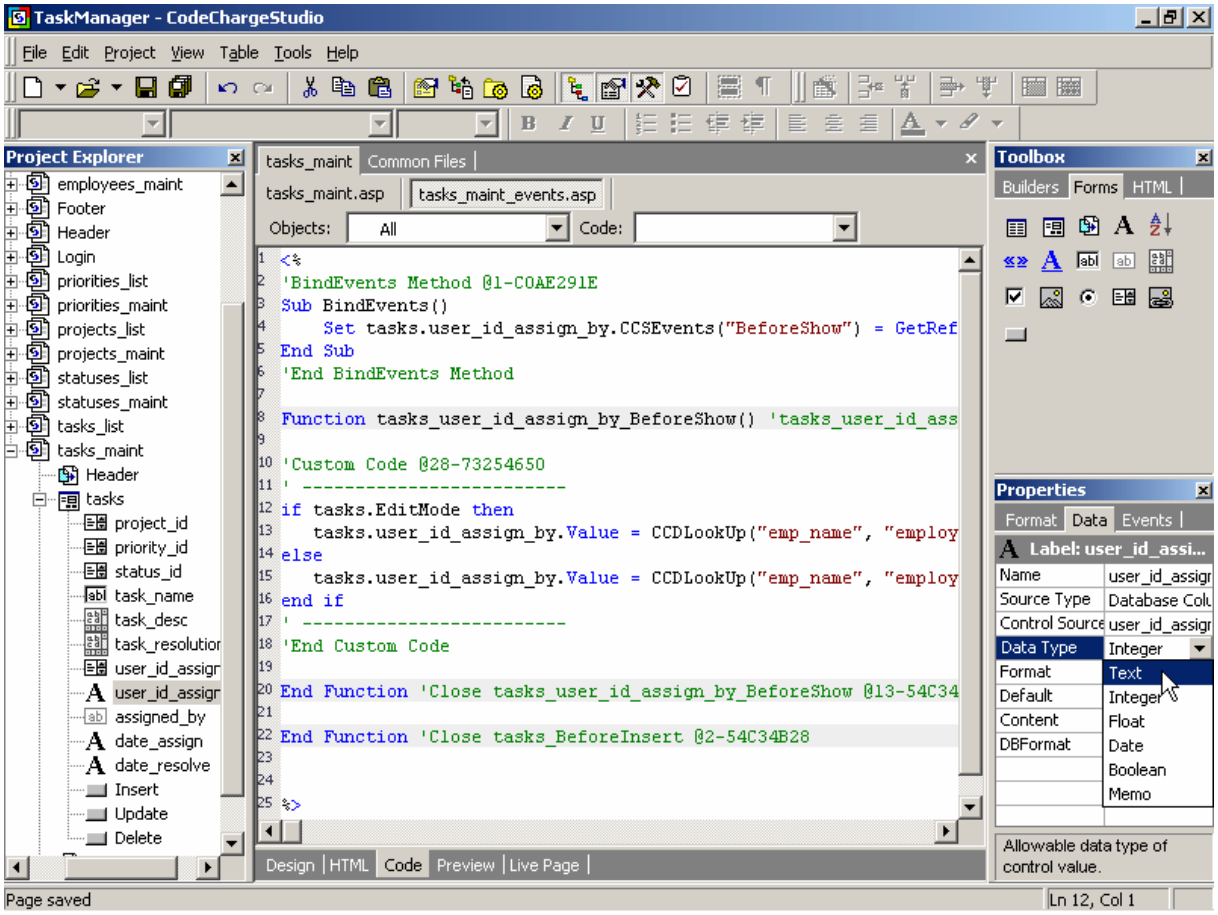
```
if tasks.EditMode then
   tasks.user_id_assign_by.Value = CCDLookUp("emp_name", "employees", "emp_id=" &
      CCToSQL(tasks.user_id_assign_by.Value,"Integer") , DBIntranetDB)
else
   tasks.user_id_assign_by.Value = CCDLookUp("emp_name", "employees", "emp_id=" &
      CCToSQL(CCGetUserID,"Integer"), 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 *tasks* Grid.
CCDLookup –CodeCharge function that supports retrieving database value based on field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table under condition that the key (*emp_id*) equals the current value of the Label.
CCToSQL – 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 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 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 *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 CCDLookup function that uses *IntranetDB* connection and *CCGetUserID* function that obtains current user’s ID.
”

Refer to the CodeCharge Studio Programming Reference for more information about functions and properties available in CodeCharge-generated programs.

Now that you programmatically modified the value of the *user_id_assign_by* Label to output Employee Name instead of the ID, you will also need to specify that this field is now a Text field, instead of Numeric.
Click on “Data” tab in Properties window, and select “Text” as the Data Type.



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

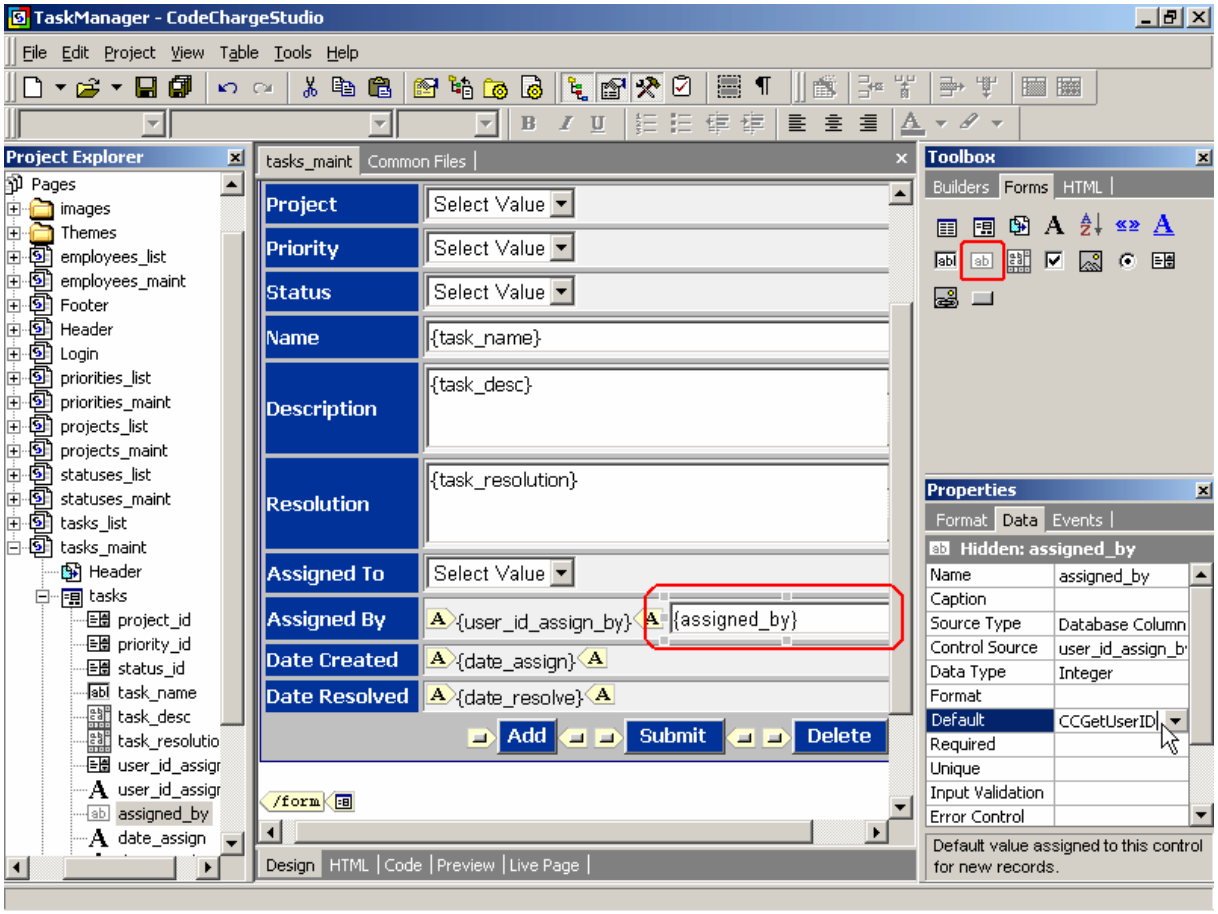
You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.
First, add a Hidden field to your page from the "Forms" tab on the "Toolbox" window. This field type isn't visible in the browser, but will be used to store values and update the database.
Then 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.

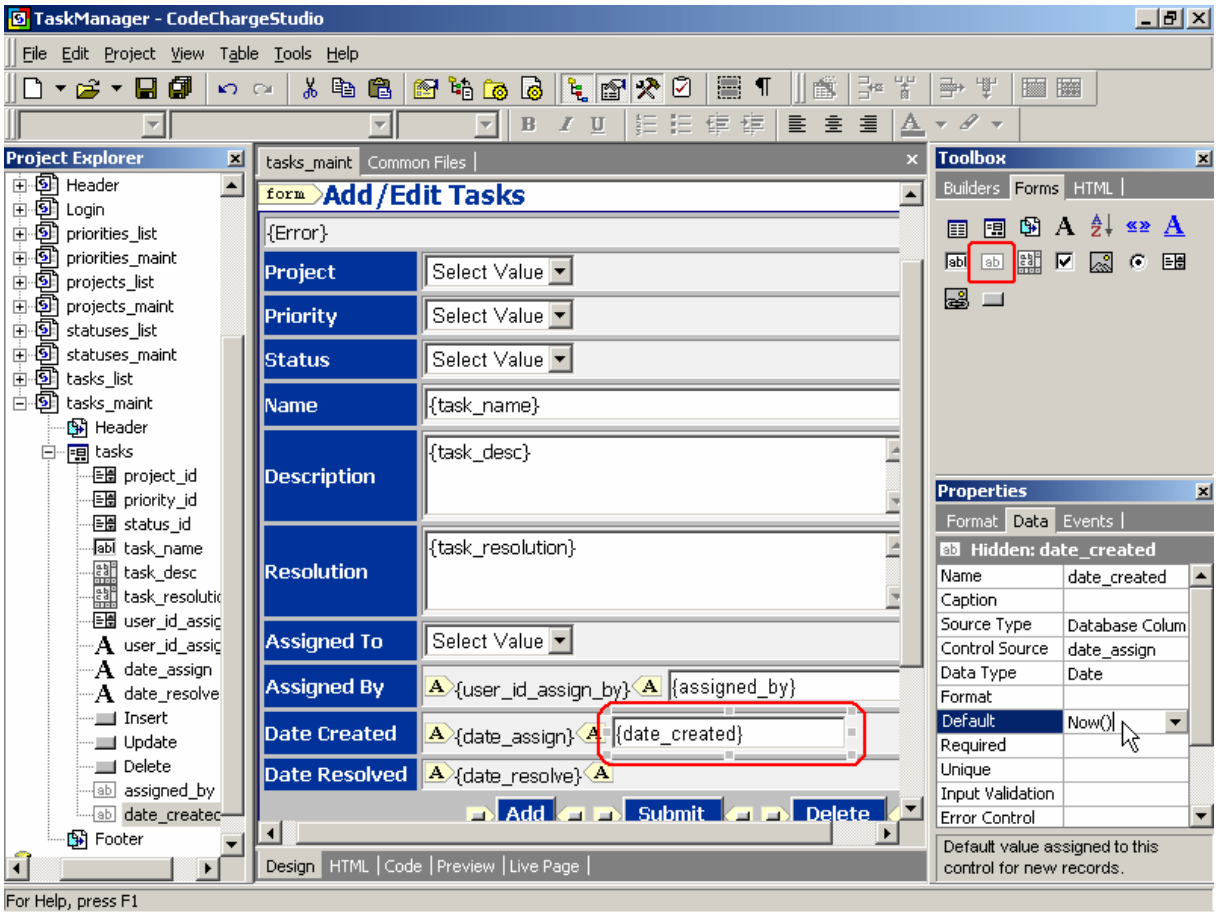
Default: *CCGetUserID* – default value for this field if empty. CCGetUserID is a CodeCharge function that retrieves the ID of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
Pages
images
Themes
employees_list
employees_maint
Footer
Header
Login
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolutio
user_id_assigr
user_id_assigr
assigned_by
date_assign
tasks_maint
Common Files
Project
Select Value
Priority
Select Value
Status
Select Value
Name
{task_name}
Description
{task_desc}
Resolution
{task_resolution}
Assigned To
Select Value
Assigned By
{user_id_assign_by}
{assigned_by}
Date Created
{date_assign}
Date Resolved
{date_resolve}
Add
Submit
Delete
/form
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Hidden: assigned_by
Name assigned_by
Caption
Source Type Database Column
Control Source user_id_assign_b
Data Type Integer
Format
Default CCGetUserID
Required
Unique
Input Validation
Error Control
Default value assigned to this control for new records.

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.
Configure the new field as follows:

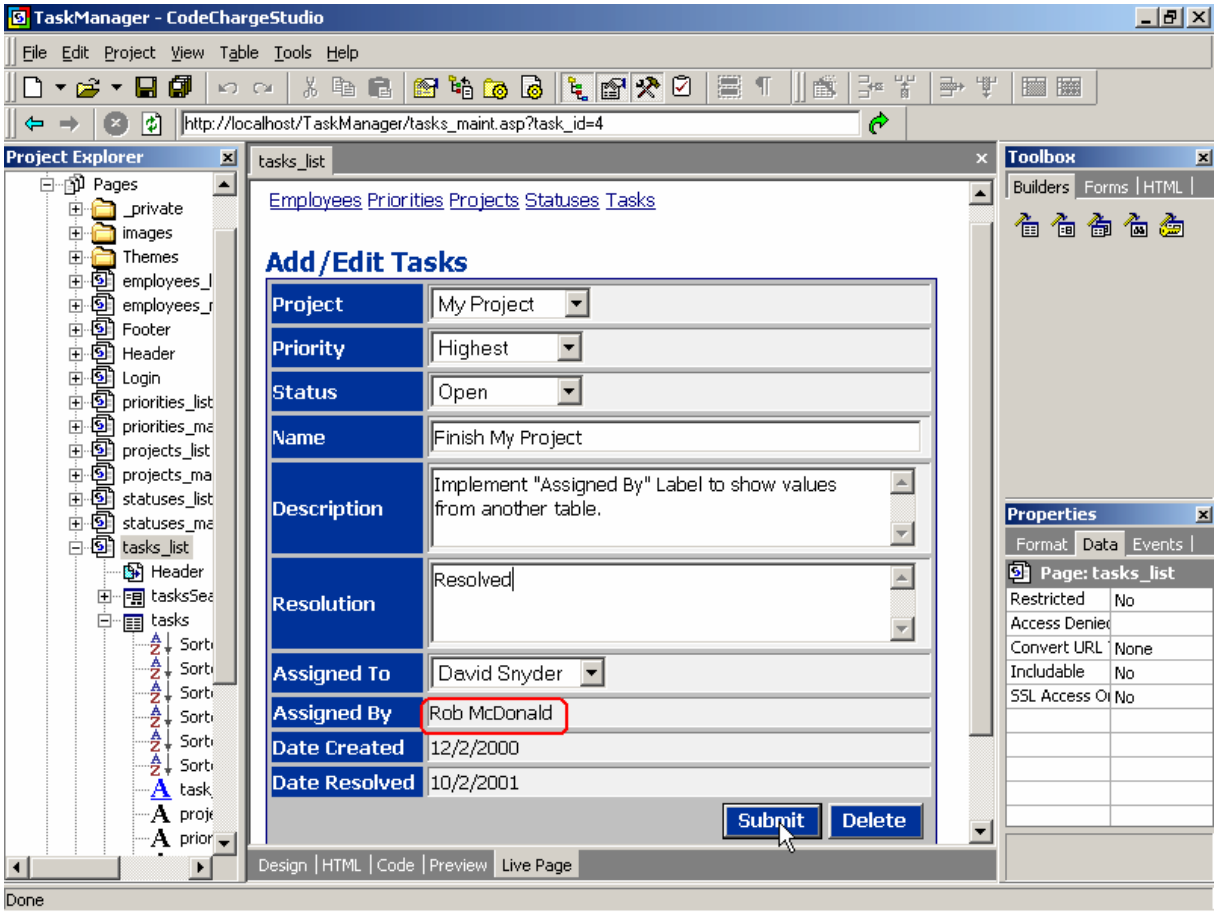
Name: *date_created*
Control Source: *date_assign*
Data Type: *Date*
Default: *Now()* – Now() is a VBScript(ASP) function that obtains the server's current date and time. Using this function allows you to automatically assign the current date and time to new tasks. The Default property doesn't affect existing records, thus the date of existing tasks won't be modified during updates.



Test the Label and Hidden Fields

Finally, you can switch to Live Page mode, select a Task, Login, and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.
Don't forget to save it!



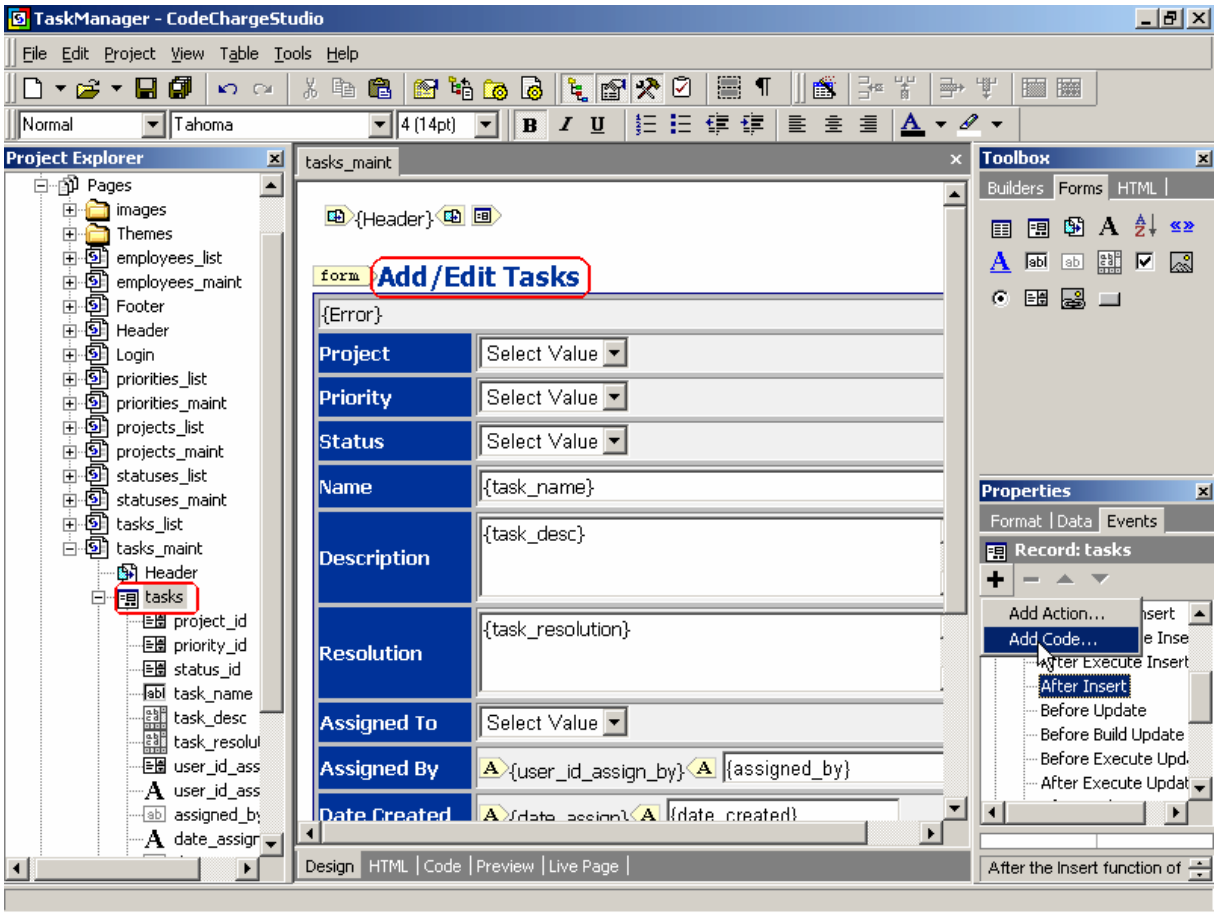
Programming the Record Form

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

Add Code in the “After Insert” Event to Send Emails

Select the “tasks” form by selecting it in the Project Explorer, or clicking anywhere within the form’s caption.
Then in the Properties window click on the “Events” tab and select the “After Insert” event.
Click on the [+] button, then select “Add Code…”



Once your 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=" & CCToSQL(CCGetUserID,"Integer") ,
  DBIntranetDB)
Mailer.FromName = CCDLookUp("emp_name", "employees", "emp_id=" &
  CCToSQL(CCGetUserID,"Integer") , DBIntranetDB)
Mailer.AddAddress CCDLookUp("email", "employees", "emp_id=" &
  CCToSQL(tasks.user_id_assign_to.Value,"Integer") , 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=" &
    CCToSQL(CCGetUserID,"Integer"), 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:

a) The above code requires that you install on your server the free Email component “ASPEmail”, which you may obtain from http://www.aspemail.com/
There are many other email components and you can modify the above program by reading support materials covering the component you choose to use.

b) 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 initialize the ASPEmail component.

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

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

```
Mailer.AddAddress CCDLookUp("email", "employees", "emp_id=" &
  CCToSQL(tasks.user_id_assign_to.Value,"Integer") , 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=" &
    CCToSQL(CCGetUserID,"Integer"), DBIntranetDB) & _
  "<br><br>" & tasks.task_desc.Text
```

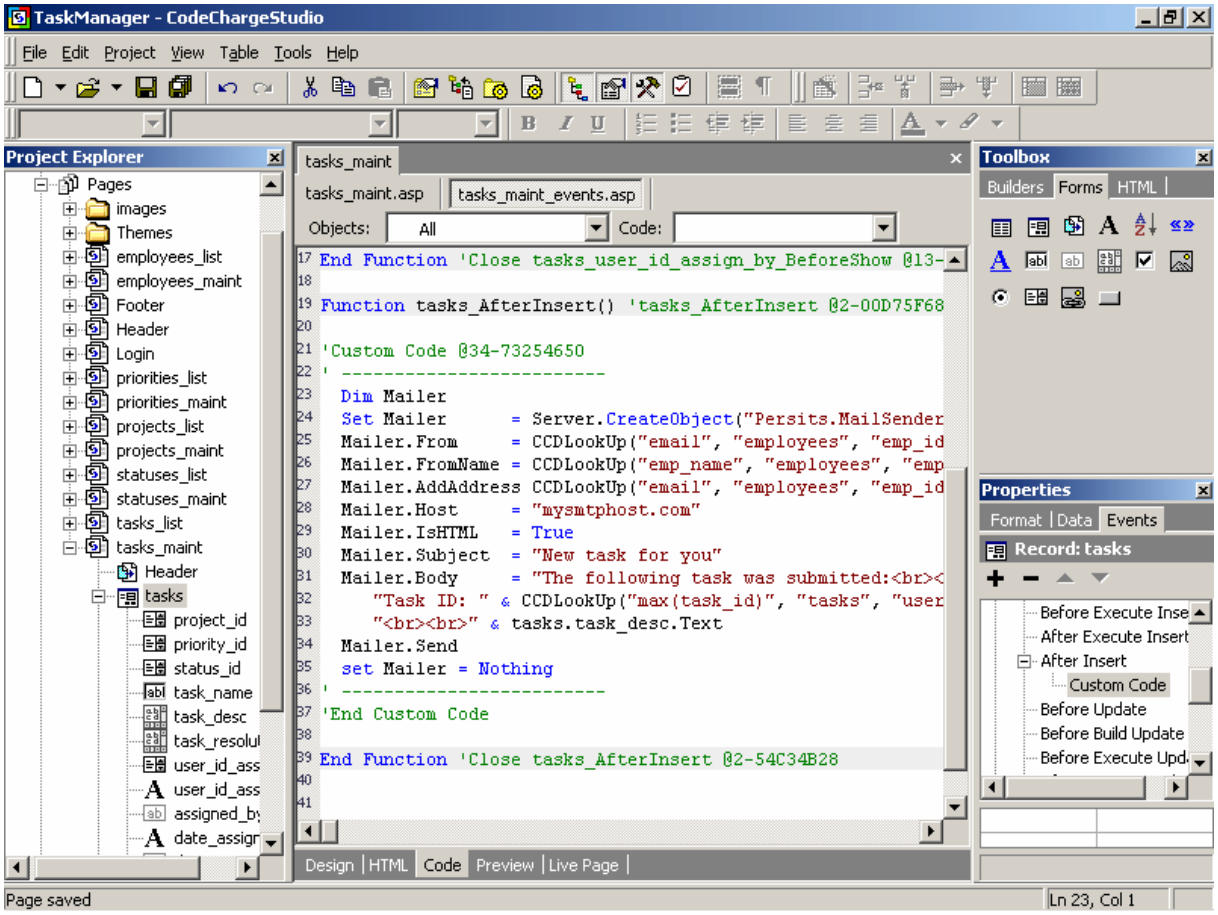
The body of the email which consists of the task description and the task id. The last inserted task id can be obtained using different methods with different databases. Unfortunately, MS Access doesn’t support the retrieval of the last inserted record, therefore you will need to use the CCDLookUp function 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.



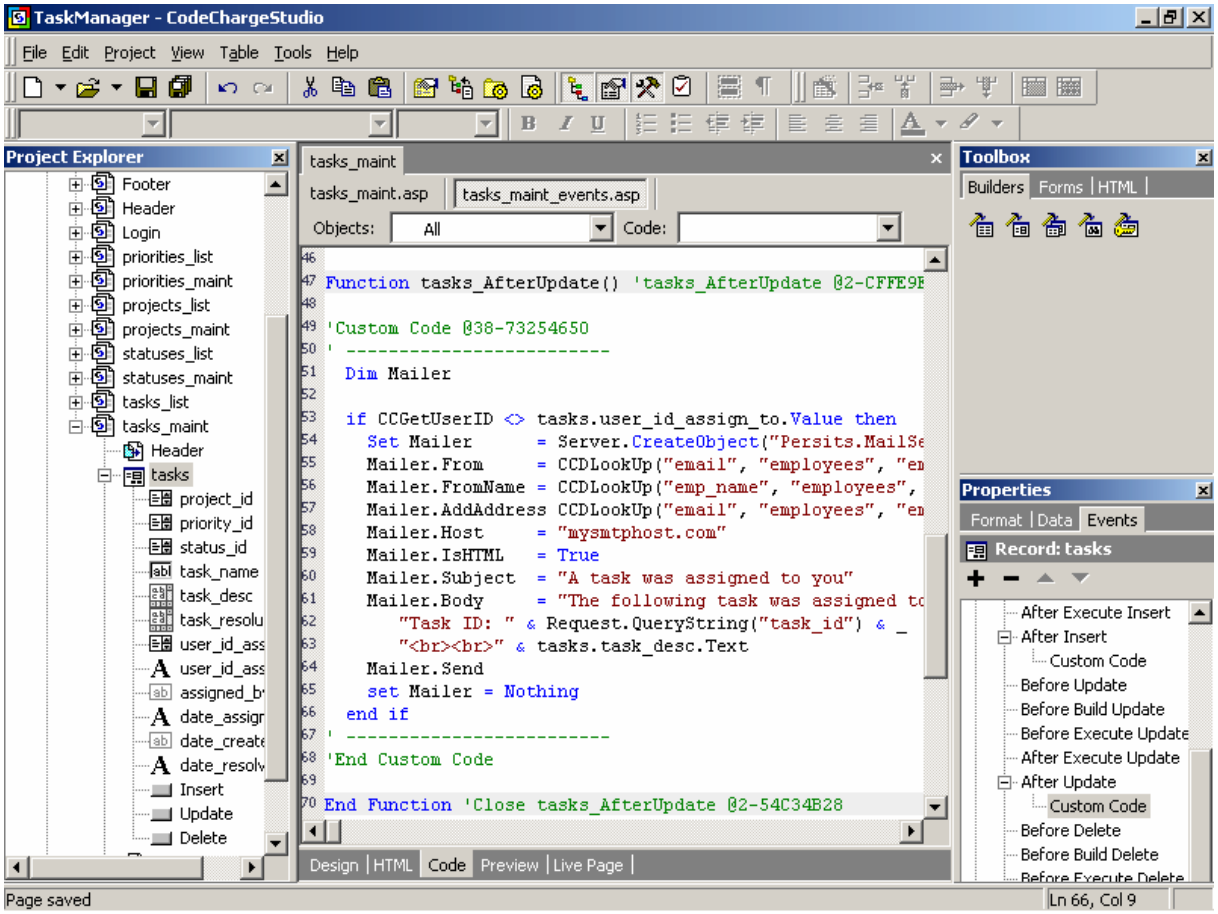
Use the "After Update" Event to Send Emails

You previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. Now implement similar functionality in "After Update" Event to notify assignee when an existing task is updated and reassigned to someone.
Click on the "tasks" form in the Project Explorer, then in the Properties window, select the "Events" tab, and then add the following "Custom Code" in "After Update" event:

```
Dim Mailer
if CCGetUserID <> tasks.user_id_assign_to.Value then
  Set Mailer      = Server.CreateObject("Persits.MailSender")
  Mailer.From     = CCDLookUp("email", "employees", "emp_id=" & CCToSQL(CCGetUserID,"Integer") ,
    DBIntranetDB)
  Mailer.FromName = CCDLookUp("emp_name", "employees", "emp_id=" &
    CCToSQL(CCGetUserID,"Integer") , DBIntranetDB)
  Mailer.AddAddress CCDLookUp("email", "employees", "emp_id=" &
    CCToSQL(tasks.user_id_assign_to.Value,"Integer") , DBIntranetDB)
  Mailer.Host     = "mysmtphost.com"
  Mailer.IsHTML   = True
  Mailer.Subject  = "A task was assigned to you"
  Mailer.Body     = "The following task was assigned to you:<br><br>" & _
    "Task ID: " & Request.QueryString("task_id") & _
    "<br><br>" & tasks.task_desc.Text
  Mailer.Send
  set Mailer = Nothing
end if
```

The main differences between the above code and the one you used in "After Insert" event are as follows:

- An "if" condition was added to send an email only if a user assigns a task to someone other than himself/herself
- *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. A such a URL would look like this:
  *http://localhost/TaskManager/tasks_maint.asp?**task_id=9***


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
Footer
Header
Login
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolu
user_id_ass
user_id_ass
assigned_b
date_assigr
date_create
date_resolv
Insert
Update
Delete
tasks_maint
tasks_maint.asp
tasks_maint_events.asp
Objects: All
Code:
Function tasks_AfterUpdate() 'tasks_AfterUpdate @2-CFFE9
'Custom Code @38-73254650
' ----------
Dim Mailer
if CCGetUserID <> tasks.user_id_assign_to.Value then
Set Mailer = Server.CreateObject("Persits.MailSe
Mailer.From = CCDLookUp("email", "employees", "en
Mailer.FromName = CCDLookUp("emp_name", "employees",
Mailer.AddAddress CCDLookUp("email", "employees", "en
Mailer.Host = "mysmtphost.com"
Mailer.IsHTML = True
Mailer.Subject = "A task was assigned to you"
Mailer.Body = "The following task was assigned to
"Task ID: " & Request.QueryString("task_id") & _
"

" & tasks.task_desc.Text
Mailer.Send
set Mailer = Nothing
end if
' ----------
'End Custom Code
End Function 'Close tasks_AfterUpdate @2-54C34B28
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Record: tasks
After Execute Insert
After Insert
Custom Code
Before Update
Before Build Update
Before Execute Update
After Execute Update
After Update
Custom Code
Before Delete
Before Build Delete
Page saved
Ln 66, Col 9

Test Email Delivery

Before testing the system, you should add new users to your database with correct email addresses, or modify the existing test users by changing their email address. You can do this by opening the Intranet.mdb database that is located in your project directory. Alternatively, you may use the Task Manager itself and go to the Employees page to view and modify user emails there.
(Note: You will need Ms Access 2000 to manually edit the database file.)

Once you have users configured with test emails, save your project and switch to Live Page mode to test your system. In the Task Maintenance page, if you experience an error like "Invalid class string", it probably means that your ASPEmail component wasn't installed correctly. If your email component was properly installed, 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.

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 explain several of them.

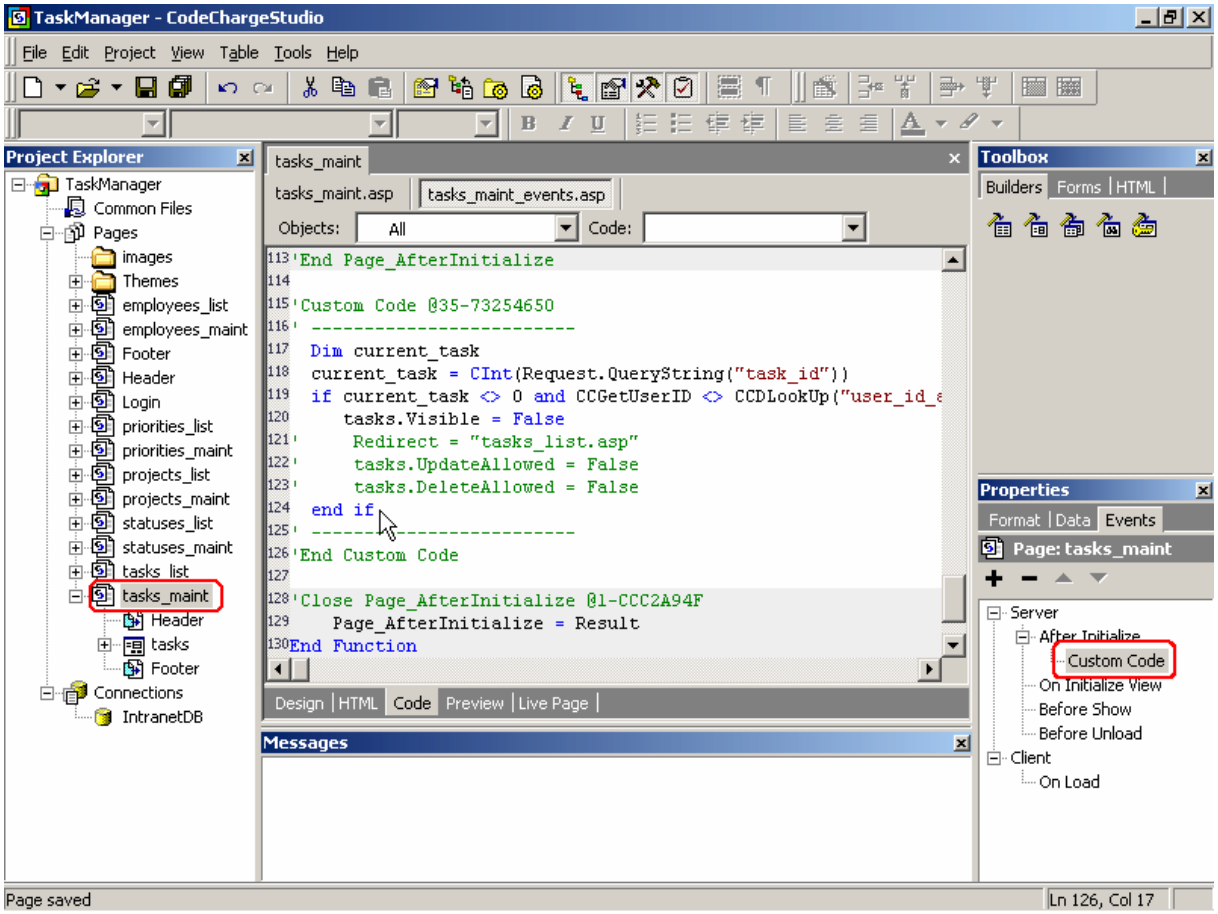
First, click on the “tasks_maint” page in the Project Explorer, then select “Events” tab in the Properties window. Add “Custom Code” to the “After Initialize” event of the page.

Once in the Code view, replace the generated comment:

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

with the code below:

```
Dim current_task
current_task = CInt(Request.QueryString("task_id"))
if current_task <> 0 and CCGetUserID <> CCDLookUp("user_id_assign_to", "tasks", "task_id=" &
CCToSQL(current_task, "Integer"),  DBIntranetDB) then
    tasks.Visible = False
'    Redirect = "tasks_list.asp"
'    tasks.UpdateAllowed = False
'    tasks.DeleteAllowed = False
end if
```



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

1. **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.
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 opposite, mutually exclusive criteria.

2. **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:

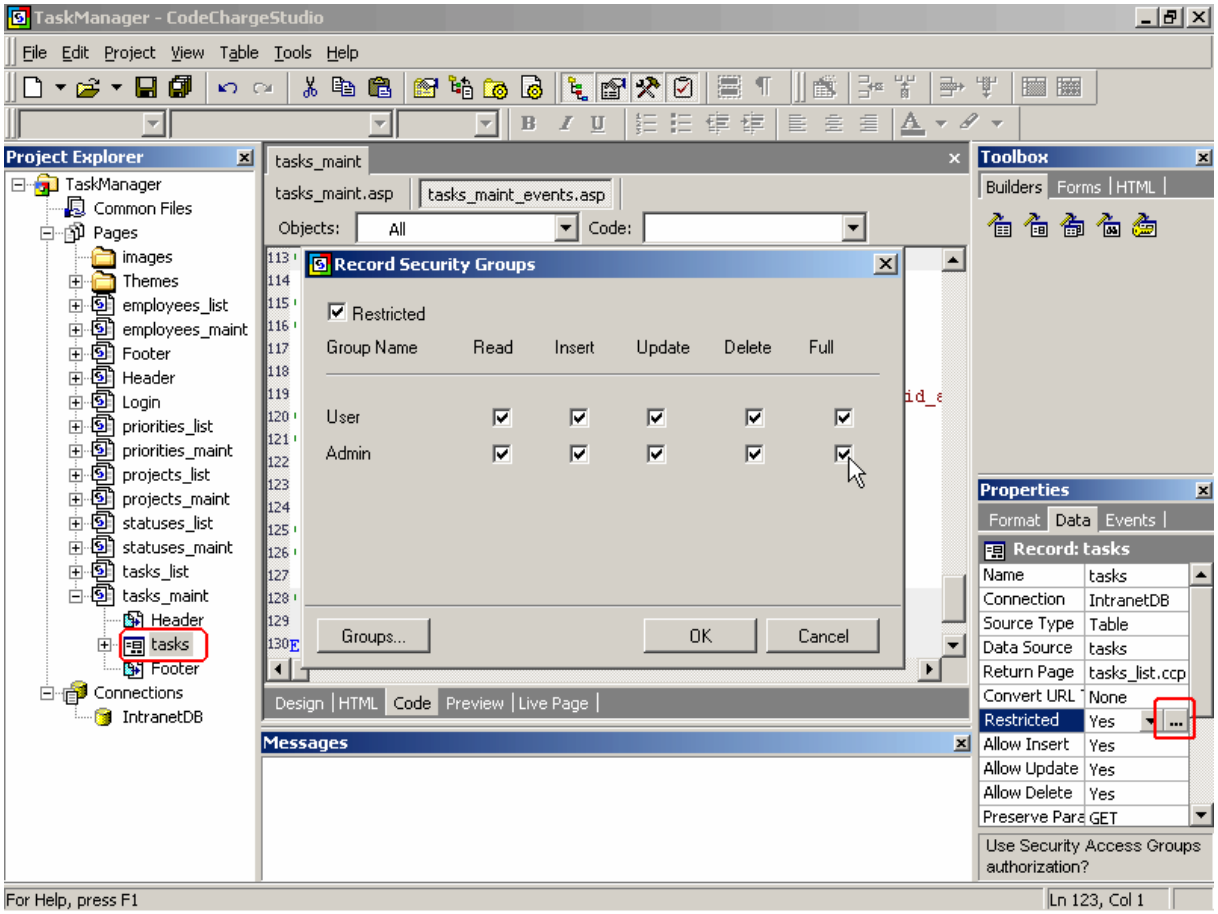
```
Dim current_task
current_task = CInt(Request.QueryString("task_id"))
if current_task <> 0 and CCGetUserID <> CCDLookUp("user_id_assign_to", "tasks", "task_id=" &
CCToSQL(current_task, "Integer"),  DBIntranetDB) then
'    tasks.Visible = False
    Redirect = "tasks_list.asp"
'    tasks.UpdateAllowed = False
'    tasks.DeleteAllowed = False
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 trying to modify a task that is not assigned to you.

3. **Disable Update/Submit and Delete buttons for unauthorized users.**
Comment out the “Redirect” statement and uncomment the two lines of code below it, as shown here:

```
Dim current_task
current_task = CInt(Request.QueryString("task_id"))
if current_task <> 0 and CCGetUserID <> CCDLookUp("user_id_assign_to", "tasks", "task_id=" &
CCToSQL(current_task, "Integer"),  DBIntranetDB) then
'    tasks.Visible = False
'    Redirect = "tasks_list.asp"
    tasks.UpdateAllowed = False
    tasks.DeleteAllowed = False
end if
```

This code shows how you can manipulate the *UpdateAllowed* and *DeleteAllowed* properties of a record form. These properties control the appearance of the “Update” and “Delete” buttons on the page. If set to False, the buttons will not appear. Additional security is implemented to make impossible to update the record even if someone saves the page, adds the missing buttons and submits the page externally. However, the above code won’t work unless you also set the “Restricted” property of your form to “Yes” and assign permissions to everyone. That’s because CodeCharge Studio generates the code appropriate for hiding and disabling buttons only when there is a need to do so, and restricting page access indicates that certain users are not allowed to add, update or delete records.



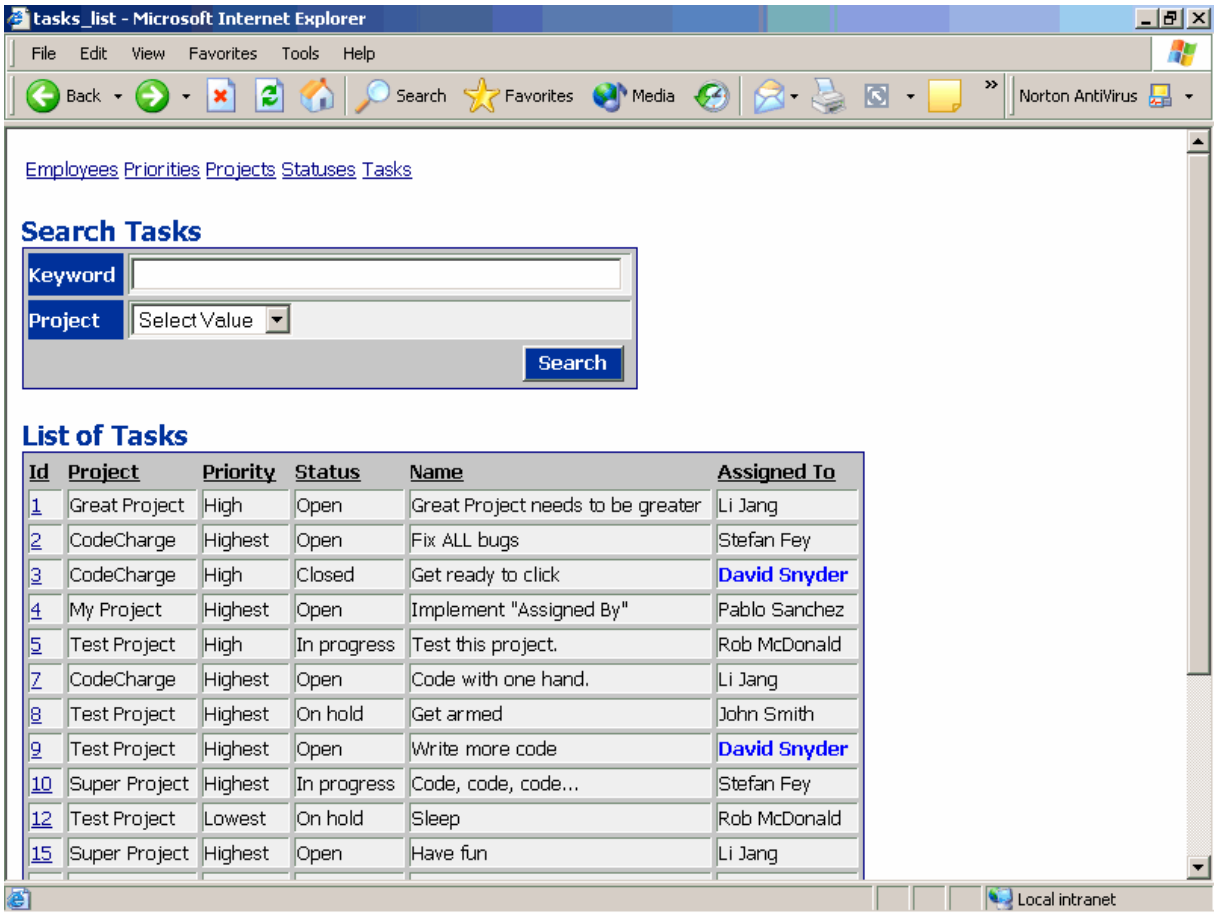
4. One more, although less secure method of disabling buttons on the record form is to hide the Update and Delete buttons on the page by adding the following custom code to the “Before Show” event of the form:

```
Dim current_task
current_task = CInt(Request.QueryString("task_id"))
if current_task <> 0 and CCGetUserID <> CCDLookUp("user_id_assign_to", "tasks", "task_id=" &
      CCToSQL(current_task, "Integer"),  DBIntranetDB) then
   tasks.Update.Visible = False
   tasks.Delete.Visible = False
end if
```

Conclusion

During the course this chapter of the tutorial, 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 User's Manual for more information.

Enjoy!



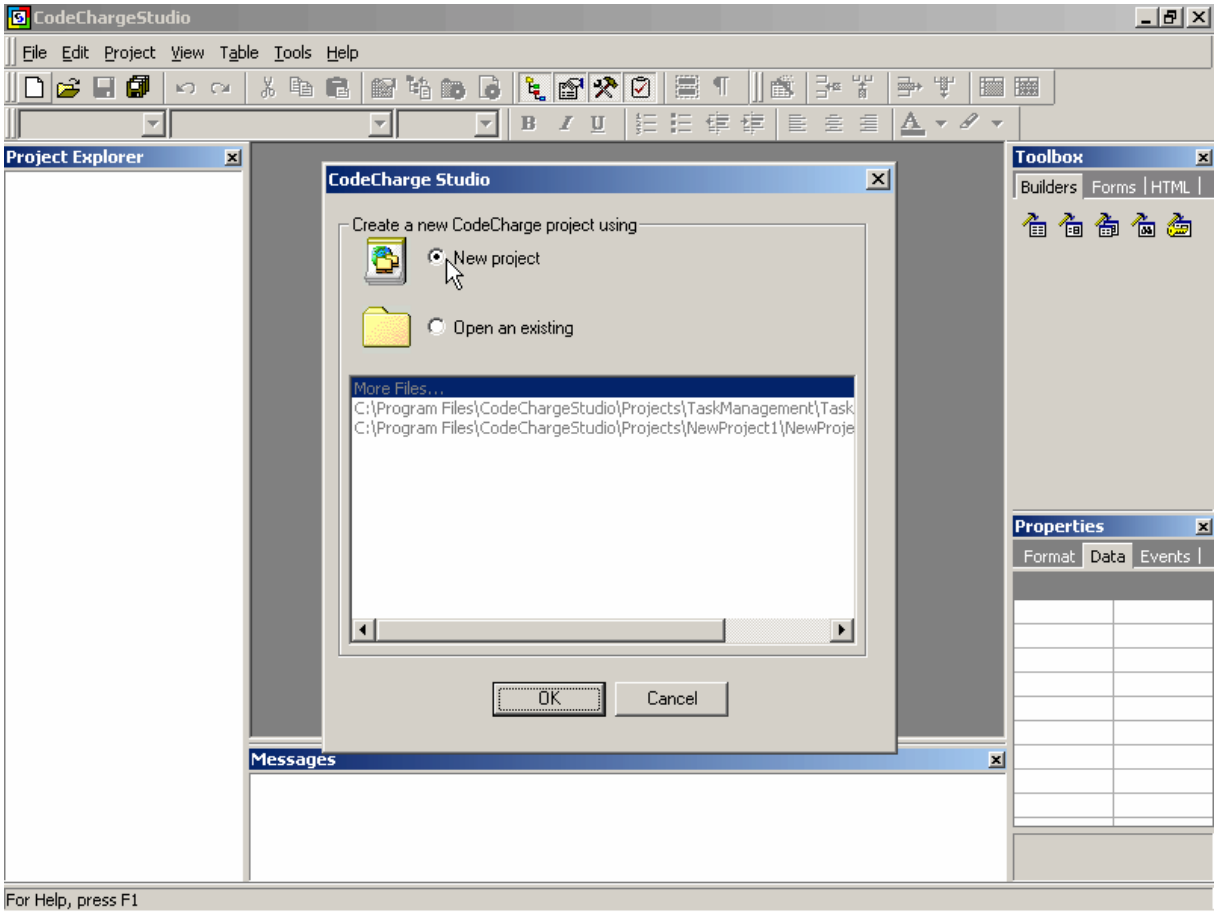
Chapter 2: Creating an Employee Directory

CodeCharge Studio provides all the tools you need to get started building your Web applications, including builders for creating search, grid and record maintenance forms. In this chapter, we will describe how you can utilize various builders, forms, controls and other features to create a basic employee directory from scratch. Since we will not be using the Application Builder, you will gain more insight into setting up project properties, creating database connections and working with other features that extend your application.

Creating a New Project

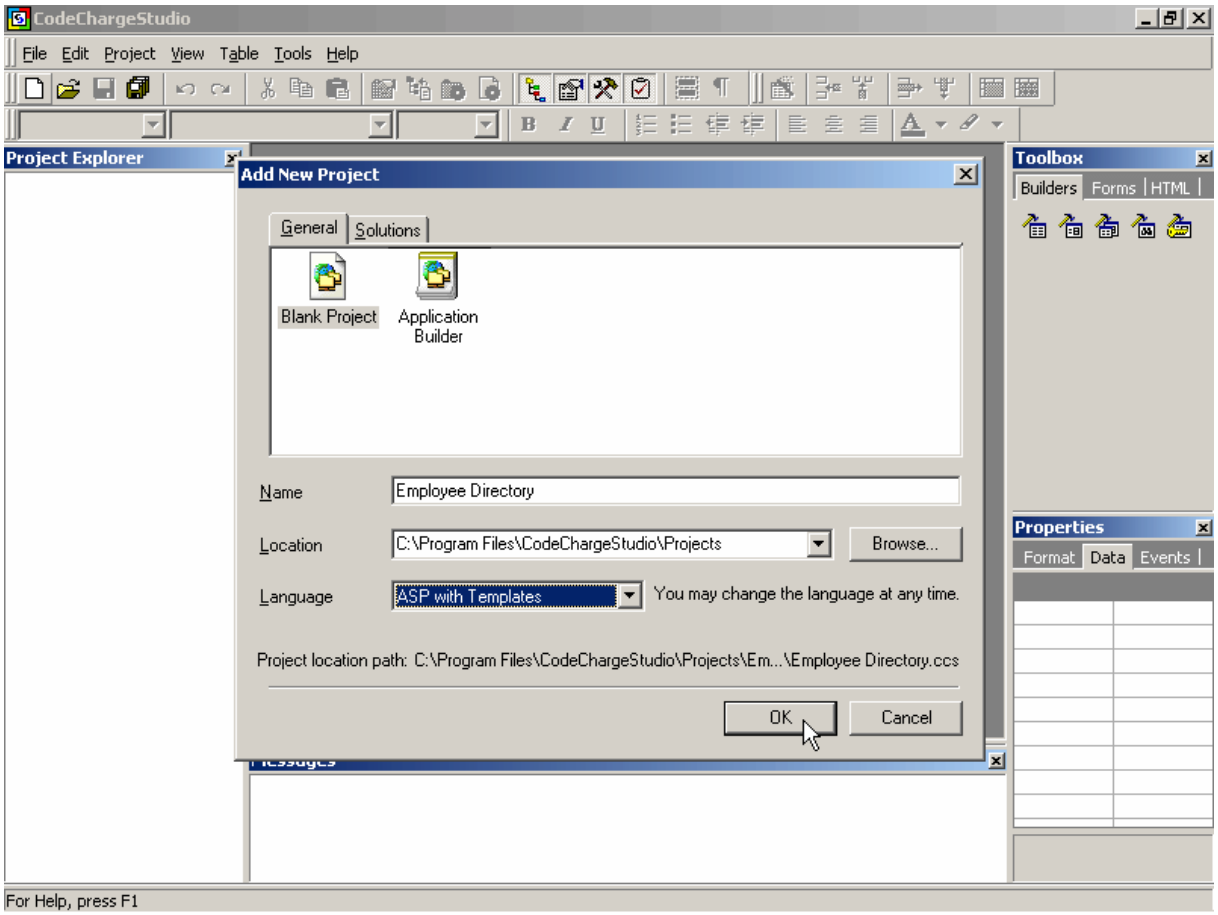
Create a New Project

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



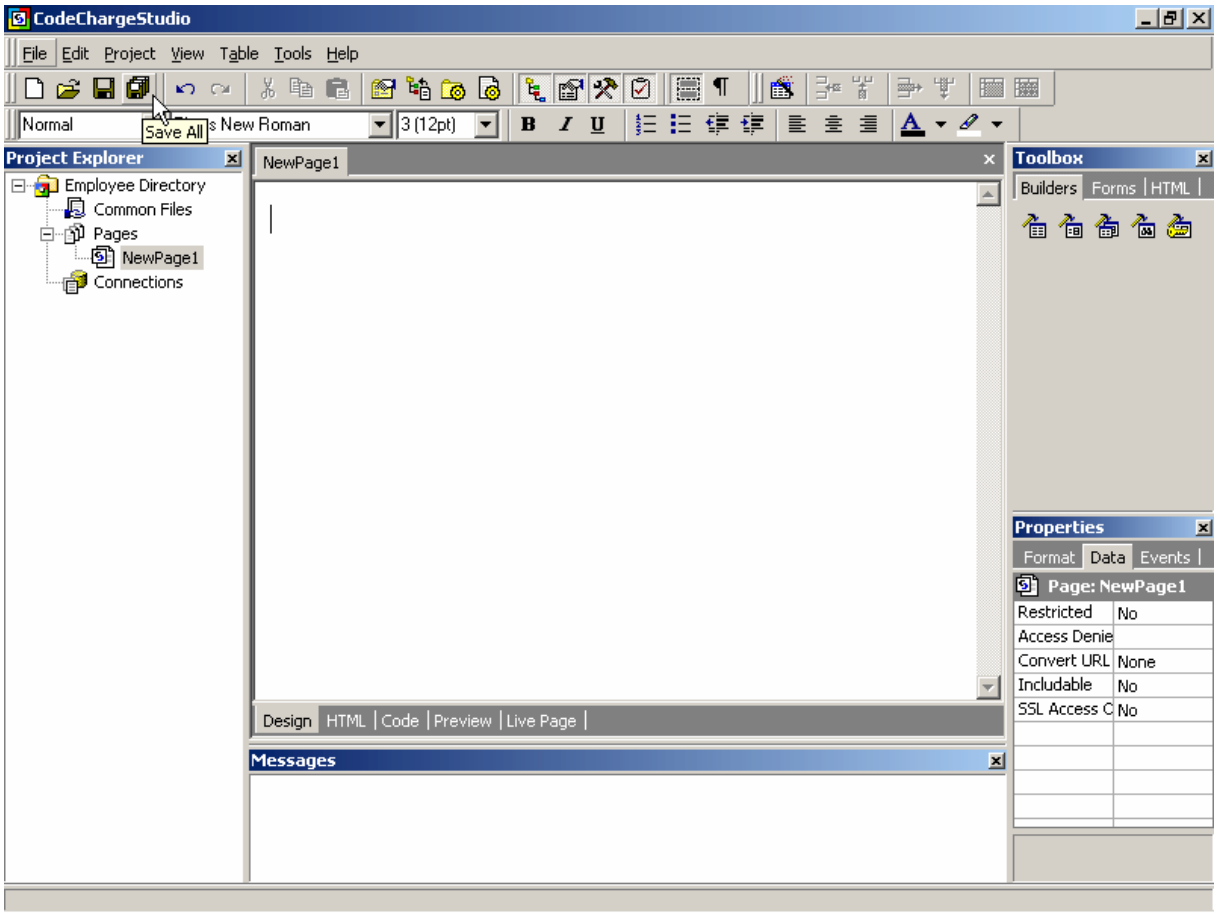
Create a “Blank Project”.

For the project name, enter: “Employee Directory”.
Also enter the Location where the project should be saved on the disk, then click the “OK” button to confirm and create a new 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 pres CTL+S to save the current page.

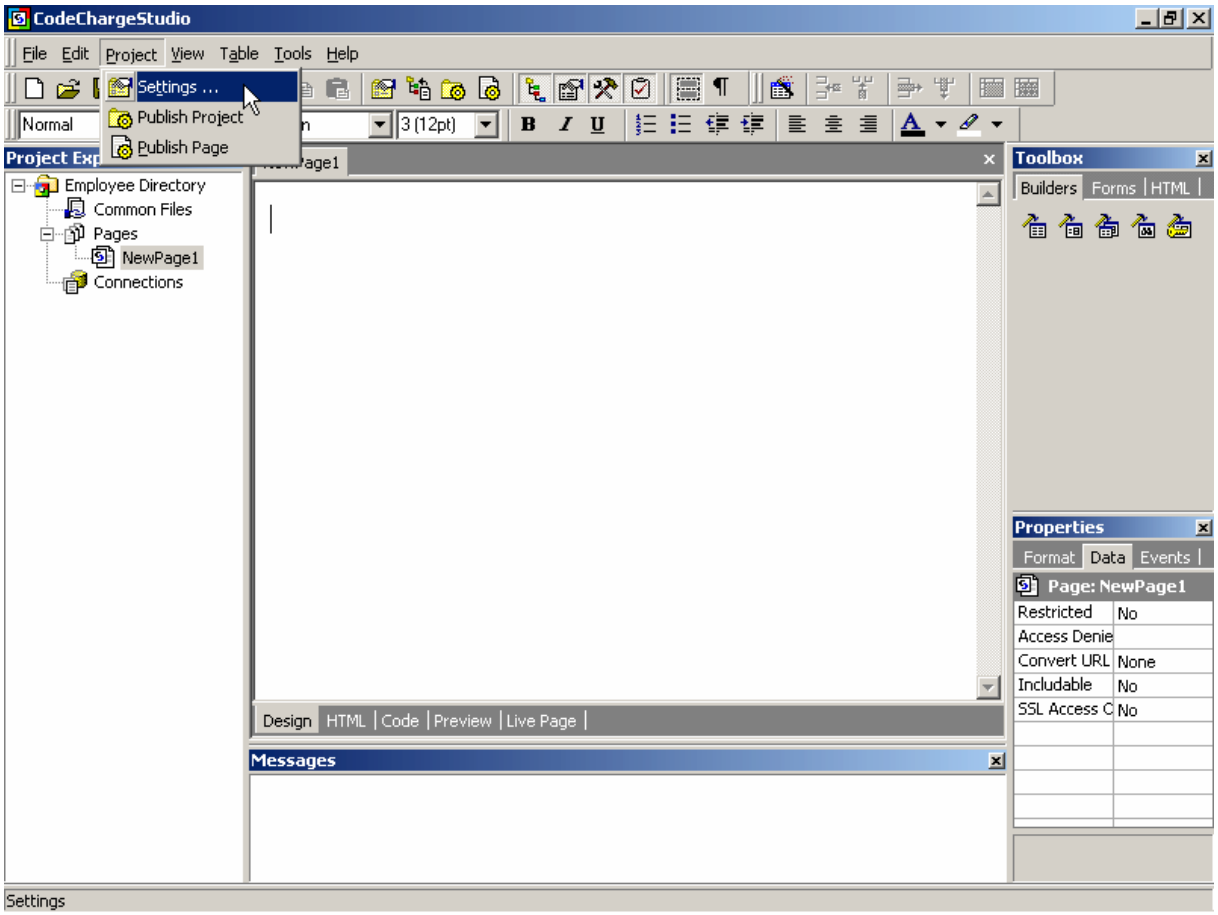


Specifying Project Settings

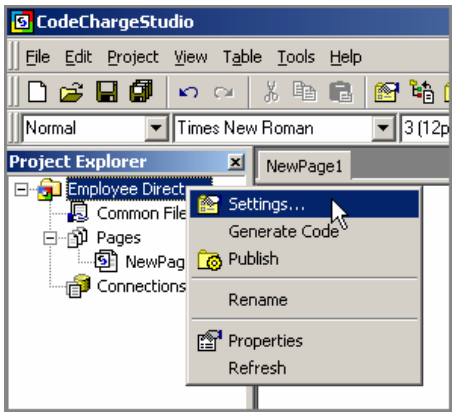
Project settings allow you to specify how to generate your web application and where to publish it. You can also specify the programming language, publishing directory, database connection, site authentication and various additional details concerning the project's configuration.

Open Project Settings

Select ***Project*** => ***Settings…*** on the main menu bar.

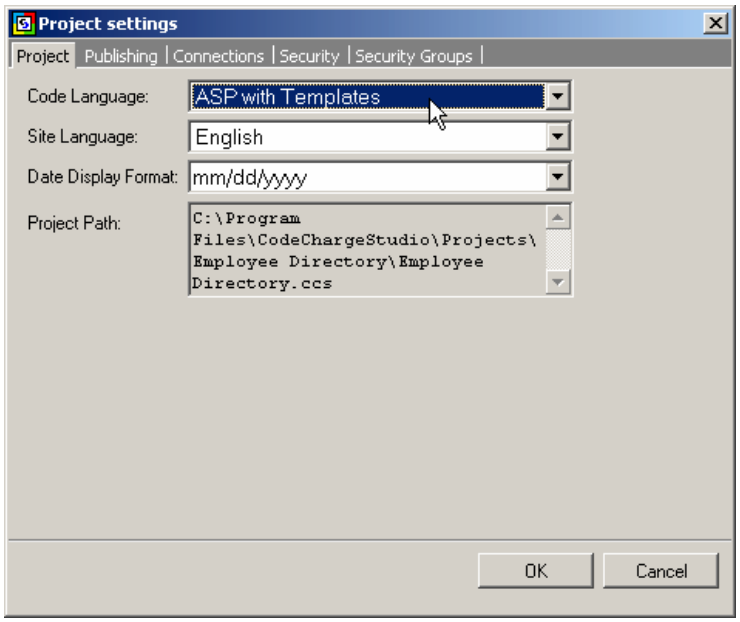


You can also right-click on the Project Name (“Employee Directory”) in the Project Explorer window and select the ***Settings…*** option.



Specify the General Project Settings

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



Code Language:
The currently generated programming languages are:

- ***ASP 3.0 with Templates*** – generates ASP 3.0 files that uses separate .html files as templates during run-time.
- ***ASP.Net 1.0 C#*** - generates .aspx files with C# 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 1.2***. – generates JSP 1.1 code.
- ***PHP 4.0 with Templates*** – generates PHP 4.0 code (.php) and separate .html template files.

Servlets 2.2 JDK 1.2 with Templates – generates Java code that utilizes .html templates.

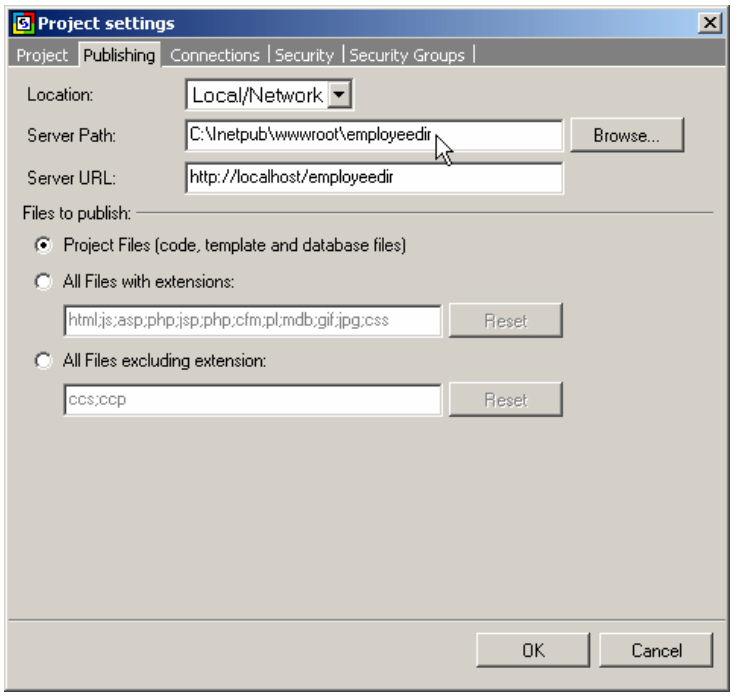
Site Language:
Specify the spoken language to use when generating text messages for the site. For example the text “No records” that appears when no more records are to be displayed within a grid, could be generated in any one of the supported languages.

Date Display Format:

This is the default format for the date fields within the project, for example if you display the employee hiring date on a page.

Enter the Publishing Settings

Specify the folder where CodeCharge Studio should output generated files during the publishing process.



Location:
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.

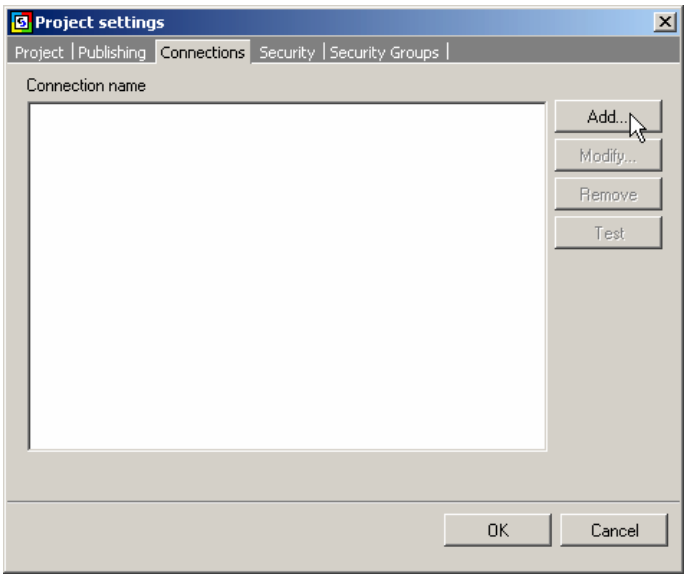
Server URL:
The web address corresponding to the **Server Path**. This URL will be used to view the page in Live Page mode.

Files to publish:
Leave the default option unless you want to publish specifics files or exclude others.

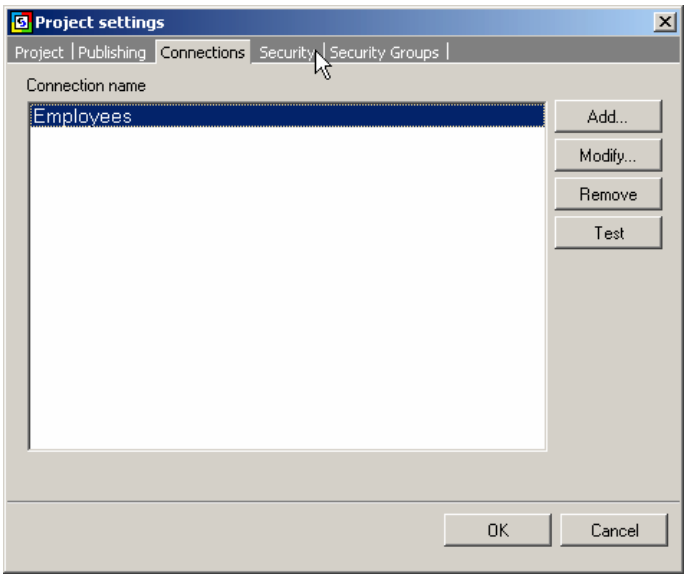
Create Database Connection(s) for the Project

Click on the “Connections” tab to setup a new database connection.

Click “Add…” and follow the steps described in the Create Database section to complete creating a database connection(s).



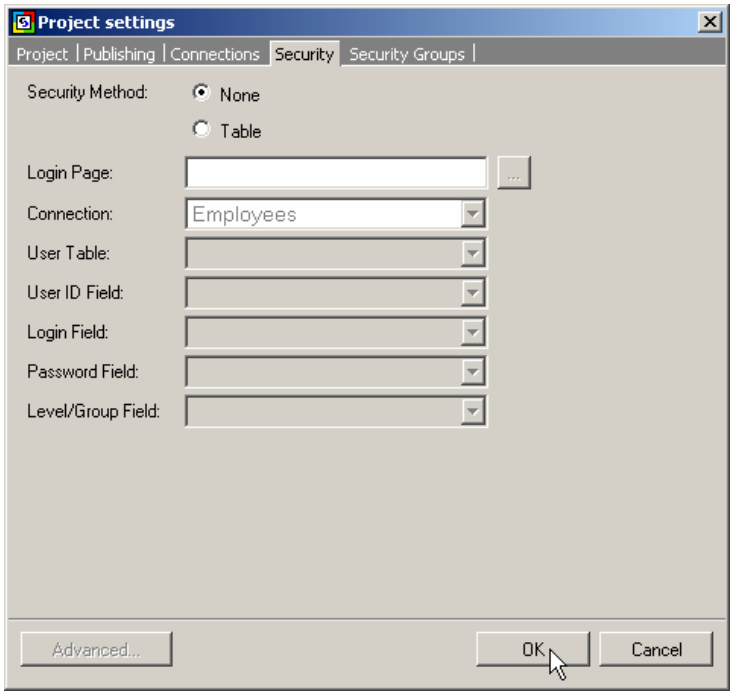
Once completed building the connection, click on the “Security” tab.



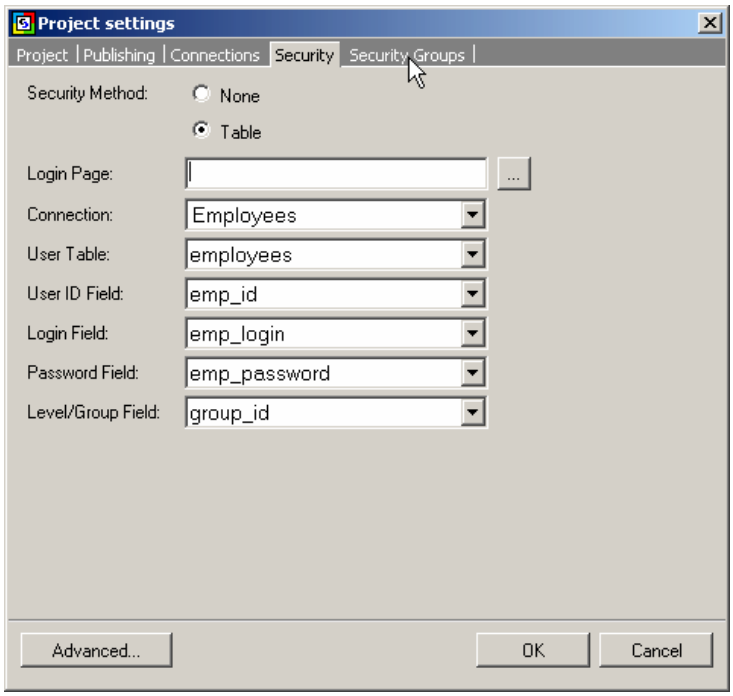
Setup Security Settings for the Project

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 and 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.



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.

User ID Stored As:
The method of storing user IDs, which could be a cookie or session.
URL authentication is also available, which converts all URLs and appends a special URL parameter that tracks users.

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.

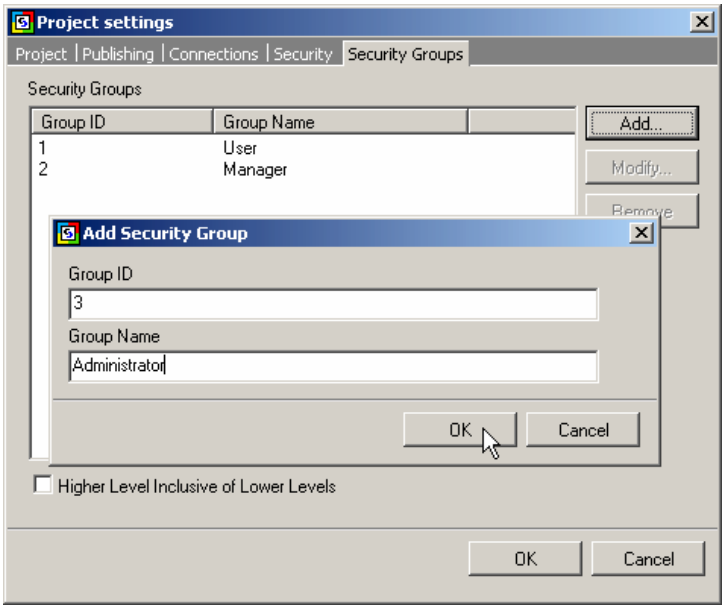
Login Field:
The field in the user table that contains the user's login name.

Password Field:
The field in the user table that contains the user's password.

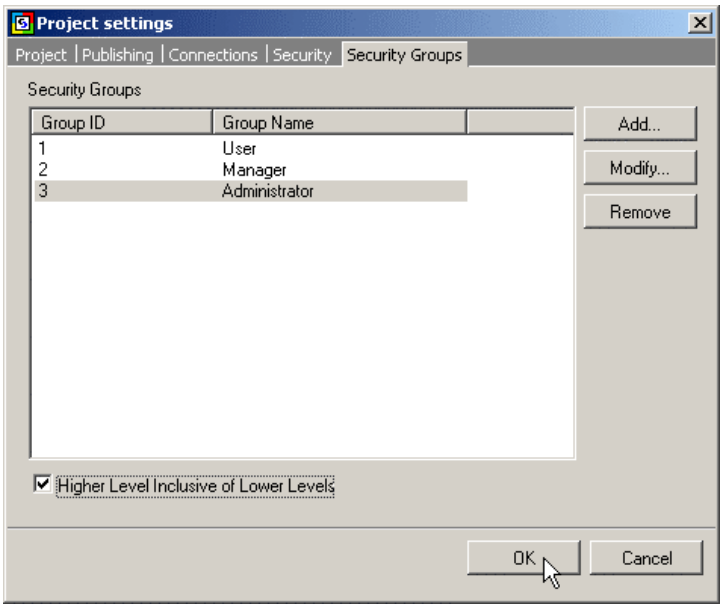
Level/Group Field:
The field containing user's security level or group, which will be used to verify access authorization.
In addition, levels or groups should be configured under the "Security Groups" tab.

Configure Security Groups for the Project

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.



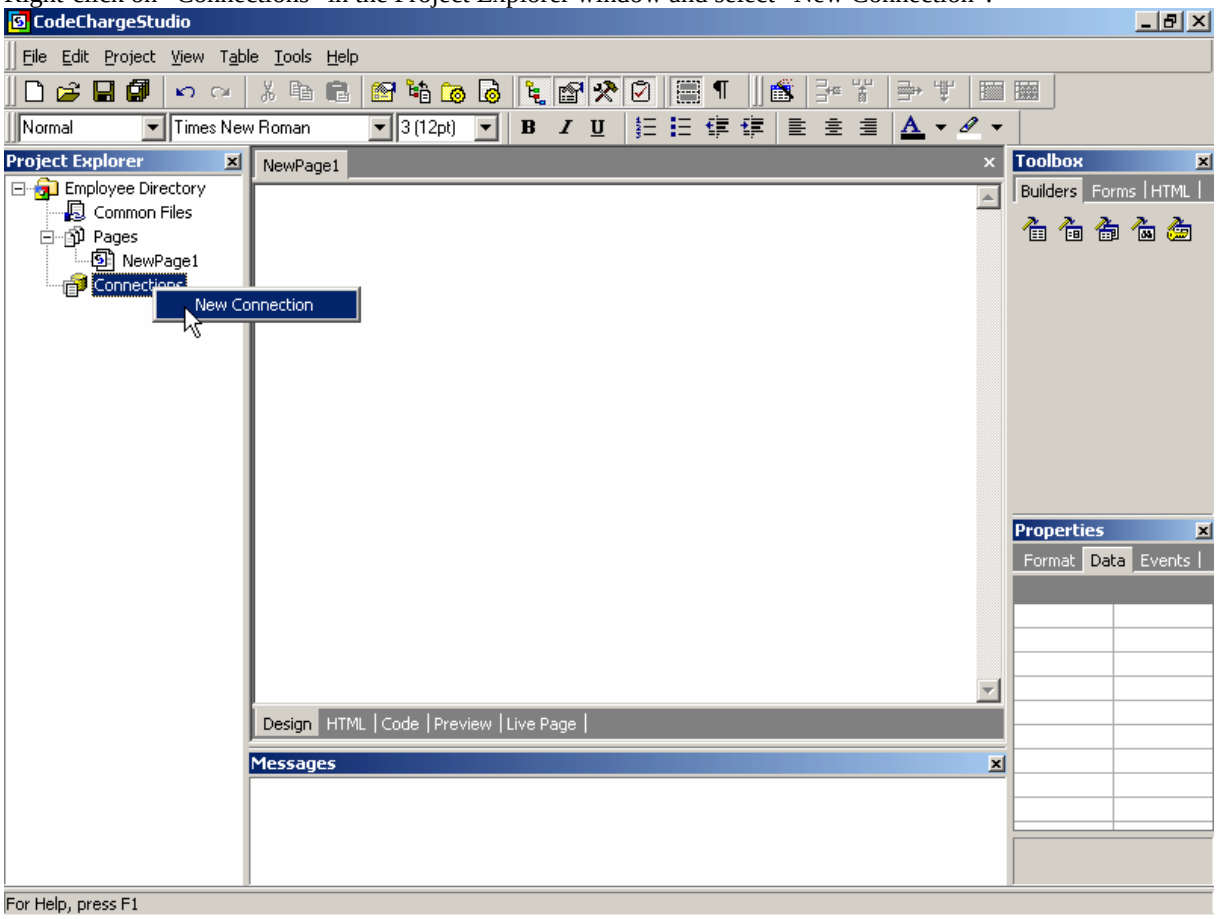
If you use numbers as your Group IDs, you can select the option “Higher Level Inclusive of Lower Levels”, which will cause the generated programs to assume that users with a higher security level can access pages with lower security levels. For example, users with level 4 will be able to access pages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



Creating a Database Connection

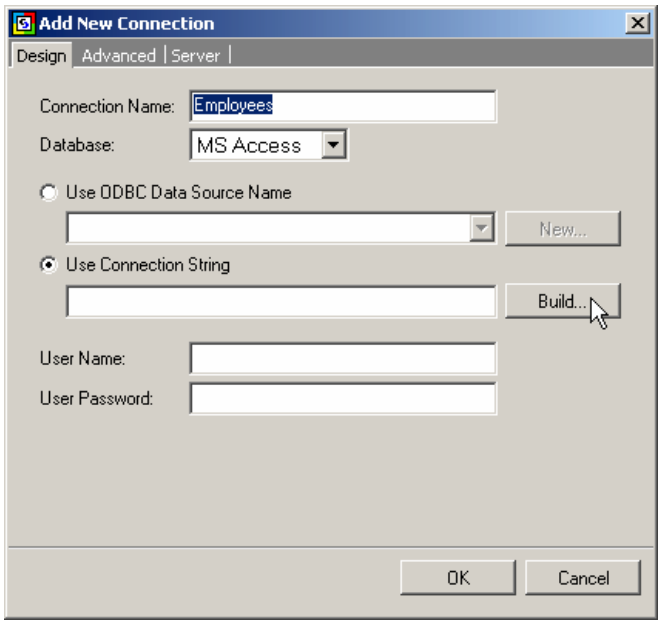
Create an Initial Database Connection

Right-click on “Connections” in the Project Explorer window and select “New Connection”.



Build the Design Connection

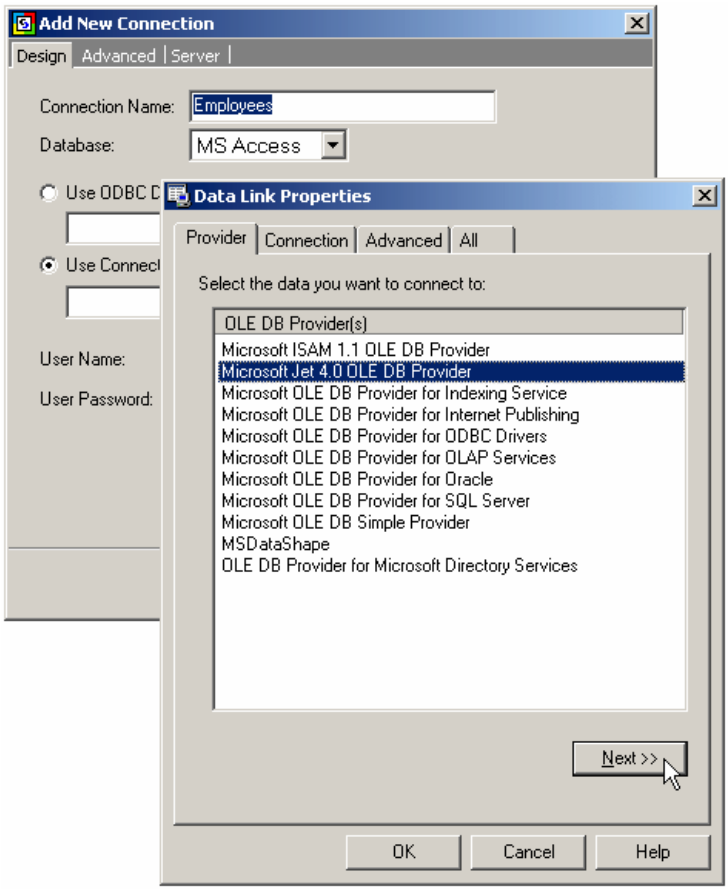
Enter a name for your connection, then click “Build…” to specify connection parameters.



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.

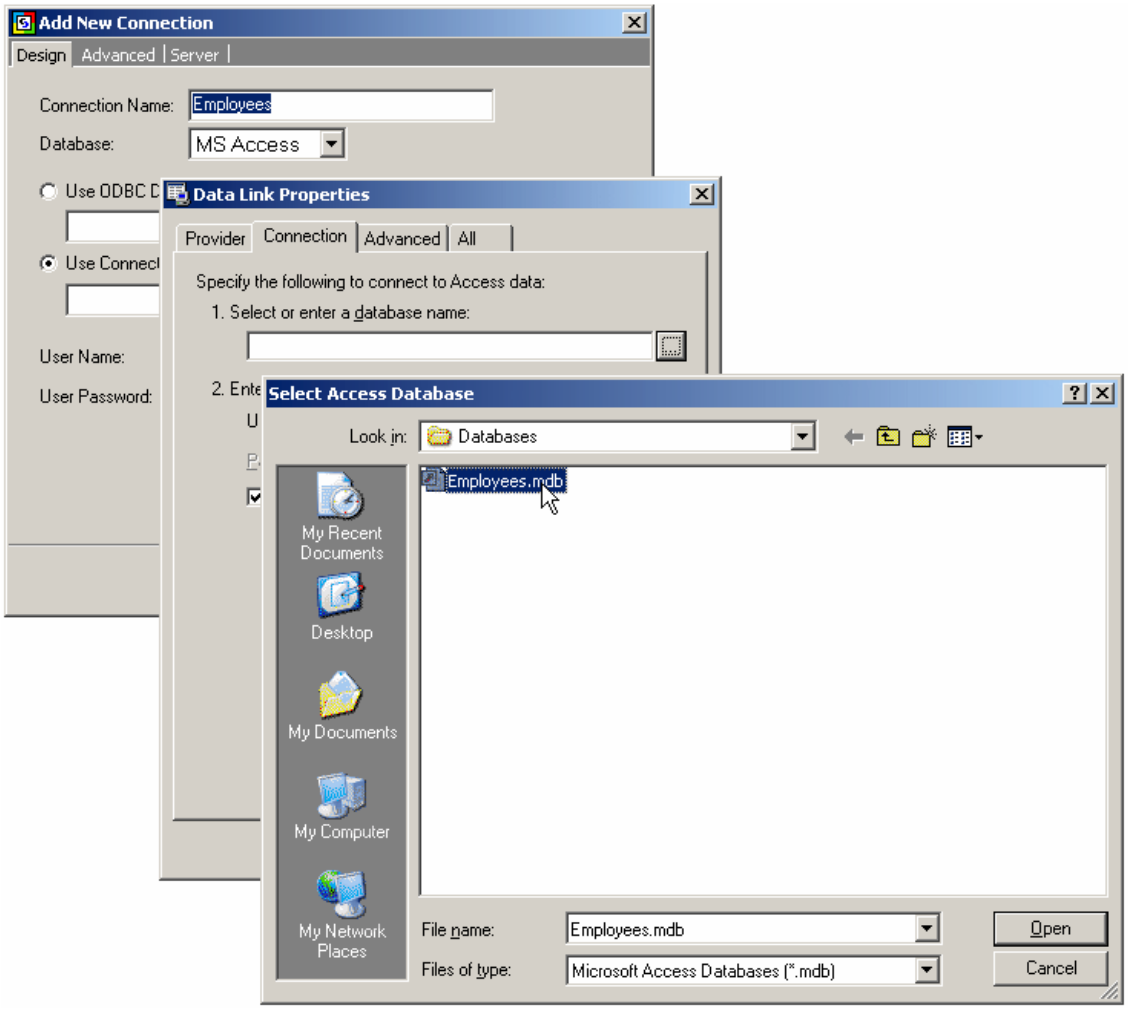
Specify the Data Provider (JET, ODBC, etc.)

In the “Data Link Properties” window, select Jet 4.0 as the Provider. This will allow you to connect directly to a database file, such as MS Access .mdb. 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.



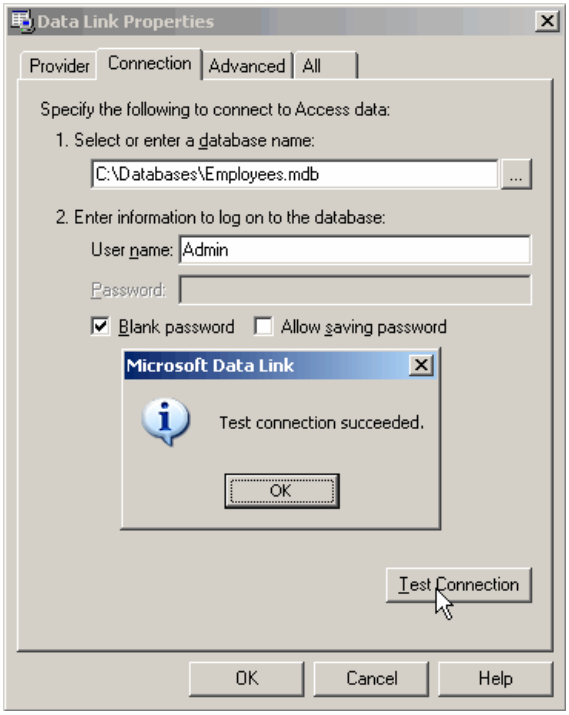
Specify Connection Parameters (Database Filename)

If using the JET provider, select the database file to be used for this connection.
If using ODBC or other provider, select the DSN (Data Source Name) or other parameters needed to establish the database connection.



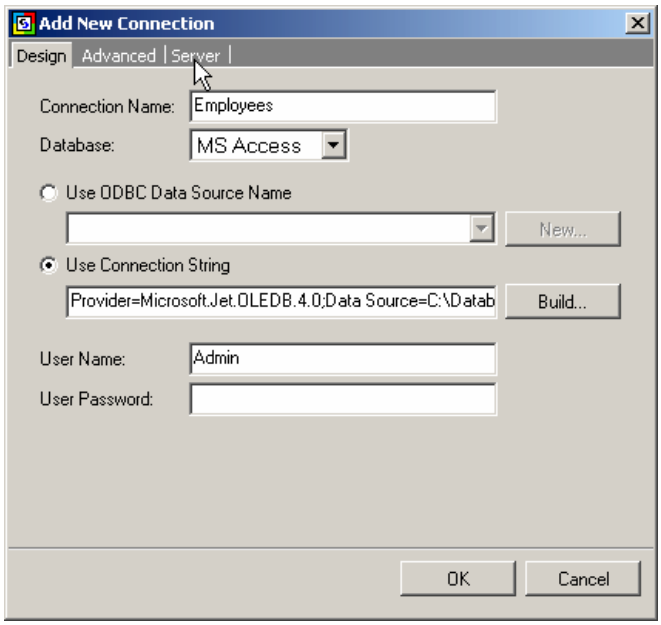
Test the Database Connection

Click “Test Connection” to check if the connection to the database is working correctly.



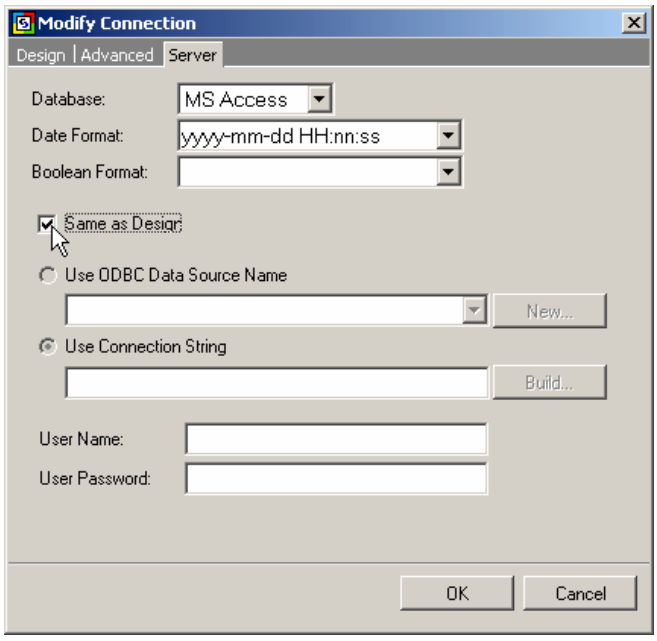
Complete the Build Process of the Design Connection

Confirm that the connection string was created in the “Use Connection String” field, then click on the “Server” tab to create the server connection.



Setup the Server Connection

Specify that the Server connection is the same as the design connection.



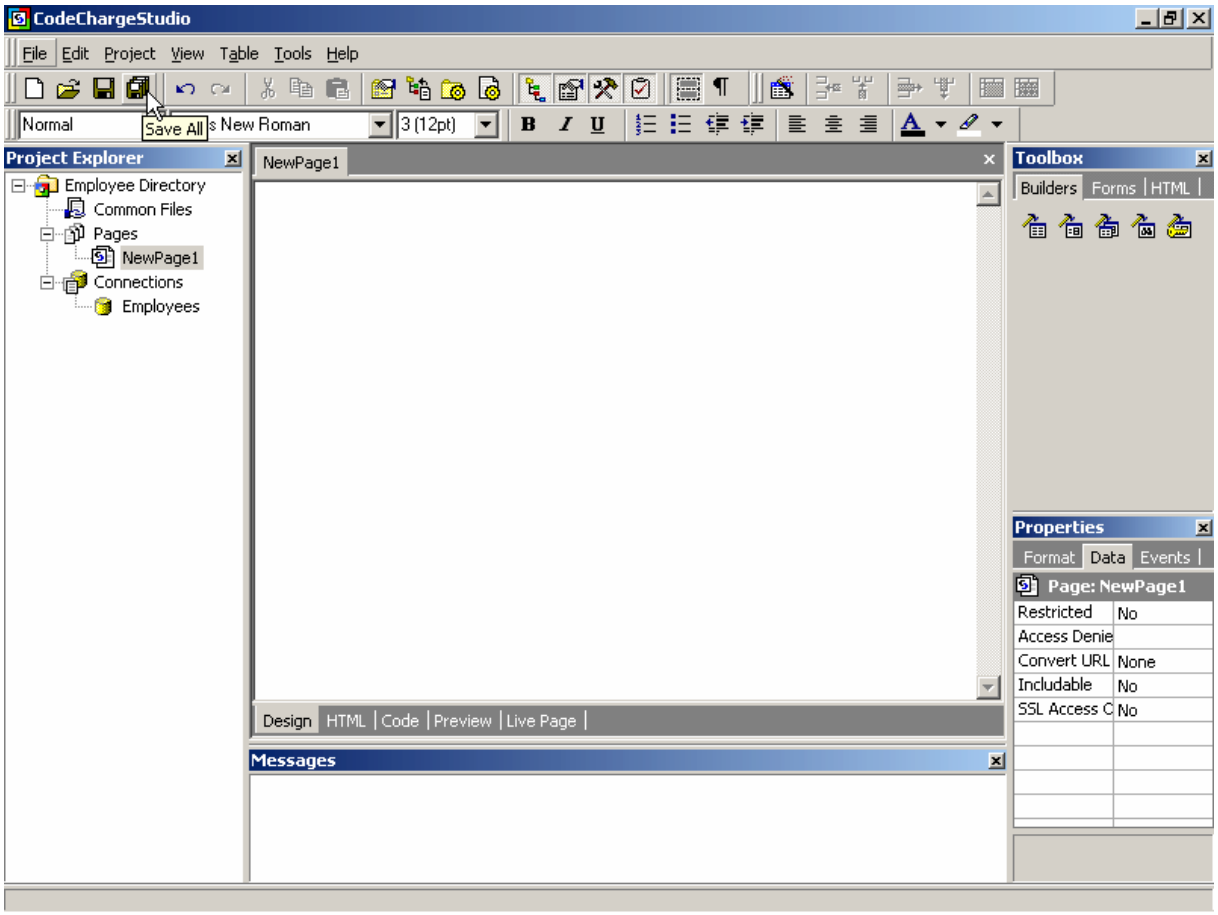
The Server Connection is the database connection utilized by the generated pages to retrieve and update the data. This connection can be different from the Design Connection that is used by the CodeCharge Studio GUI.

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.

Save Project Settings

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

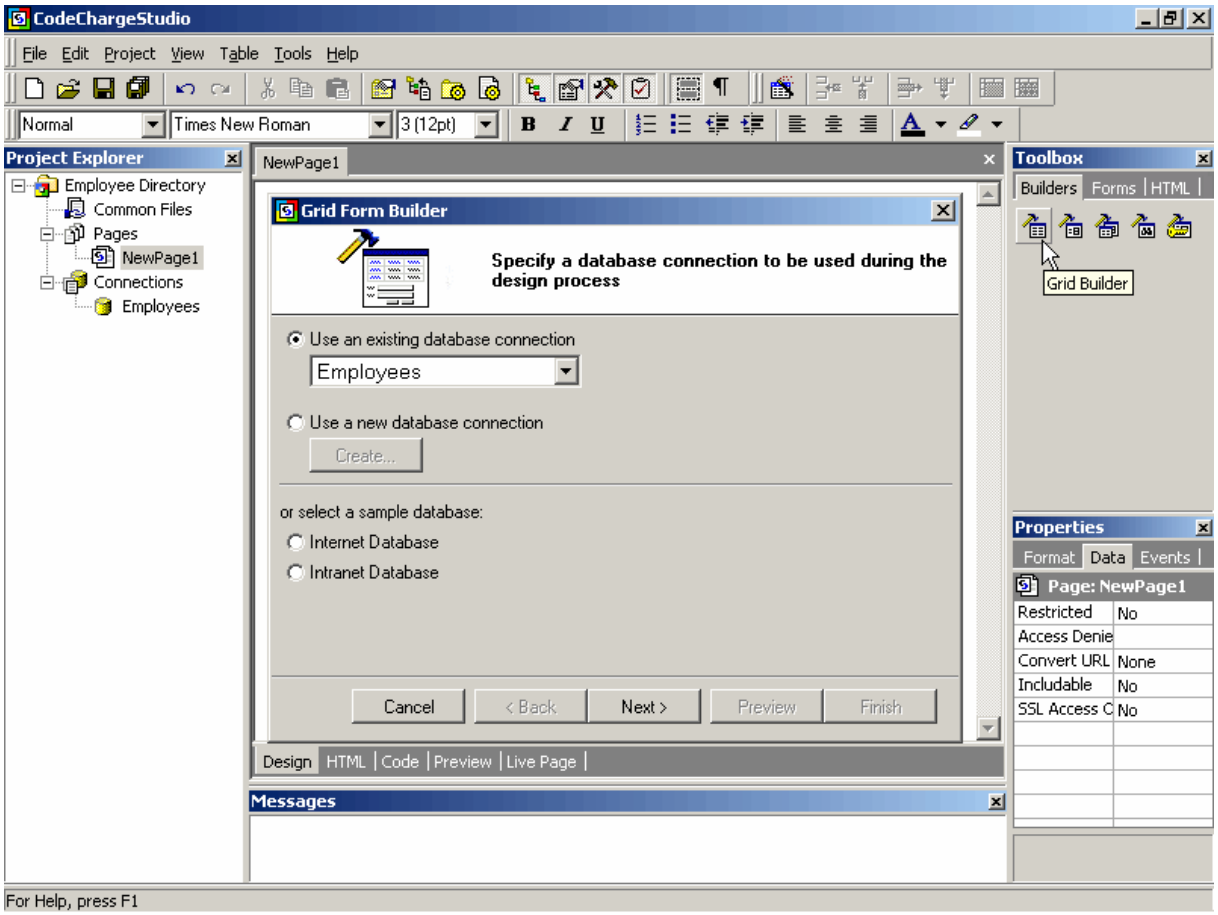


Creating a Grid using the Grid Builder

The Grid Builder creates a grid that displays multiple database records for viewing.

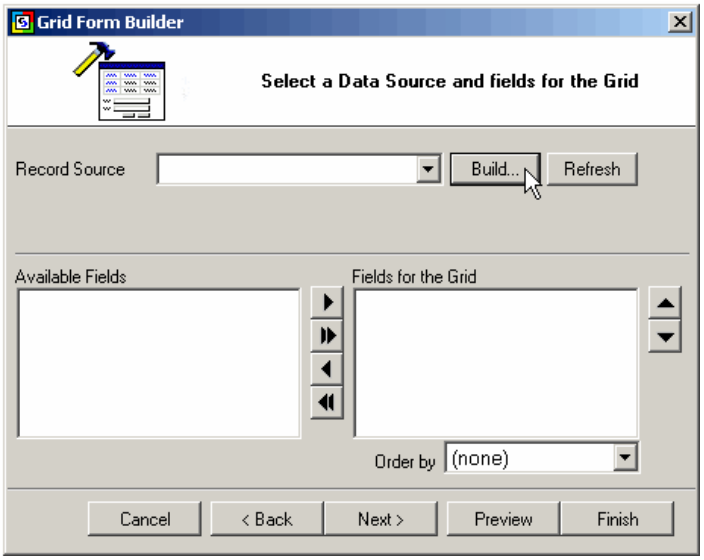
Launch the Grid Builder

Click the **Grid Builder** icon in the Toolbox to start the Grid Builder.
Select an existing database connection or create a new one, then click “Next”.
You can also select one of the example databases included with CodeCharge Studio: Internet or Intranet.



Launch the Visual Query Builder

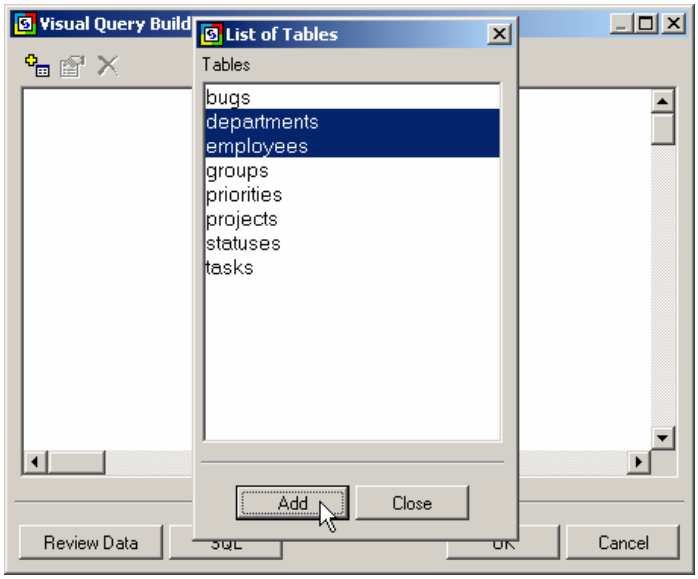
To aid you in the process of selecting database tables and fields to be used in the grid, CodeCharge Studio has a visual Query Builder. Click “Build…” to access it.



Select the following tables to be used as the data source for the grid:

departments
employees

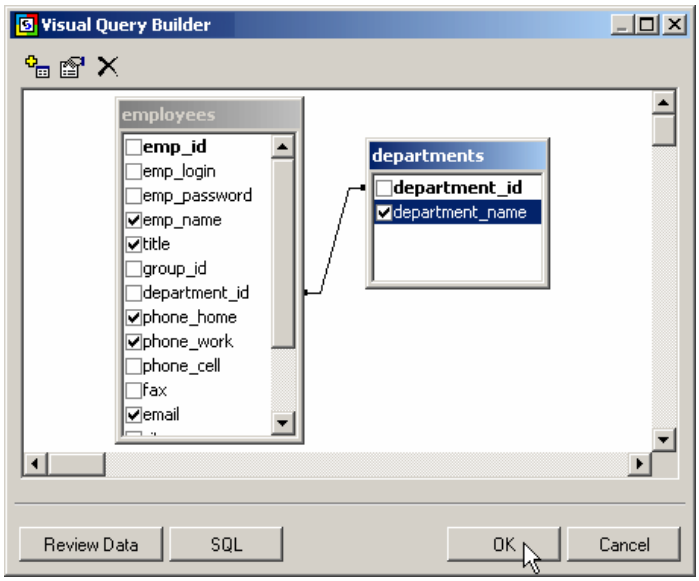
Click “Add” to place the tables in the Query Builder.



Specify Database Fields in the Visual Query Builder

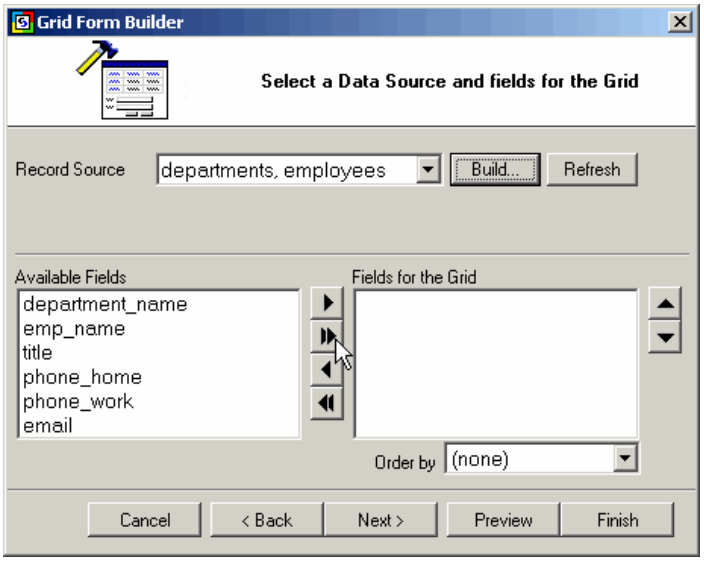
Set the checkboxes next to all database fields that you would like to include in your grid.
For this tutorial, select the following fields:

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

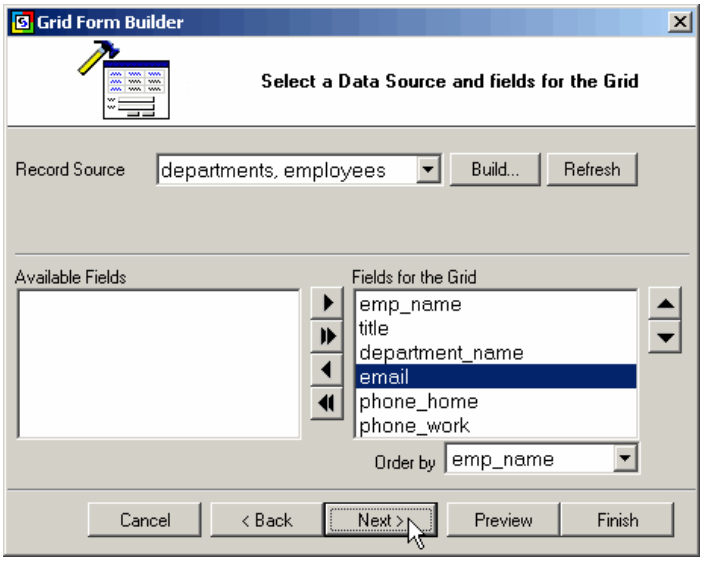


Select Database Fields for the Grid Data Source

Once finished using the Visual Query Builder, the Grid Builder will display all database fields available for inclusion in the grid. Click on the double right arrow (>>) to include all fields into the grid.



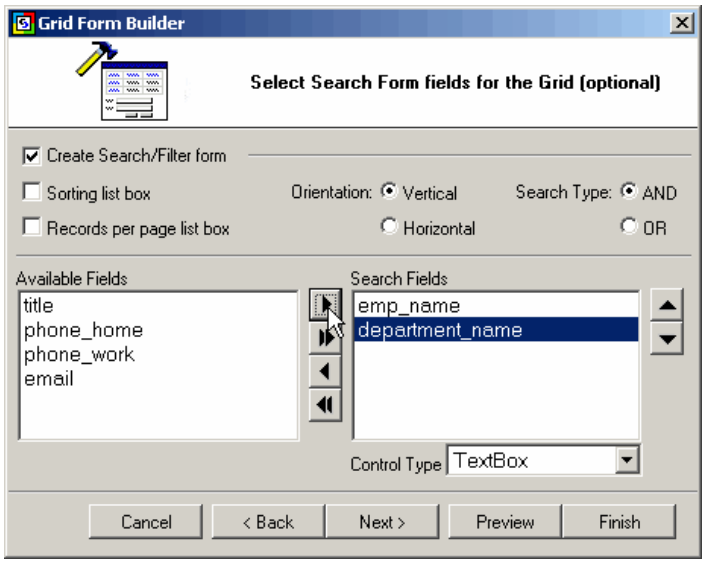
Click on the up and down arrows to move fields and specify the order in which they will appear in the grid.
Click on the "Order by" drop-down menu and select the field that will be used as the basis of the sort order for the grid. For example, if you select the field "emp_*name*", by default the grid will be sorted by employee name.



Click "Next" when finished.

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.
Activate the “Create Search/Filter form” checkbox then specify fields to be included in it.



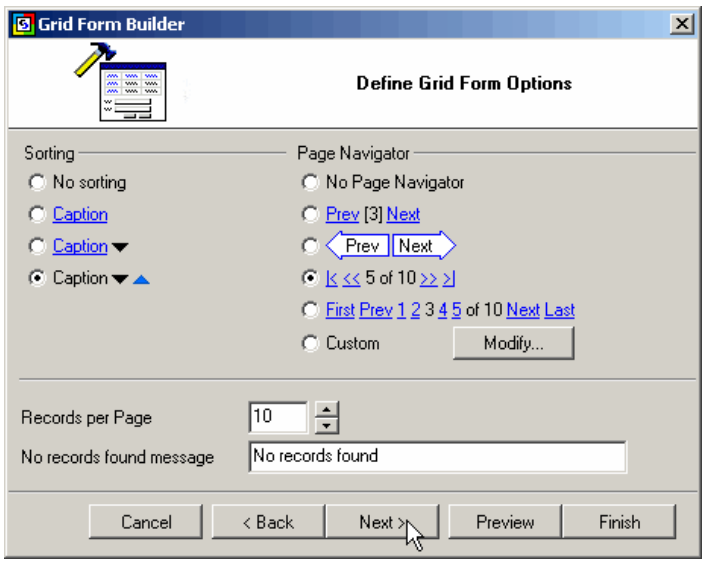
For each field you can specify its “Control Type” by selecting TextBox, ListBox or other control that will be used to represent this field.
We will use one TextBox field (*emp_name*) for keyword search and one ListBox field (*department_name)* for specifying the department when searching/filtering the data.
ListBoxes require additional configuration therefore let’s select both *emp_name* and *department_name* fields as TextBox for now (in the Control Type field) and discuss this topic later.

Click “Next” when finished configuring the screen as shown above.

Define Grid Sorting and Navigation

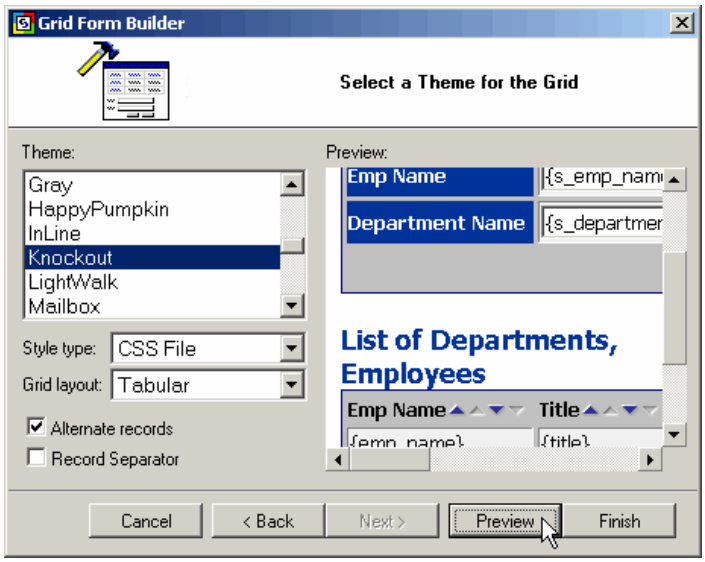
Specify if users can sort the Grid by clicking on the column headings and if users can navigate the grid by clicking on page numbers or *First/Last/Previous/Next* page links.

You can also specify if sorting and navigation should be represented by graphical icons or plain text. Additionally, 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.



Select a Theme for the Grid

Select one of the available themes for the grid and specify other options such as the Style Type and Grid Layout.



Style Type:
This specifies the type of style to be used on the page. This can be either HTML itself or CSS (Cascading Style Sheets). If you choose to use CSS, specify if you want to generate a single CSS file for all pages (CSS File), one CSS file for each page (CSS page) or have CSS embedded within the HTML (CSS Inline).

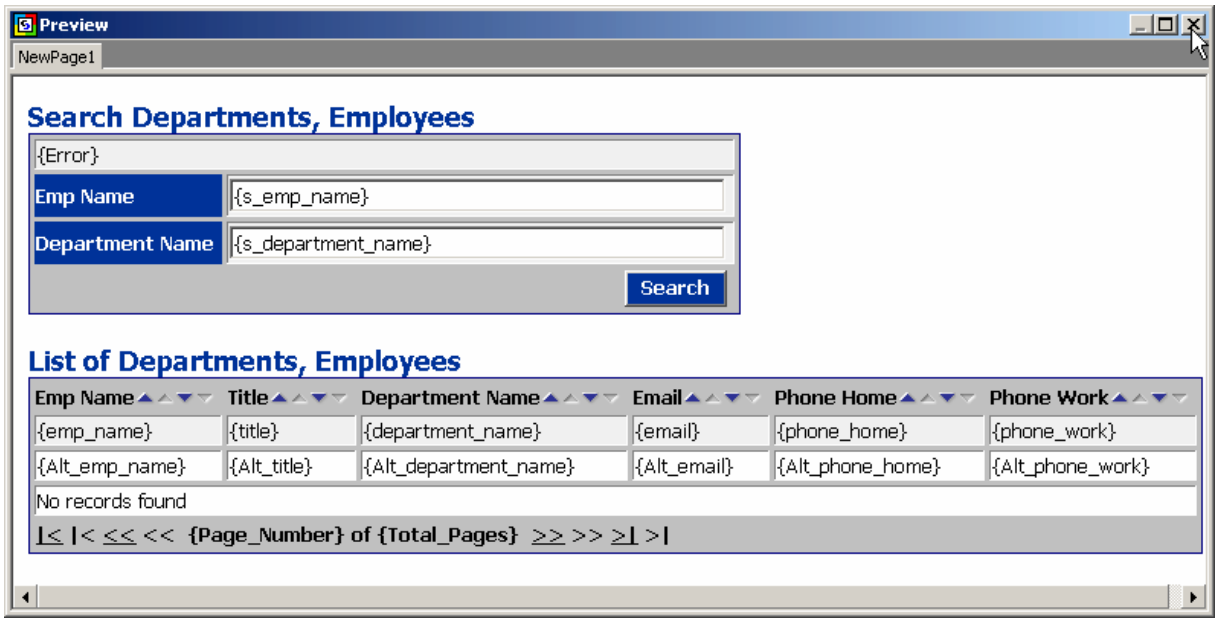
Grid Layout:
Tabular – standard grid that looks like a table or spreadsheet
Columnar – alternative grid type for newspaper/column type list of records
Justified – another alternative grid type

Alternate Records:
Specify whether even grid rows should have a different shade/background than odd rows.

Record Separator:
Specify whether grid records should be generated as separate tables. This allows for additional flexibility in how the grid looks. For example, you can use the space between records to display additional information.

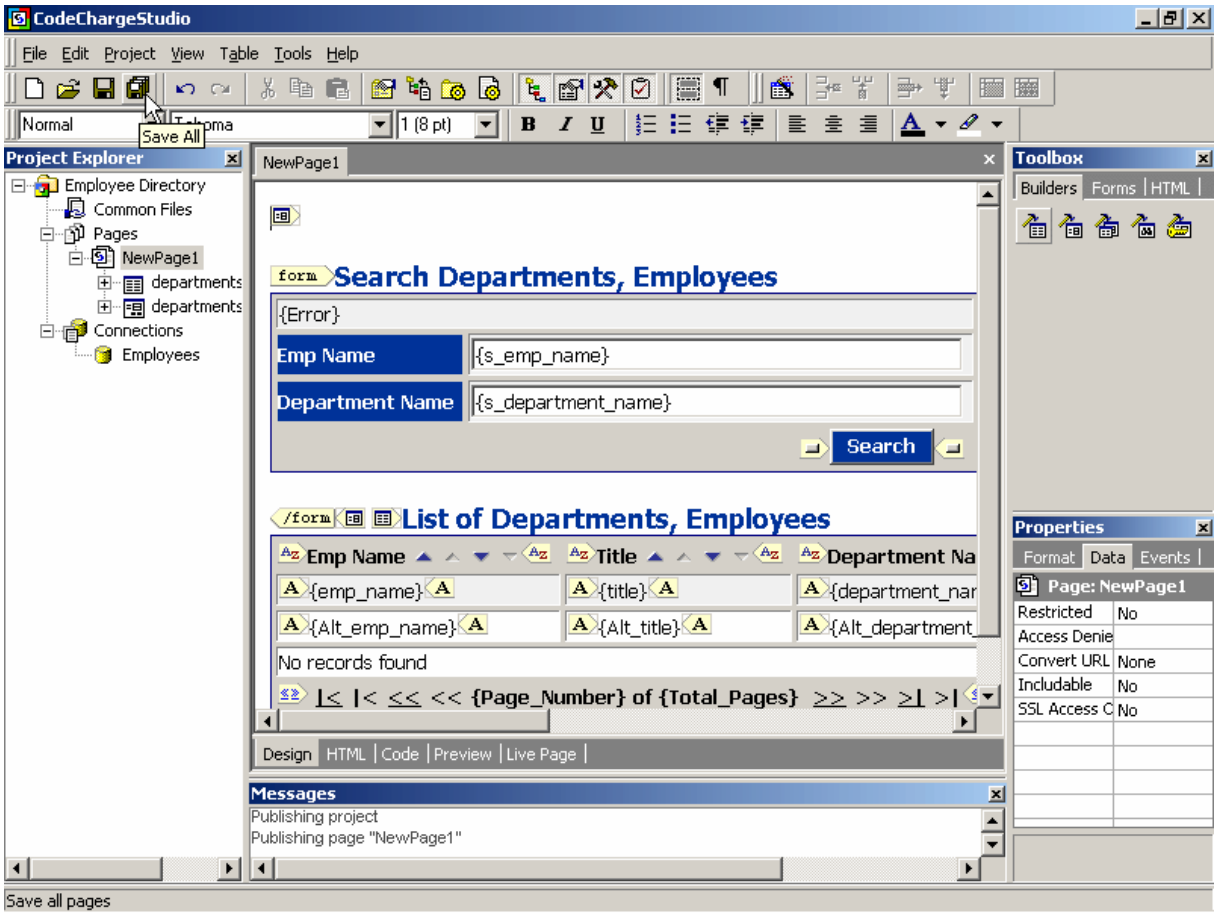
Preview the Grid

Click the **Preview** button to see the draft HTML generated by the Grid Builder.



Save the Project

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

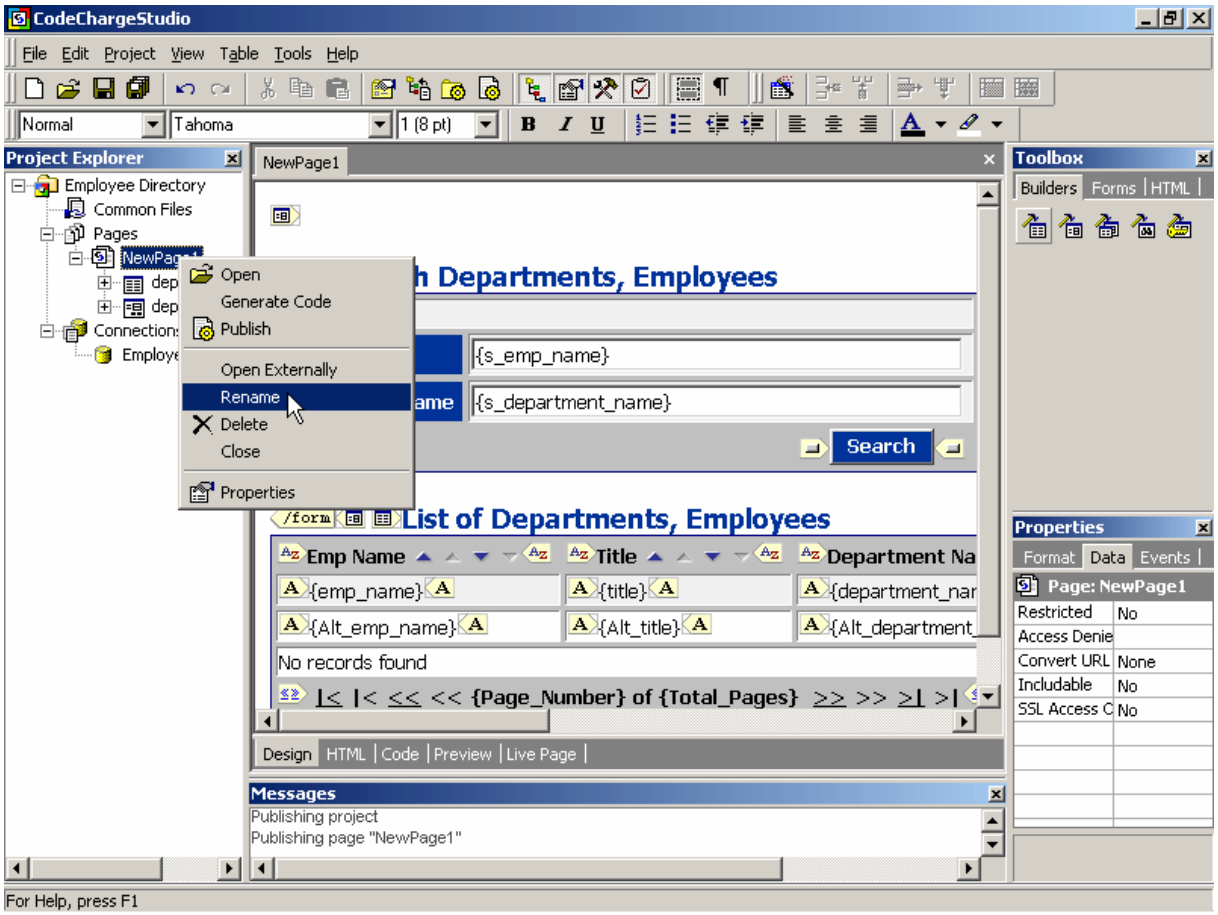


Finalizing the Search and Grid Forms created with the Builder

Builders allow you to quickly add components to your page although often you will still need to manually finish configuring some of the controls or extending the application's functionality. In this section, we will perform additional tasks to complete the Employee Directory.

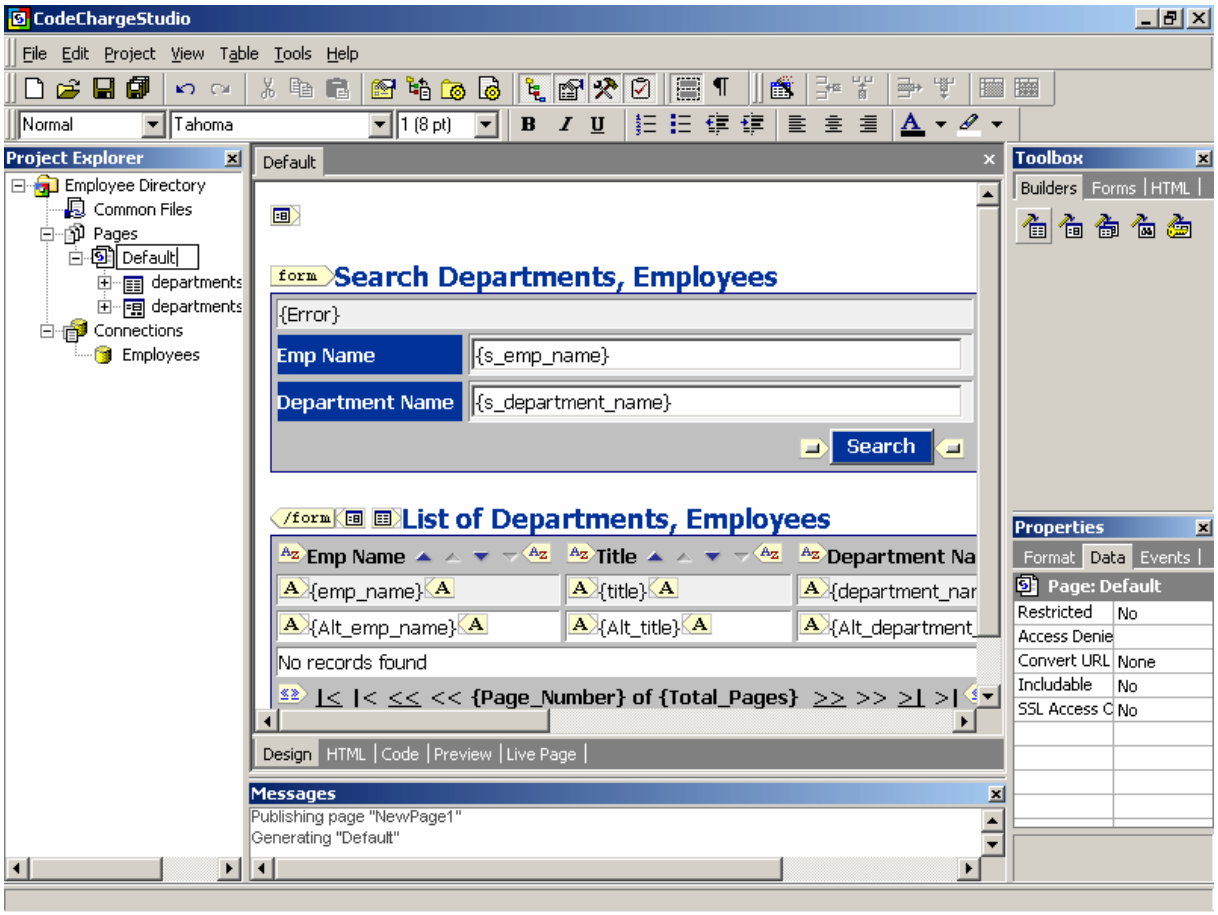
Rename the Page

First let's change the name of the page to a more appropriate name.
Right click on the current page name in the Project Explorer window then select the "Rename" option.



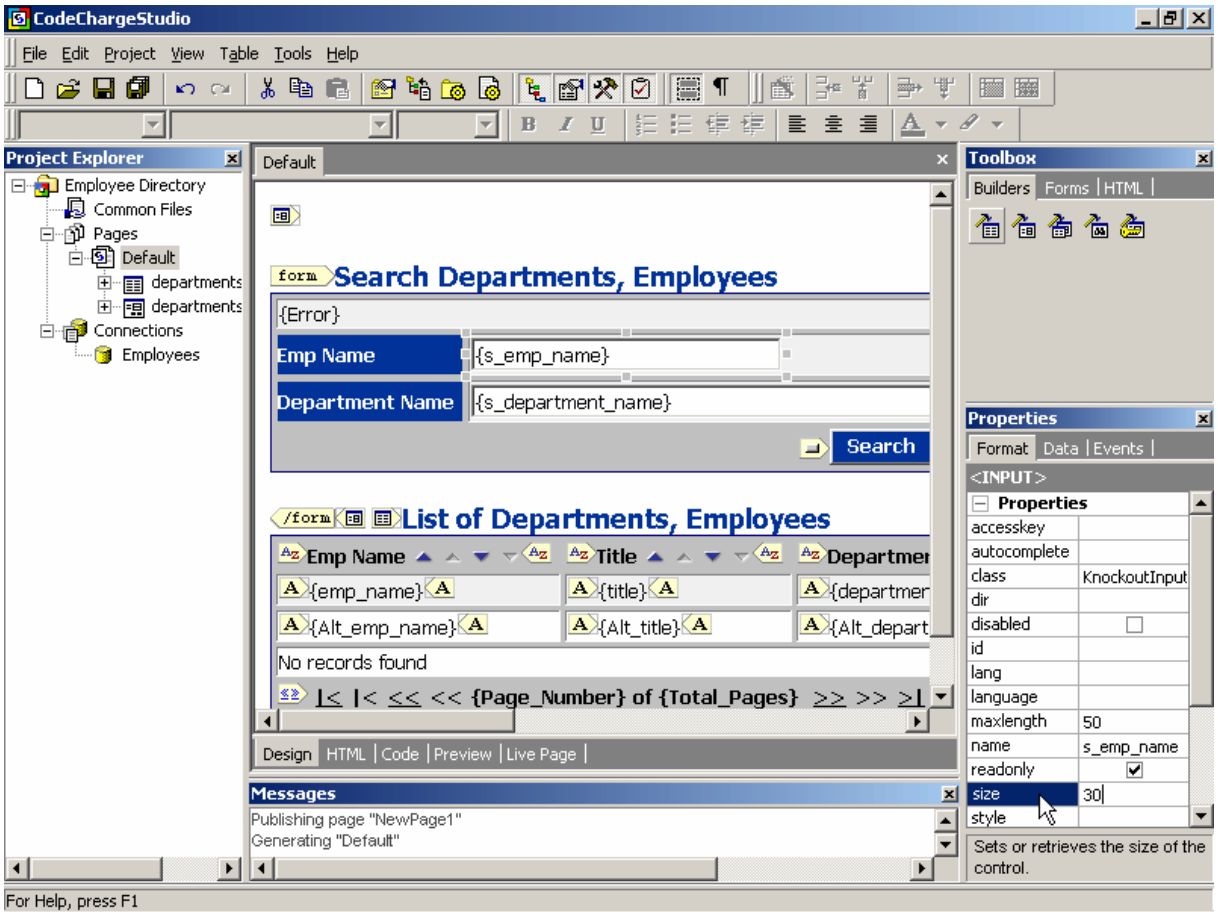
You can also rename a page by clicking on its name in the Project Explorer and pressing F2.

Type the new name for the page.



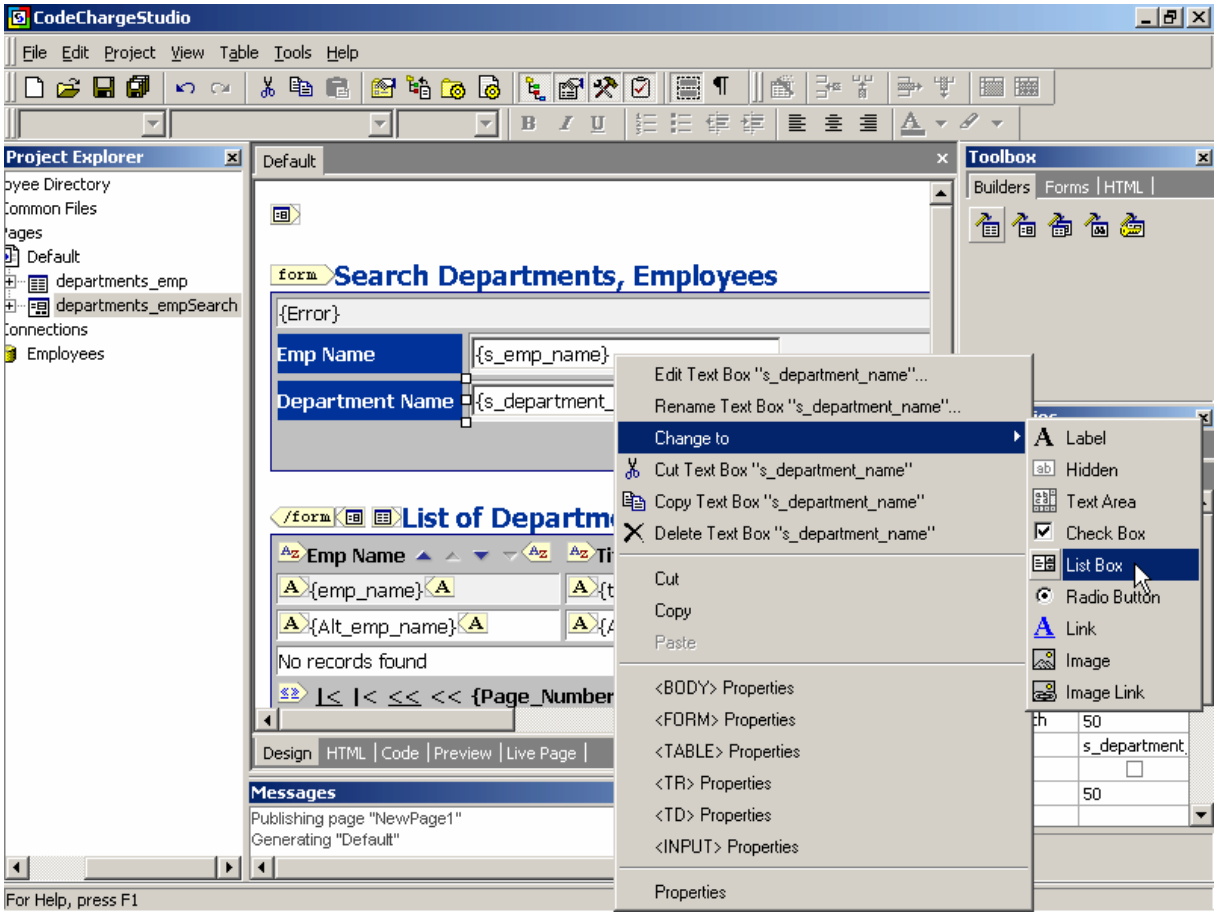
Change the Size of the Search Fields

Since some of the fields may be unnecessarily long, click on the field and adjust its Format Properties, for example by changing the size from 50 to 30.



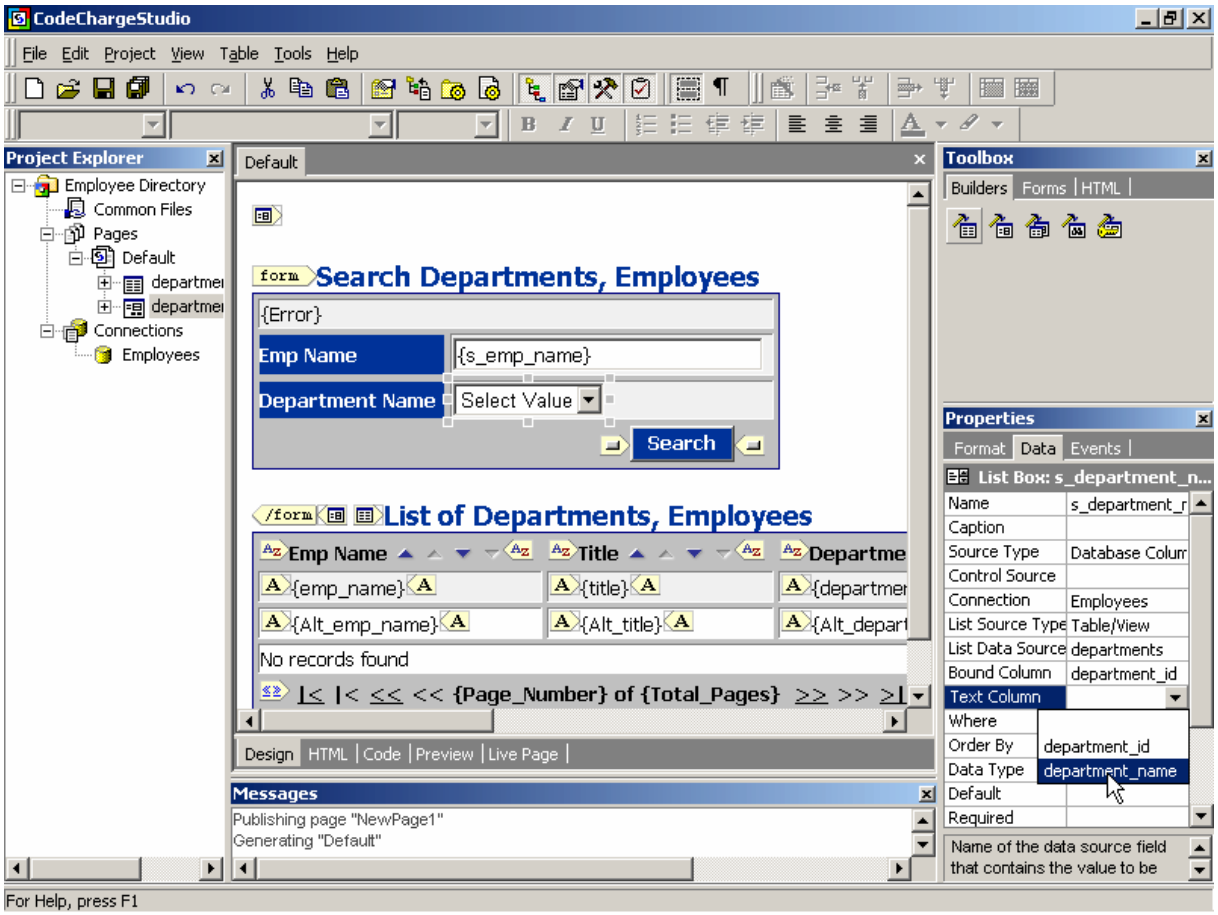
Create a ListBox Field

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.
Right-click on the *department_name* TextBox, then select Change to -> ListBox. This action will change your TextBox to a ListBox.



Configure the ListBox Field

Specify the Connection, Data Source, Bound Column and Text Column for the ListBox.



Connection: The database connection that contains the data for the ListBox.

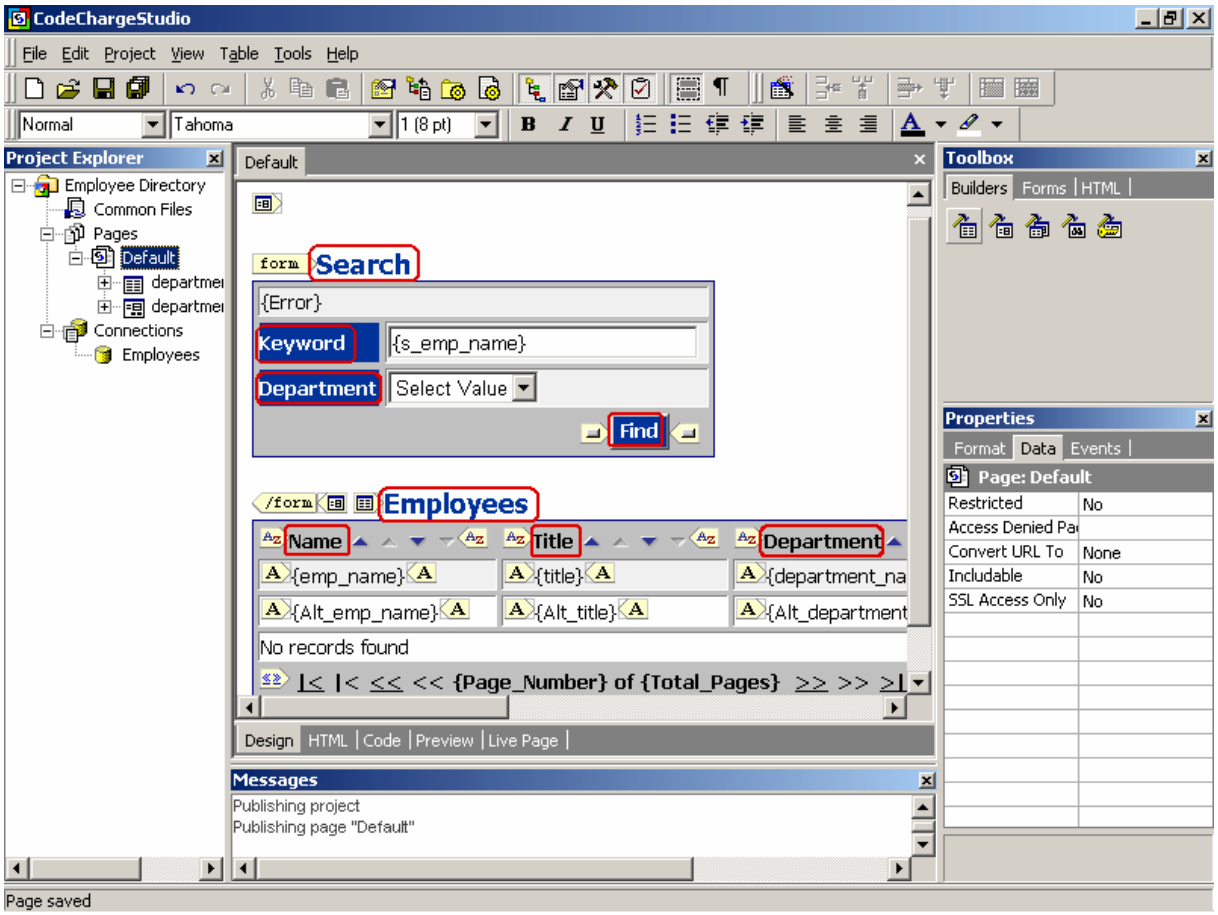
List Data Source: The table, view or SQL query to be used to retrieve database records for the ListBox.

Bound Column: The database field whose values will be submitted by the Listbox.

Text Column: The database field whose values will be displayed in the Listbox.

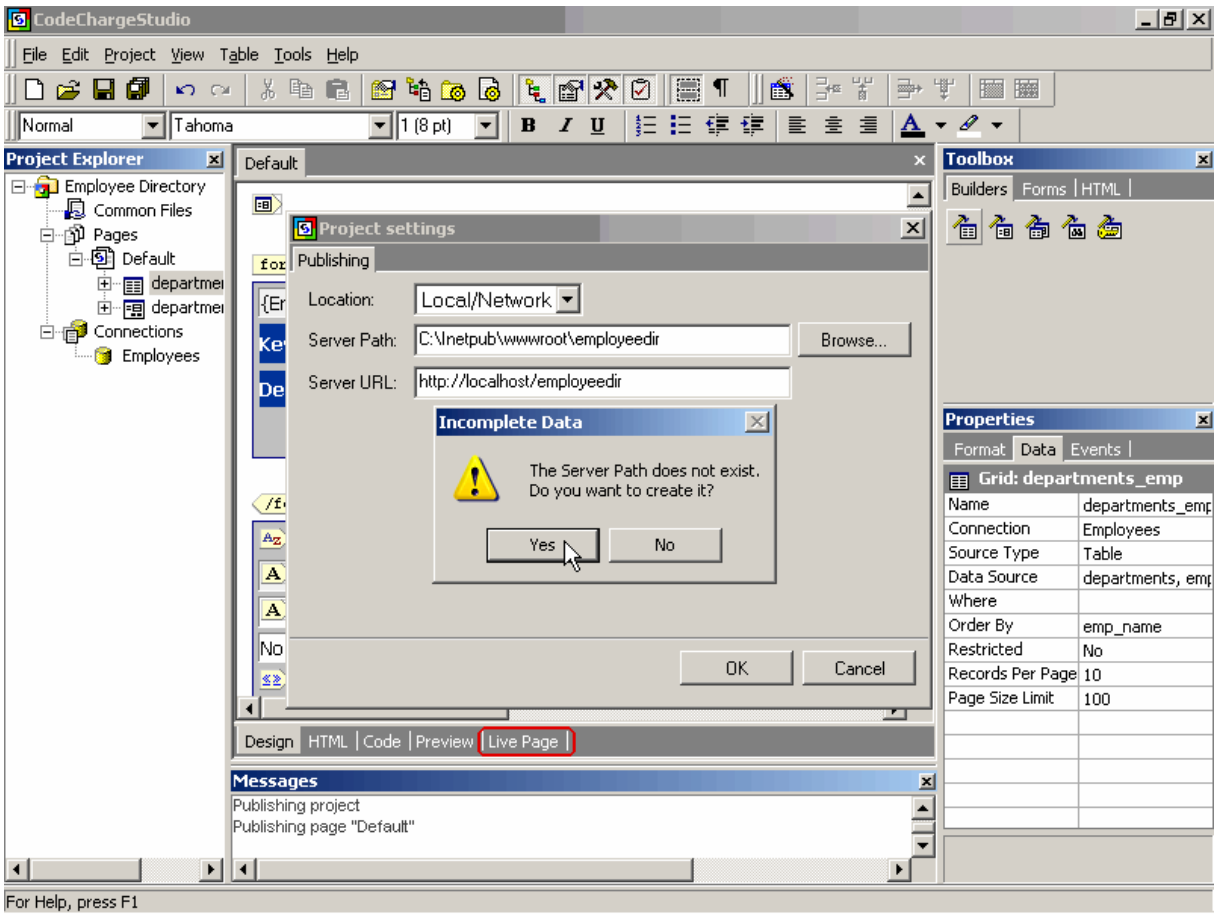
Change Field Captions

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.



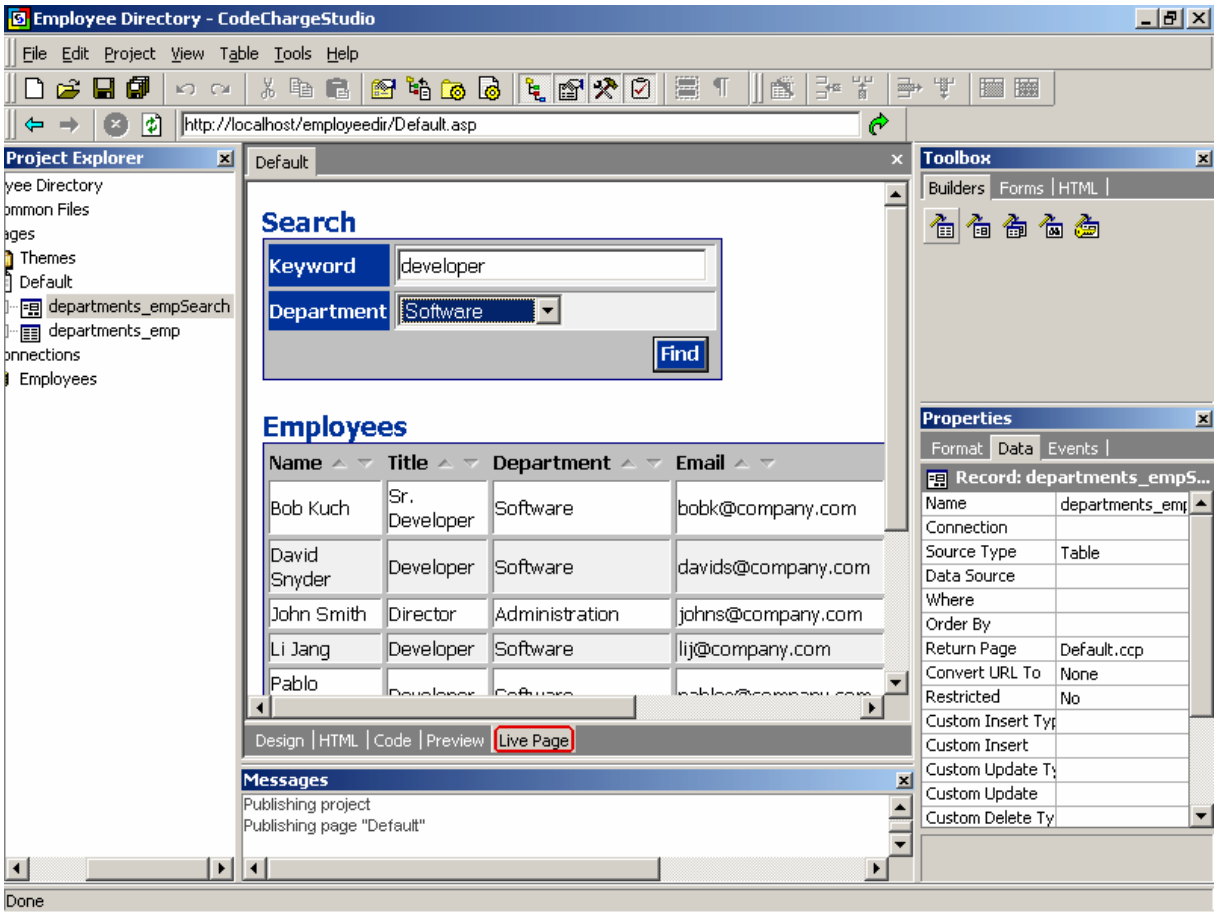
Publish the Page

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. 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.
Click “Yes” to confirm and continue.



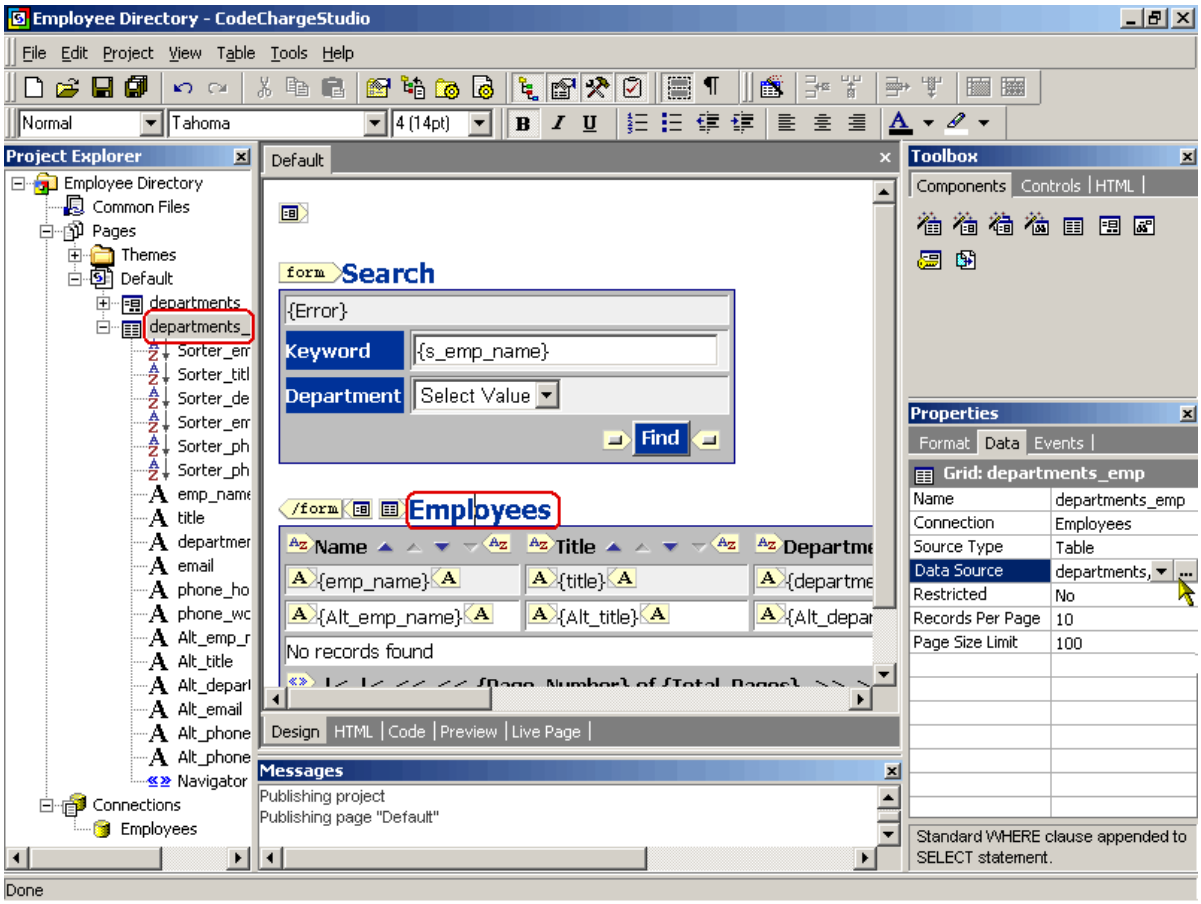
Review and Test the Published Page

When the project is published, you can view and interact with the generated page to test its functionality. Try to enter a search keyword and select a department, then click the “Find” button. You may notice that the search doesn’t work correctly at this time and doesn’t search employee Titles, or doesn’t display results that match the selected department.
This happens because you specified only one field (*employee_name*) in the Grid Builder, then renamed it to “Keyword”, but you have not specified which other fields you would like to search by. Additionally, you added a ListBox to the Search, but you didn’t specify how it should be linked to the search results shown in the grid.



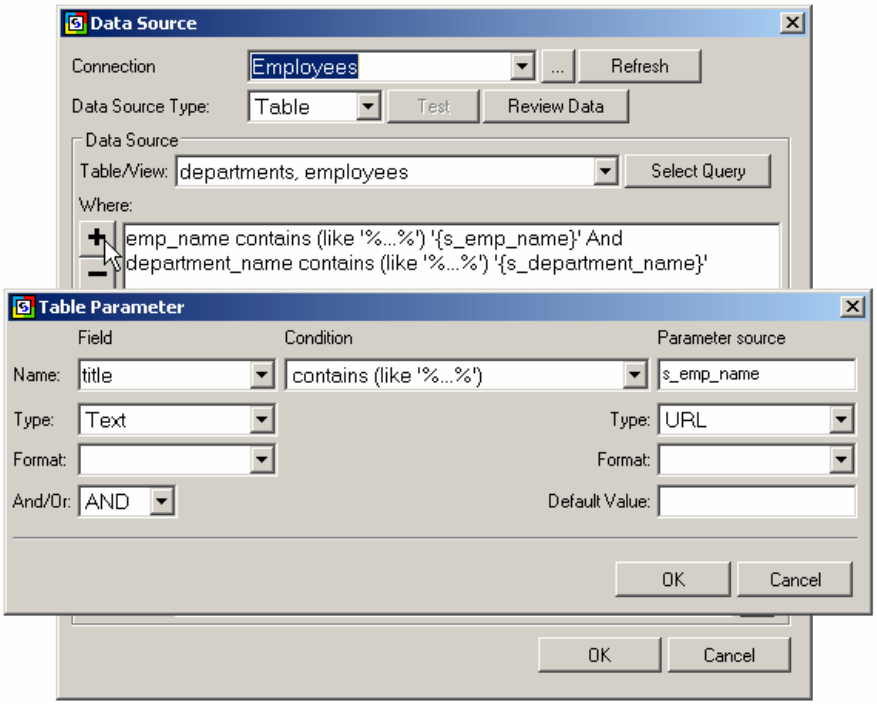
Setup Search Parameters

Go back to page design mode then select the Grid form by either selecting it in the Project Explorer or by positioning the cursor anywhere within Grid's caption on the page.
Then click on […] button in the "Data Source" property of the Grid.

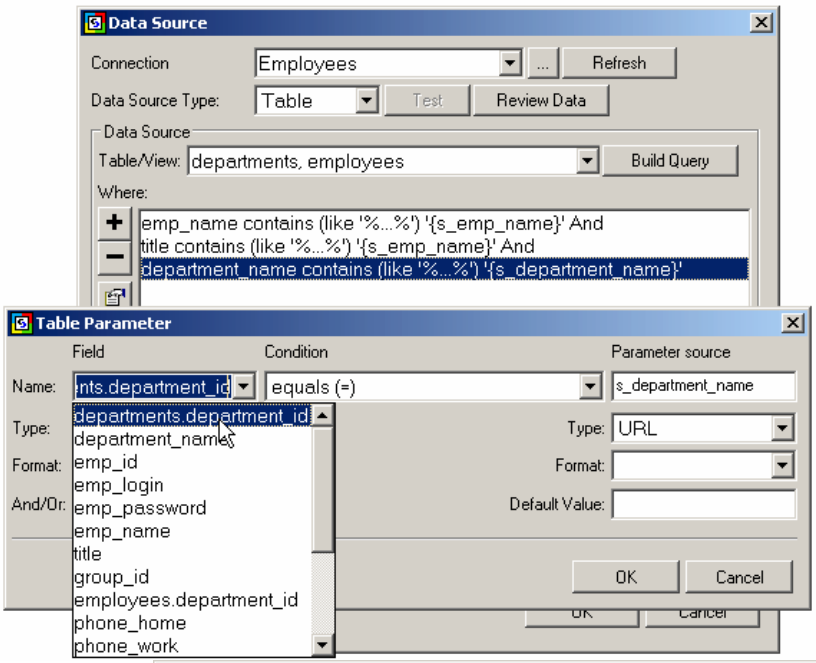


Once the Data Source window opens up you will see several parameters created by the Query Builder in the Where section. The two parameters there indicate that the Grid should be filtered by two keywords: *s_emp_name* and *s_department_name*. Both of those keywords come from the Search form and will be 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.

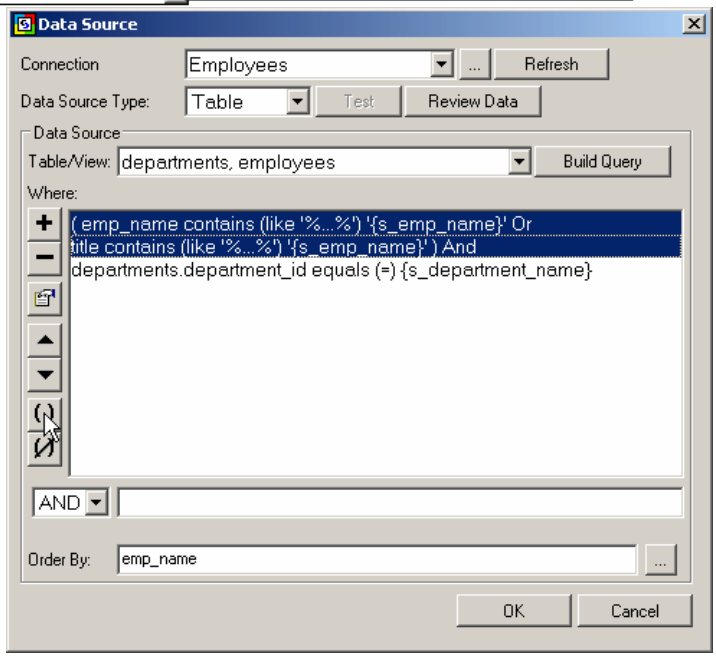
First, click the "+" sign to add a new "Where" parameter that will be used in the Grid. Then select the field *title* and specify that it should be matched against the search parameter *s_emp_name*, the same as used to search *emp_name*. As the "Condition" specify "contains (like '%…%') so that all employee titles that contain the keyword *s_emp_name* will be retrieved.



Now modify the search parameter associated with the *department_name*. Double click on it, and in the parameter setup window select *departments.department_id* as the field that will be searched with the *s_department_name* parameter. Additionally, select "equals (=)" as the Condition since users will select the exact department using the ListBox.

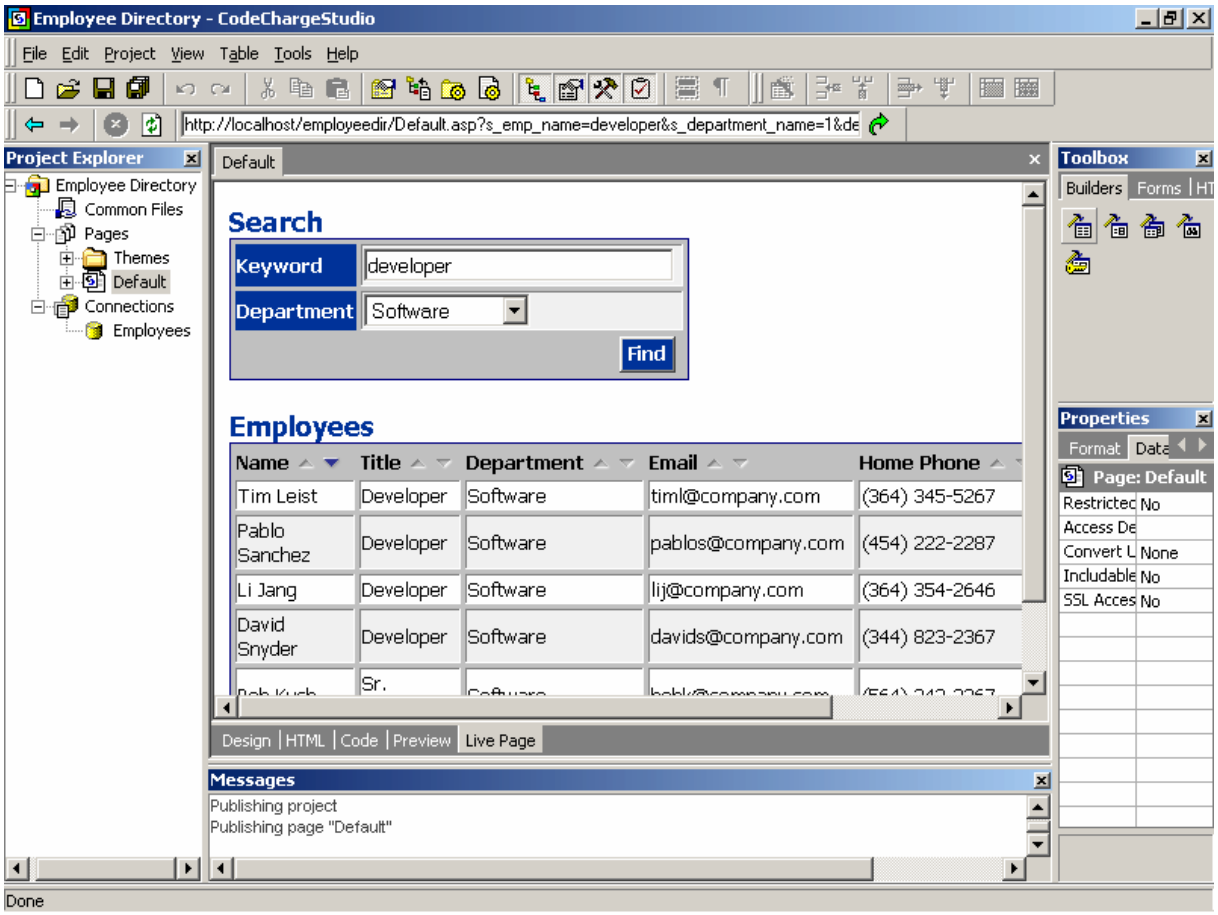


Finally, we need to specify that we want to see the results if either the Employee Name or Title matches the specified Keyword. To do this, select both the *emp_name* and *title* parameters by holding the Ctrl key and clicking on each one. Then click the parentheses button [( )] to make the search of these two parameters independent from searching the department.
Your final Where/Search parameters screen should look like the one shown here.



Preview and Test the Project

As the final step, click on the “Live Page” tab then test the page by interacting with it and testing its functionality.

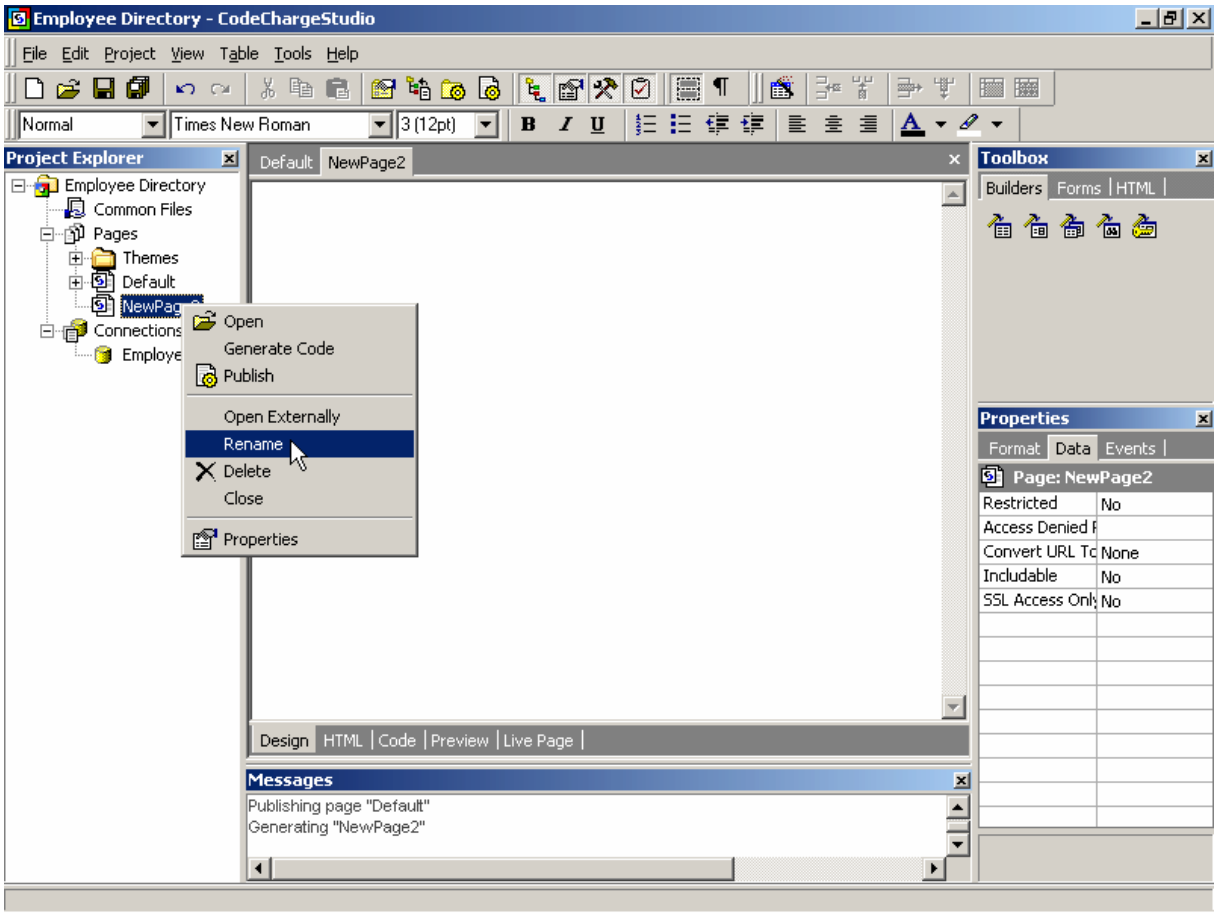


Protecting Web Pages from Unauthorized Access

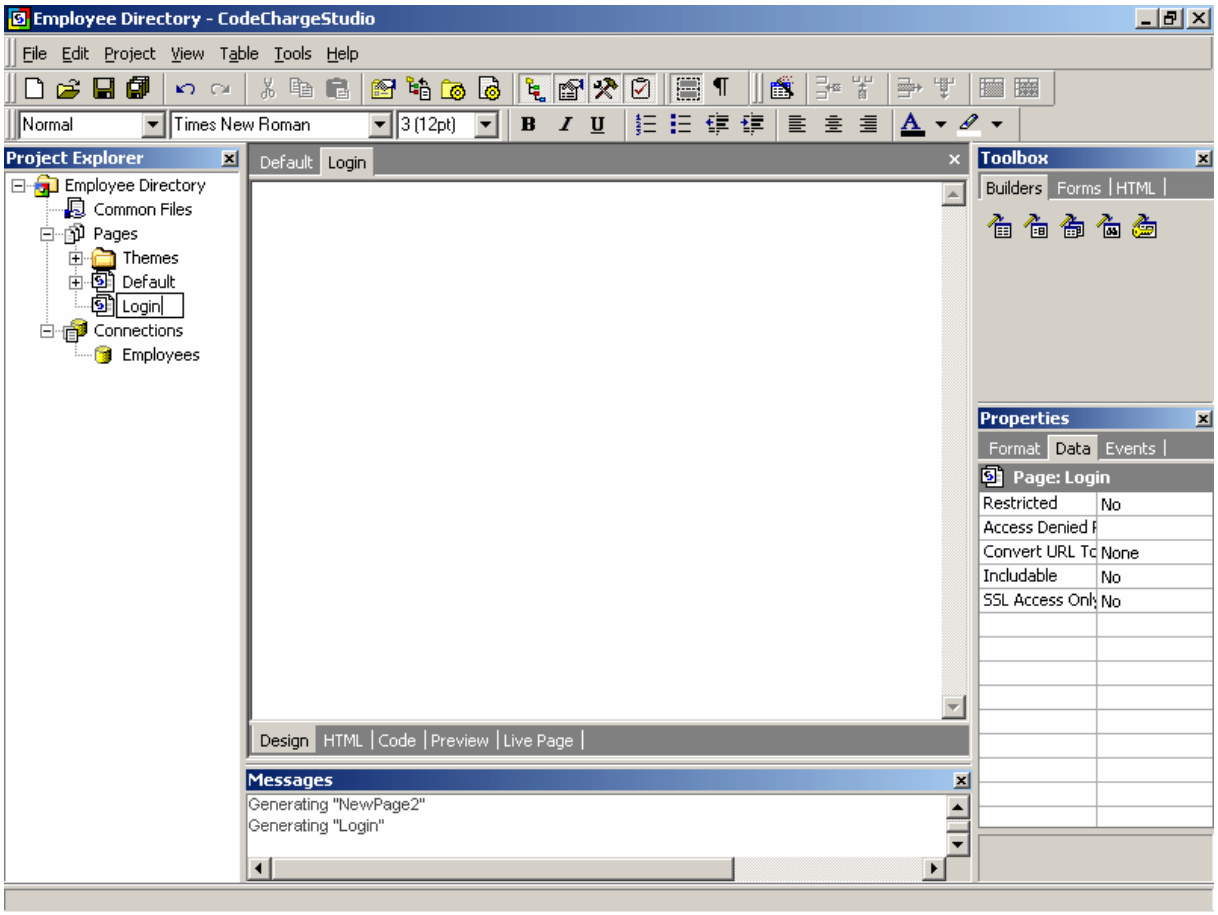
You can configure your project to utilize user authentication and protect certain (or all) pages from unauthorized access. Once correctly setup, generated pages will redirect users to the login page in case the user is not logged in or does not have sufficient privileges to access a restricted page.

Launch the Authentication Builder

Create a new page, then right-click on it's name and select the "Rename" option.

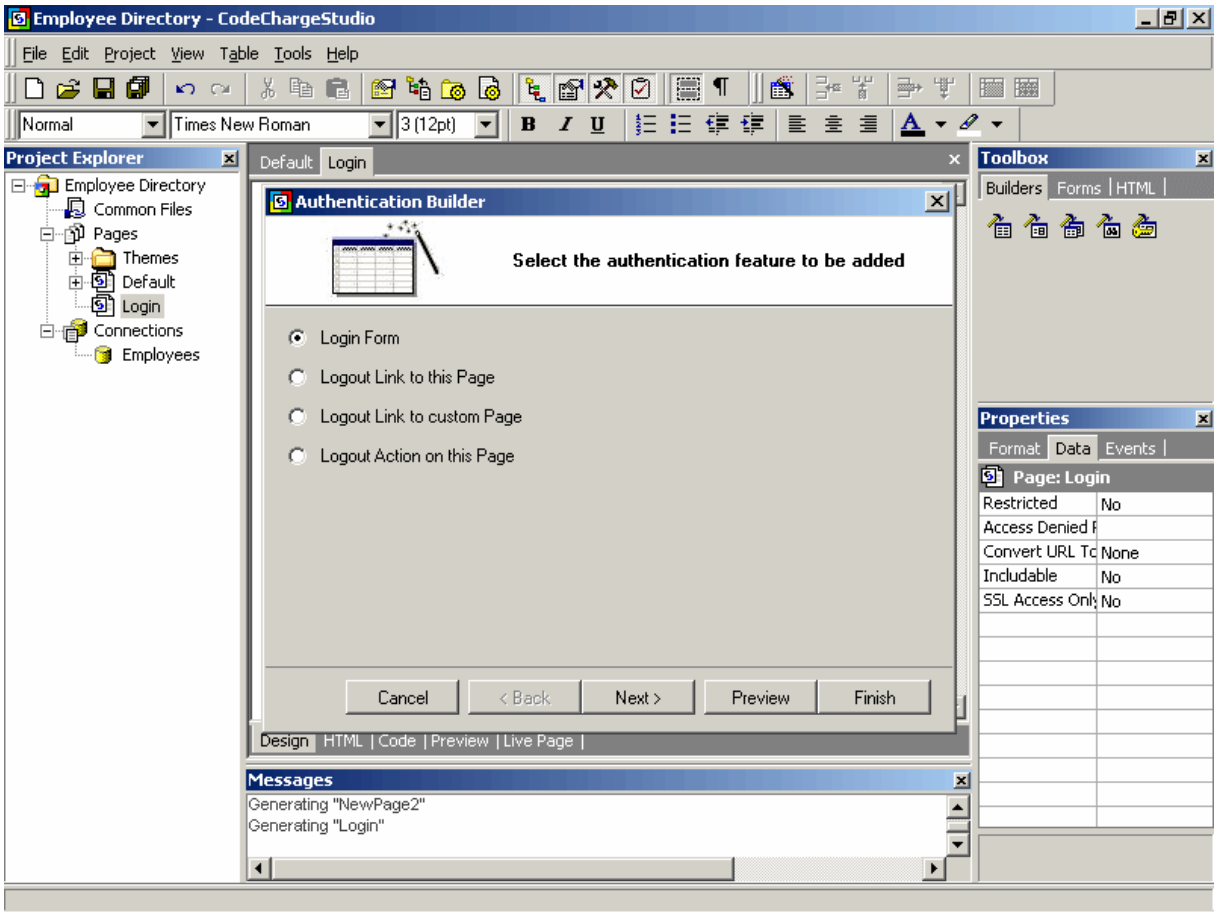


Change the name of the page to “Login”.



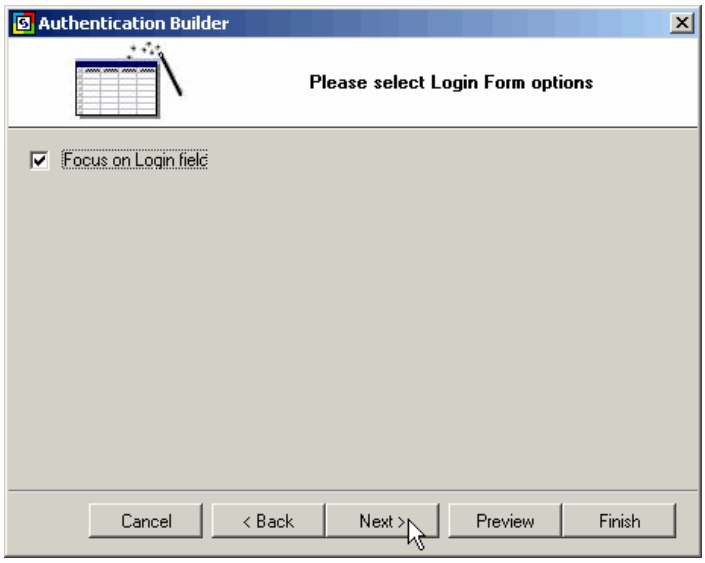
Run the Authentication Builder

Click on the “Authentication Builder” icon in the Toolbox window.
Once the Builder window opens, select “Login Form” and click “Next”.



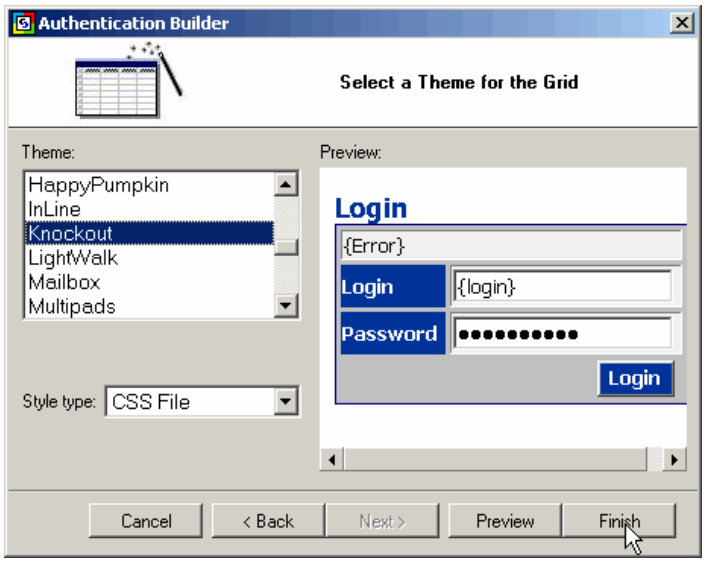
Specify Login Form Options

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 as soon as the page is shown, without the need to click on the Login field first.



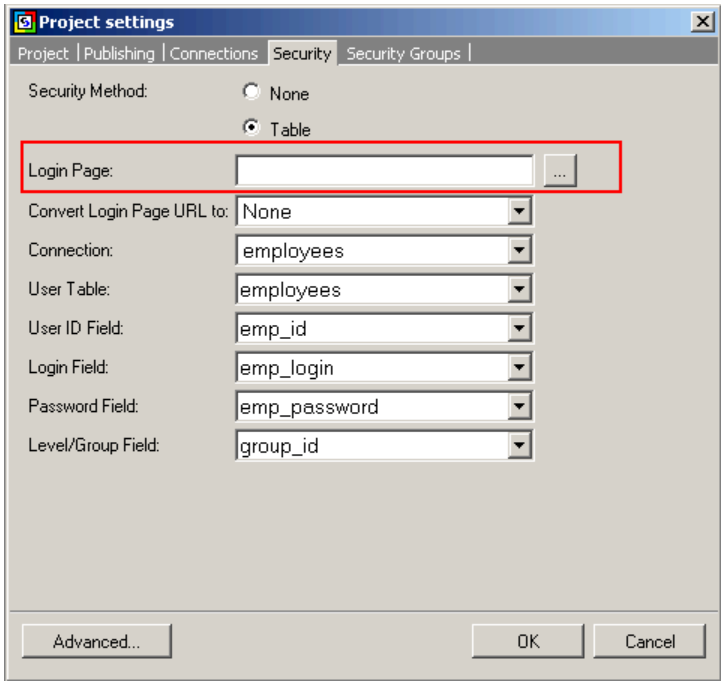
Select a Theme for the Login Form

Select one of the available color and graphics schemes that you would like to use for the login form.

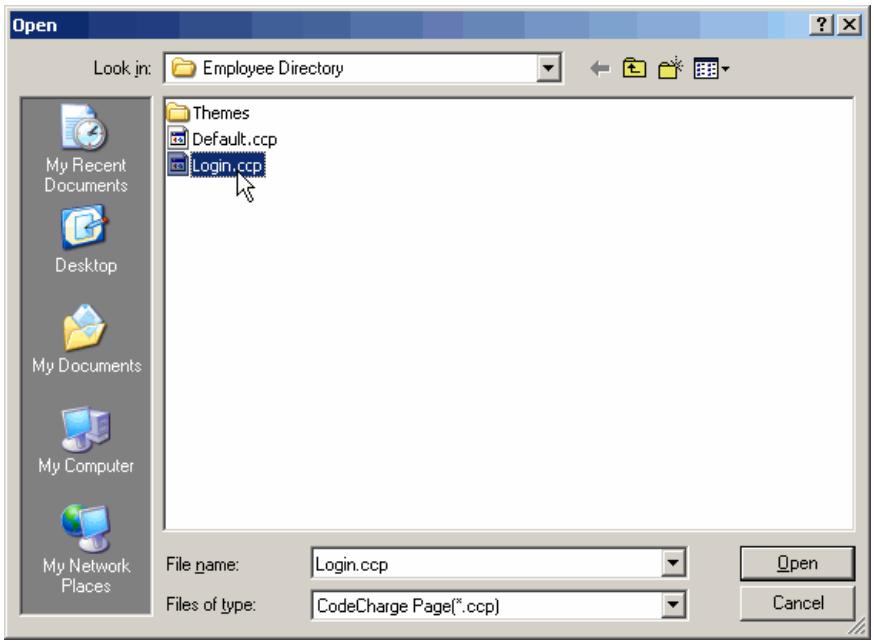


Specify the Login Page for the Project

Once back in the main CodeCharge Studio screen, use the Project→ Settings… menu to open the Project Settings window. Click on the Security tab then click the "[…]" button for the "Login page" property.

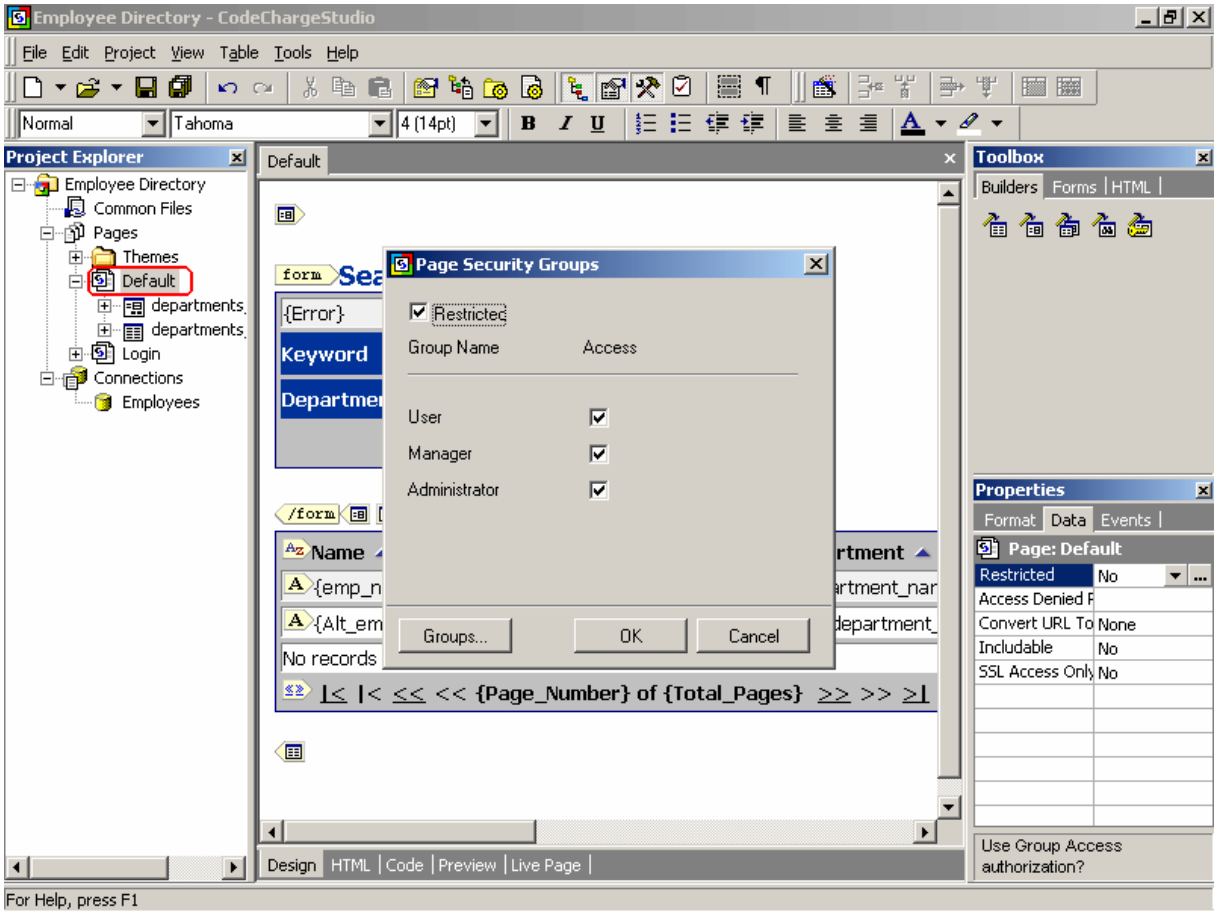


Select the recently created Login page as the main login page for the project.



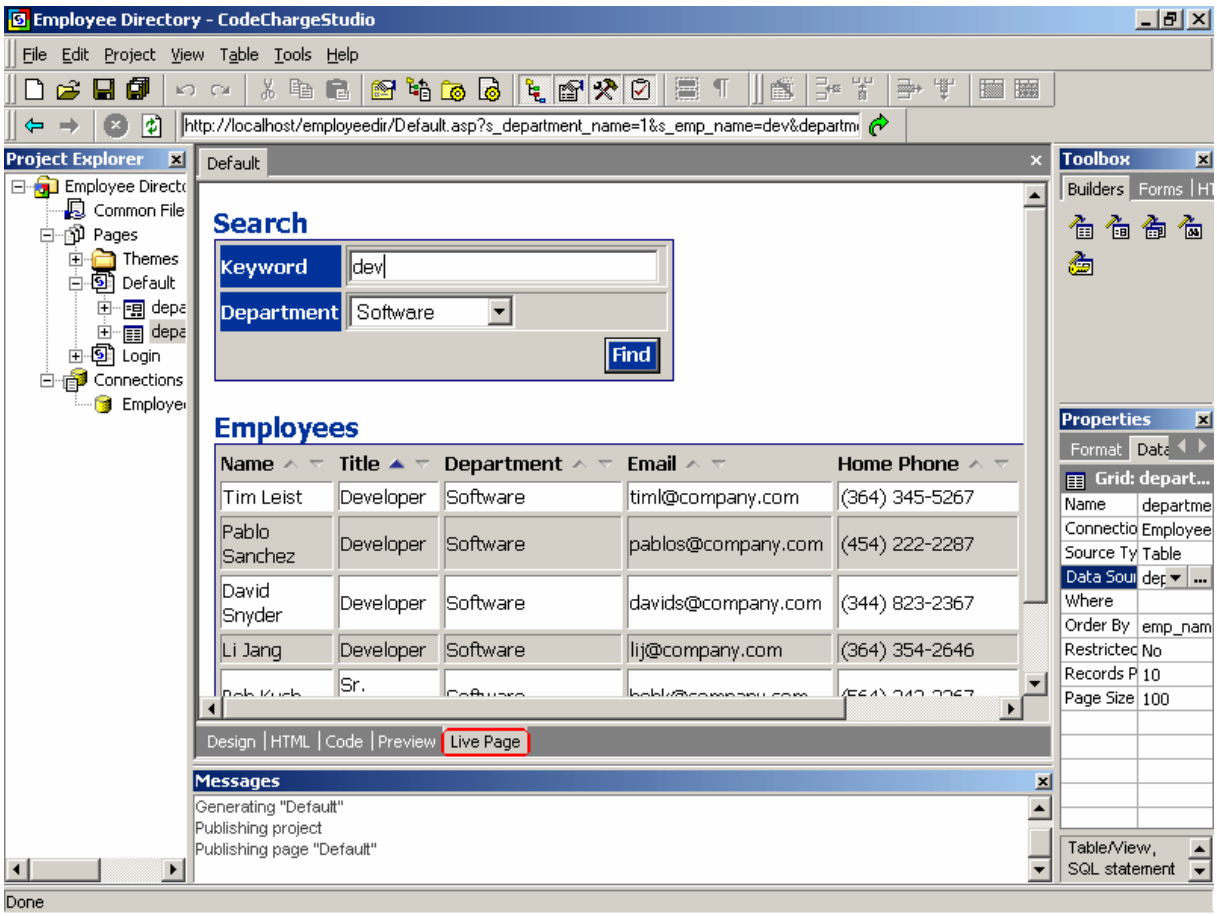
Restrict Page Access

Now you can use the authentication feature by configuring restricted access to your pages.
Select the “Default” page in the Project Explorer then click on the […] button next to the “Restricted” property. In the Page Security Groups windows, 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 via the Login page.



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.



Appendix A –Language Adaptations

This section contains the adaptations of the code included in the tutorial for use with environments other than ASP/VBScript.

Enhancing Application Functionality with Programming Events (C#)

You've probably noticed that until now you've built your Task Management application without having to deal with the programming code. CodeCharge Studio can help you build functional systems without any programming; however creating systems that are more sophisticated will require certain amount of programming. Fortunately, CodeCharge Studio makes programming easy by providing you with a first-rate code editor, in addition to Events and Actions that aid you in inserting pre-programmed snippets of code into the right place within the program.

Here are definitions of an Action and Event:

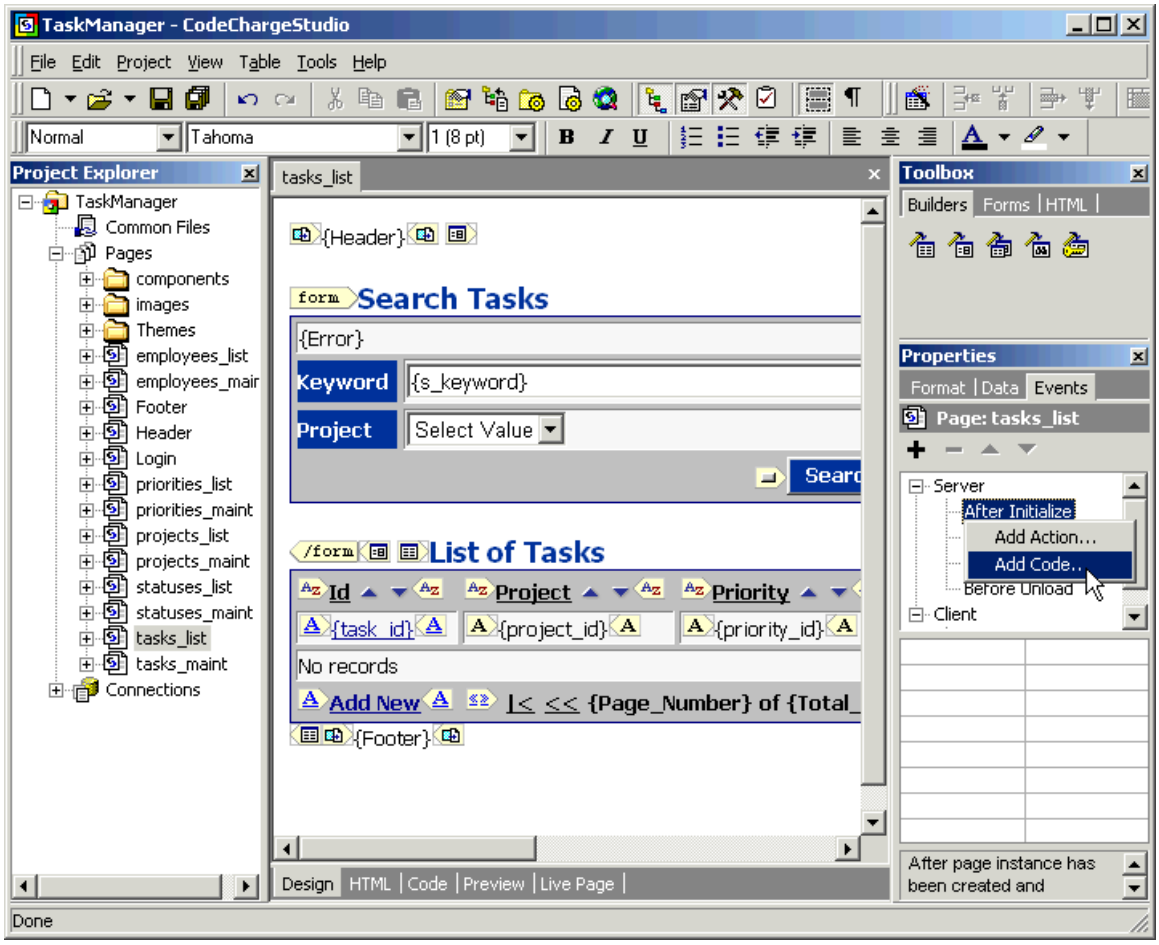
Action
User-definable code generation component, which inserts block of code into an event procedure. CodeCharge Studio comes with several predefined Actions, which are installed into the following folder:
(CCS folder)\Components\Actions
Internally, actions consist of XML and XSL code that can be easily customized.
For example an action can be set on a TextBox to validate the e-mail address.

Event Procedure
A procedure automatically executed in response to an event initiated by the program code during execution. Events are the best place for putting custom programming code.

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, you will mark the listed tasks assigned to you (the current user) by showing your name in *blue* color within the grid.
The logic for this task will be first retrieving the name of the user based on his user id from the database and storing it in a page level variable. Next, you will check the *assigned_to* field of the grid for equality with the earlier retrieved name and modify the generated HTML respectively.

Use "After Initialize" Event to Build Custom Variables

Select the *tasks_list* page in *Project Explorer* window, right-click and select "Properties" to display the properties for the *tasks_list* page. Select the "Events" tab in the *Properties* window, then right-click on the "After Initialize" event and select "Add Code…". The "After Initialize" event for the page occurs in the program after the page has been initialized. This event maps directly to the *Page_Init* event in ASP.NET.



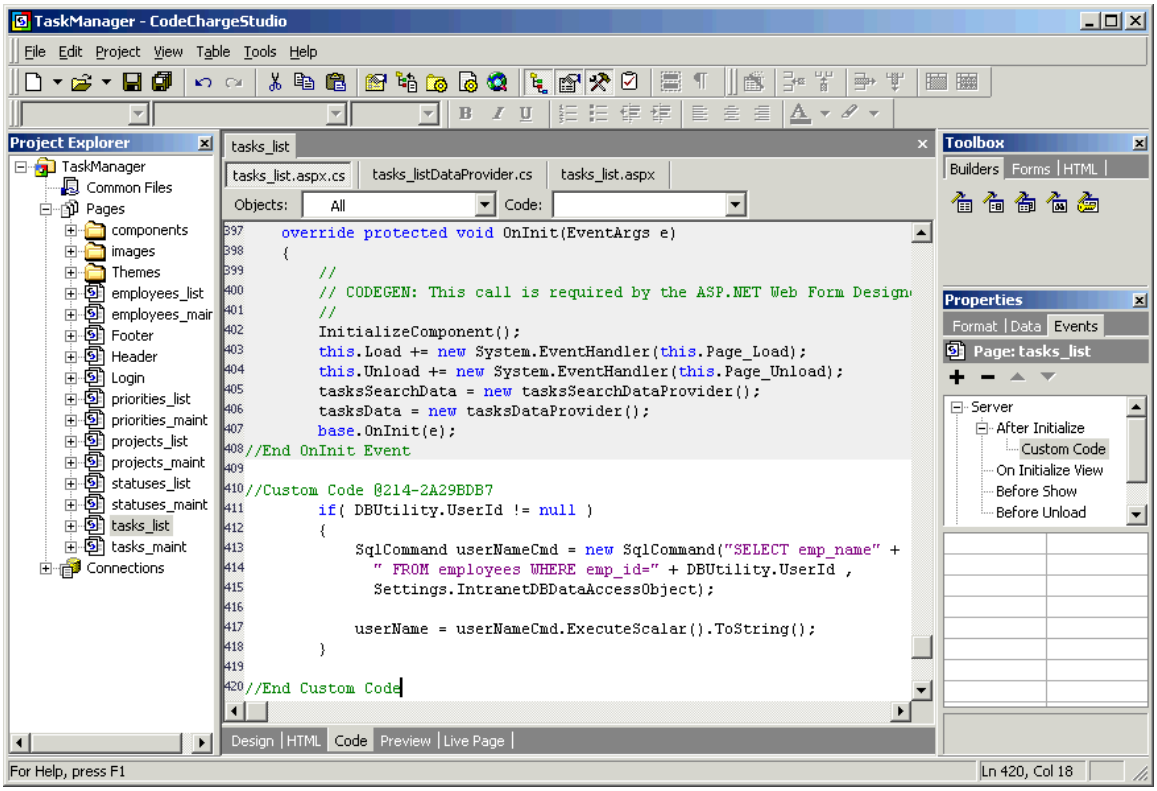
Retrieve Current User's Name from the Database

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

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

with the following lines (C#):

```
if( DBUtility.UserId != null )
{
   SqlCommand userNameCmd = new SqlCommand("SELECT emp_name" +
      " FROM employees WHERE emp_id=" + DBUtility.UserId,
      Settings.IntranetDBDataAccessObject);
   userName = userNameCmd.ExecuteScalar().ToString();
}
```



Let's see how the above code works:

```
if( DBUtility.UserId != null )
```

This is an "if" statement which is *true* only if the variable "DBUtility.UserId" is not *null*. The variable "DBUtility.UserId" holds the id of the current user. The variable "DBUtility.UserID" is set automatically on user login. The following are all default variables created by CodeCharge Studio:

DBUtility .UserID – the primary key field value of the logged in user
DBUtility .UserLogin – login name of the user currently logged into the system
DBUtility .UserGroup – security level/group of the user currently logged into the system.

Hence this statements below the "if" block will only execute if the user has been authenticated:

```
SqlCommand userNameCmd = new SqlCommand("SELECT emp_name" +
   " FROM employees WHERE emp_id=" + DBUtility.UserId ,
   Settings.IntranetDBDataAccessObject);
userName = userNameCmd.ExecuteScalar().ToString();
```

The above statement uses a CodeCharge Studio object "SqlCommand". This object is used to execute queries against the database. It is beneficial to use this object since it automatically takes care of creating the connections with the database and disposing them appropriately. The constructor of the "SqlCommand" class takes two parameters; the first parameter is the SQL query to execute again the database. As it is clear from the above code snippet, we are selecting the "emp_name" field from the "employees" table where the "emp_id" is equal to the current user's id. In short, you are retrieving the name of the current user. The second parameter passed is again a CodeCharge Studio object of the type "DataAccessObject". The *DataAccessObject* is an abstraction layer over different ADO.NET managed providers. For each database connection created in the "Project Explorer" window there is a *DataAccessObject* defined in the "Settings" class. In the next line, the "ExecuteScalar" method executes the query against the *DataAccessObject* and returns the name of the current user, which is then stored in the variable "userName".

Define Page Level Variables

The user name retrieved in the above method is needed in another method where you will be changing the HTML output based on the user name. Hence the “userName” variable used in the above code segment needs to be defined as a page level variable. Scroll your code editor above other page variables are defined ending with the line shown below.

```
//End Forms Objects
```

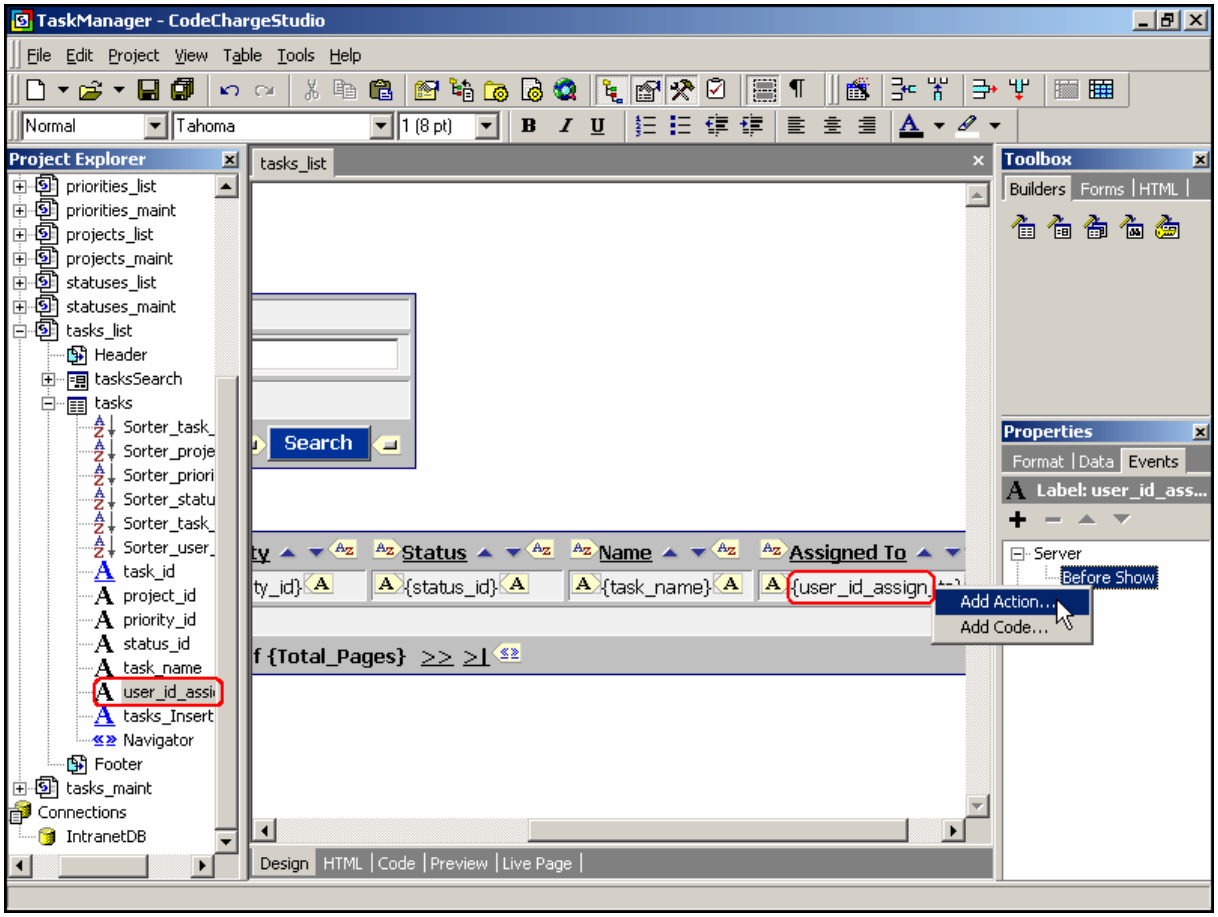
now add the following variable declaration below the above mentioned line.

```
protected string userName = "" ;
```

Once the variable has been set, now as soon as an authenticated user requests this page, his name will be stored in the above variable.

Programmatically Modify a Grid's Field

Again expand the *task* grid under the *tasks_list* page in "Project Explorer" window, right-click on the Label *user_id_assigned_to* and select "Properties". Under the "Data" tab, set the value of the "Content" property to "HTML". Next, select "Events" tab in the "Properties" window, then right-click on the "Before Show" event and select "Add Code…". The "Before Show" event for a grid filed occurs after the field has been data bound, but before being output as HTML. This event directly maps with the *ItemDataBound* event of the *Repeater* web-control. By adding code into this event, you can modify the field's value before it is shown.



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.
Replace these lines of code:

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

with the following lines (C#):

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

Let's explain how the above C# code works:

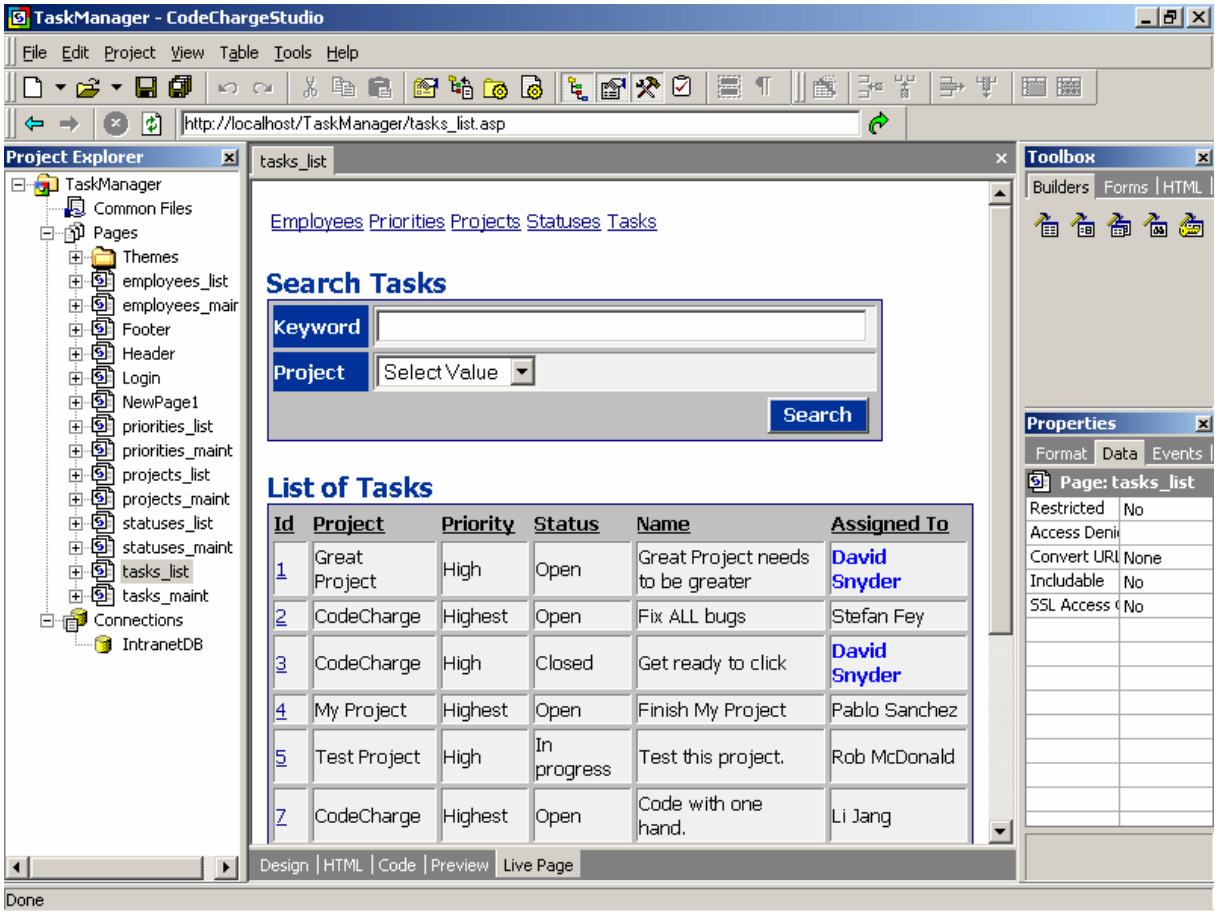
The "if" condition is *true* only if the grid field *user_id_assign_to* (containing the name of the user to whom the task is assigned) is equal to the name of the employee that is currently logged into the system. Therefore, once you login to the system, the program will recognize your tasks by comparing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

```
tasksuser_id_assign_to.Text = "<b><font color='blue'>" + userName + "</b></font" ;
```

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

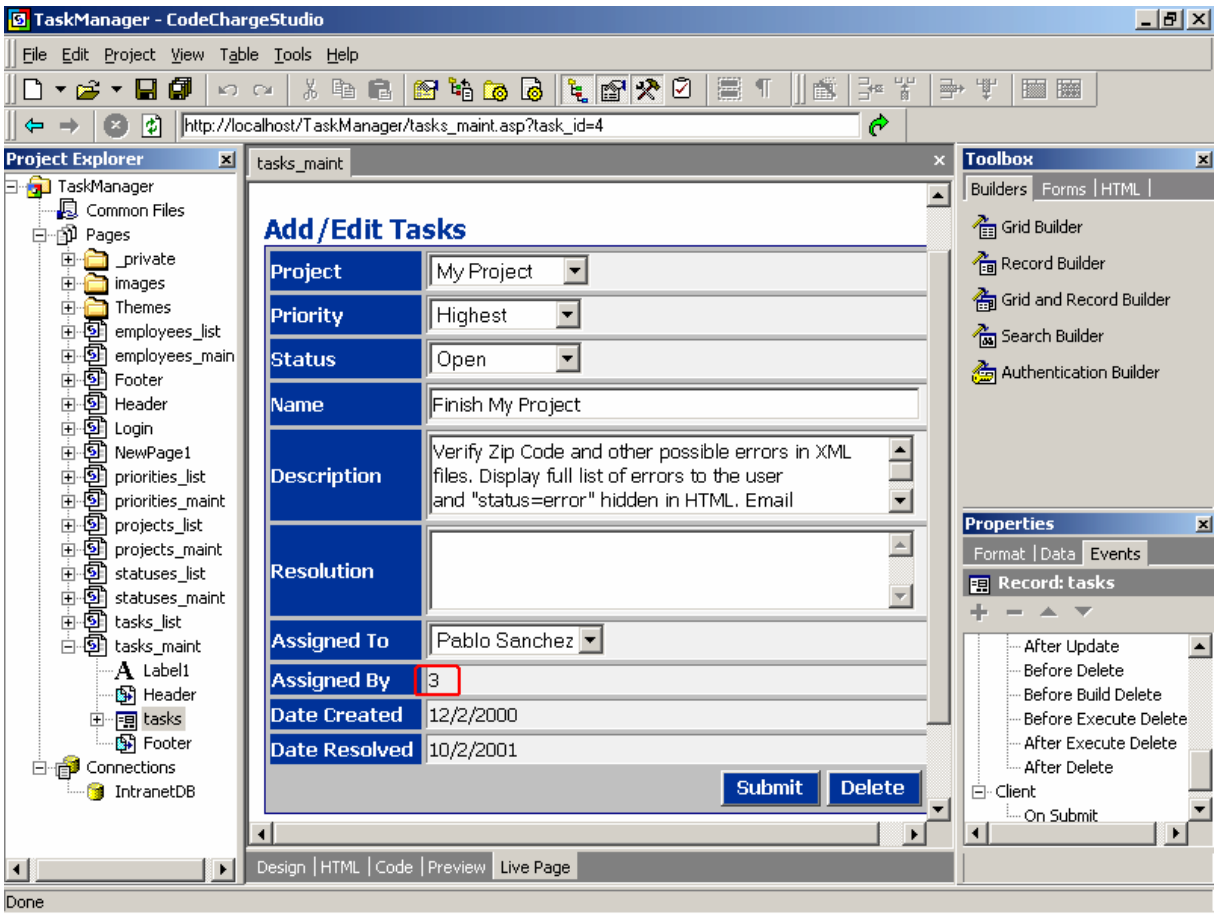
Preview the Tasks List Page

Save your project and 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.
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. Click on any of the project Ids and you should see the login page.
Login as davids / davids , then 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.



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.



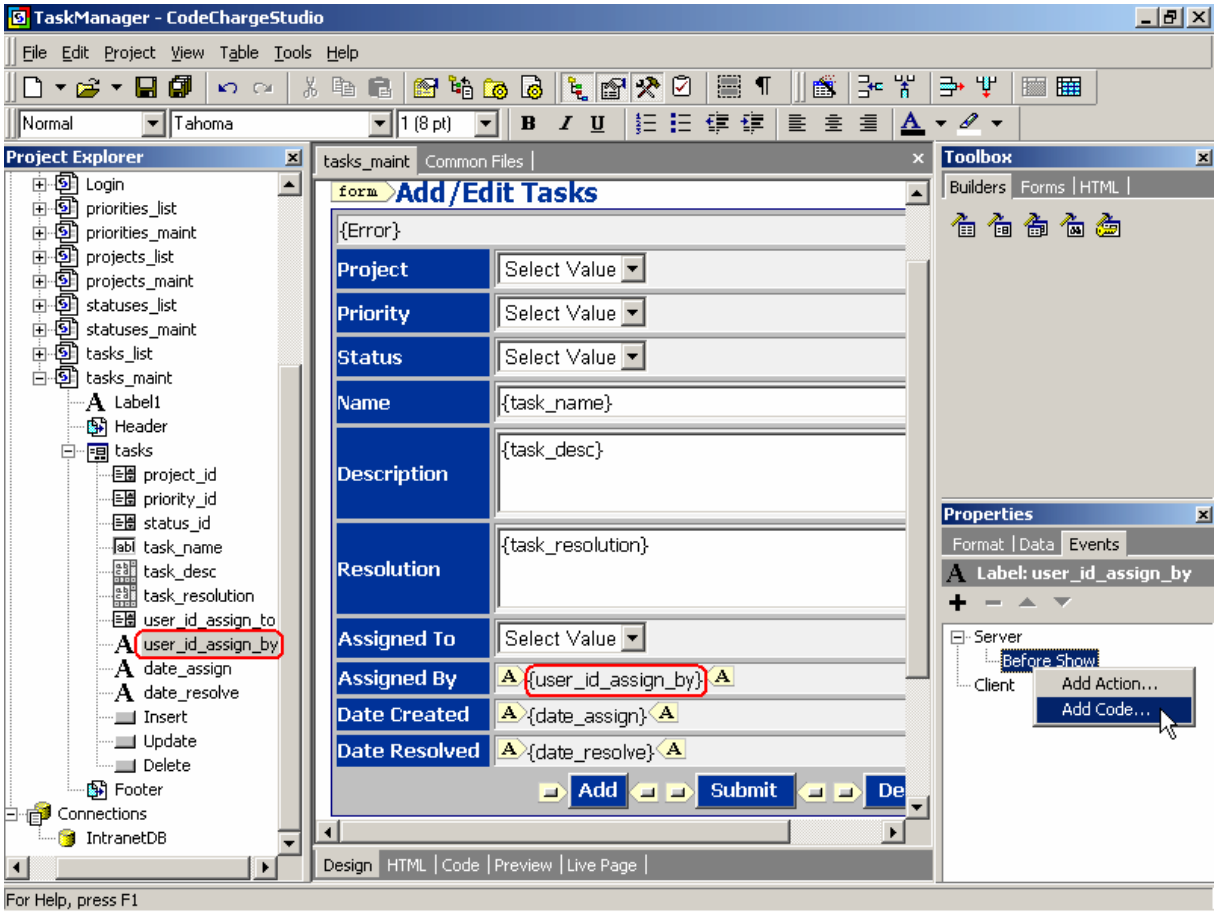
There are several potential methods of dealing with the above issue and let's explain each one in detail:

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.

Create “Before Show” Event to Alter Label’s Value

Select the Label *user_id_assign_by* in the Design mode, then in the “Properties” window click on “Events” tab.
Right-click on “Before Show” event and select “Add Code…”.
CodeCharge Studio should automatically switch to Code view at that time.
“Before Show” event is a place in the program that is executed after a value is assigned to the control, but before such value is displayed.



Use “Before Show” Event to Alter Label’s Value

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

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

replace the above lines with the following:

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

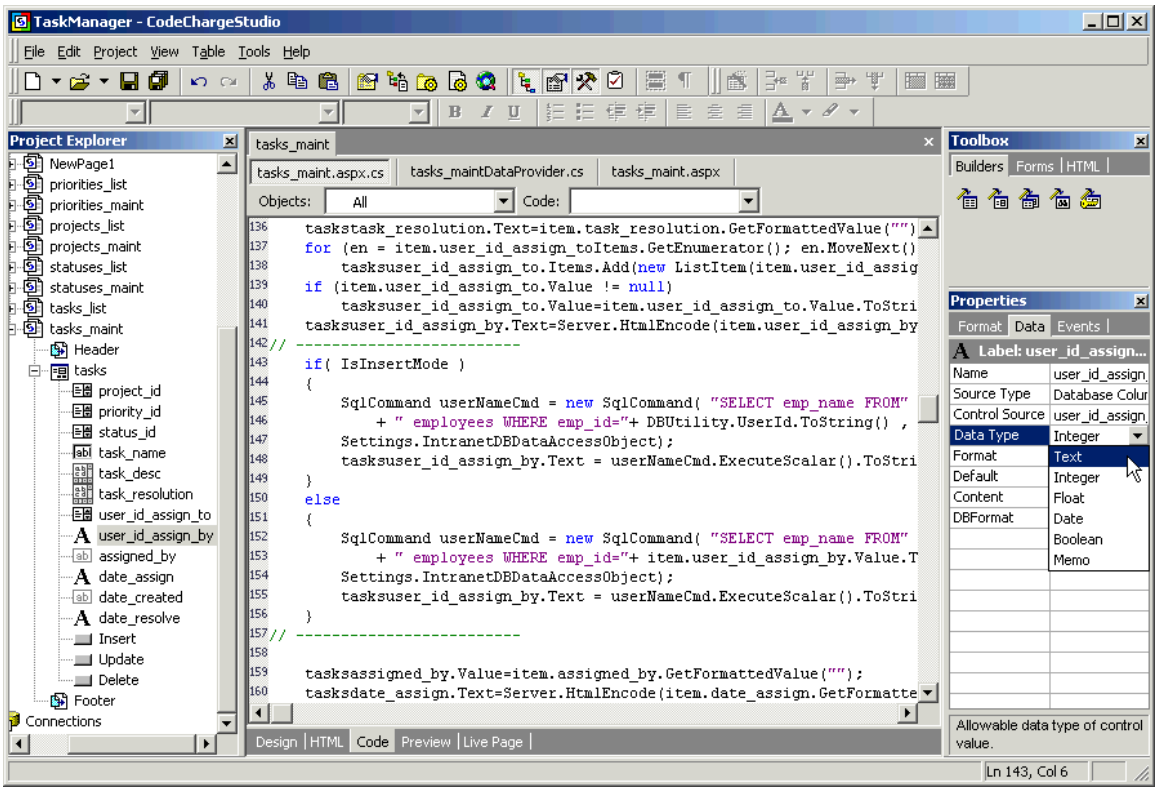
In the above code snippet, we have an “if” block which checks the value of the Boolean variable “IsInsertMode”, this variable generated by CodeCharge Studio within the “Before Show” event of the record forms. Since the same CodeCharge Studio record form can be used to update records, delete records as 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.
Within 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 actually assigned the record, 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:
Assign current to the *tasksuser_id_assign_by* Label by retrieving his/her name from *employees* table using SqlCommand object that uses IntranetDBDataAccessObject and DBUtility.UserId property that obtains current user’s id.
If a record is being edited:

Assign the name of the person who originally submitted the issue to the tasks*user_id_assign_by* Label, by looking up employee's name from *employees* table using SqlCommand object that uses IntranetDBDataAccessObject and the value of the item.user_id_assign_by property."

Refer to the CodeCharge Studio Programming Reference for more information about methods and properties available in CodeCharge-generated programs.

Now that you've programmatically modified the value of the *tasksuser_id_assign_by* Label to output employee name instead of the id, you will also need to specify that this field is now a Text field, instead of Numeric. Click on "Data" tab in "Properties" window, and select "Text" as the Data Type.



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

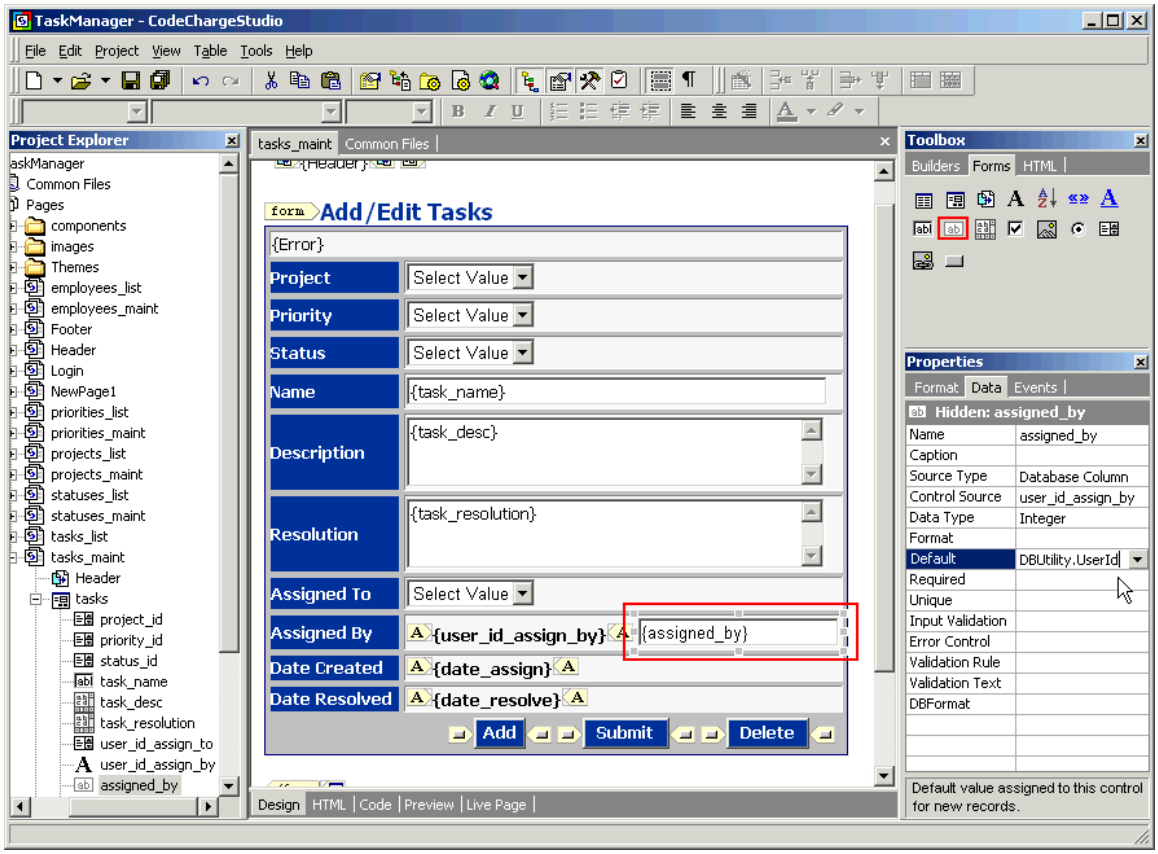
You've previously used the "Before Show" event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or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

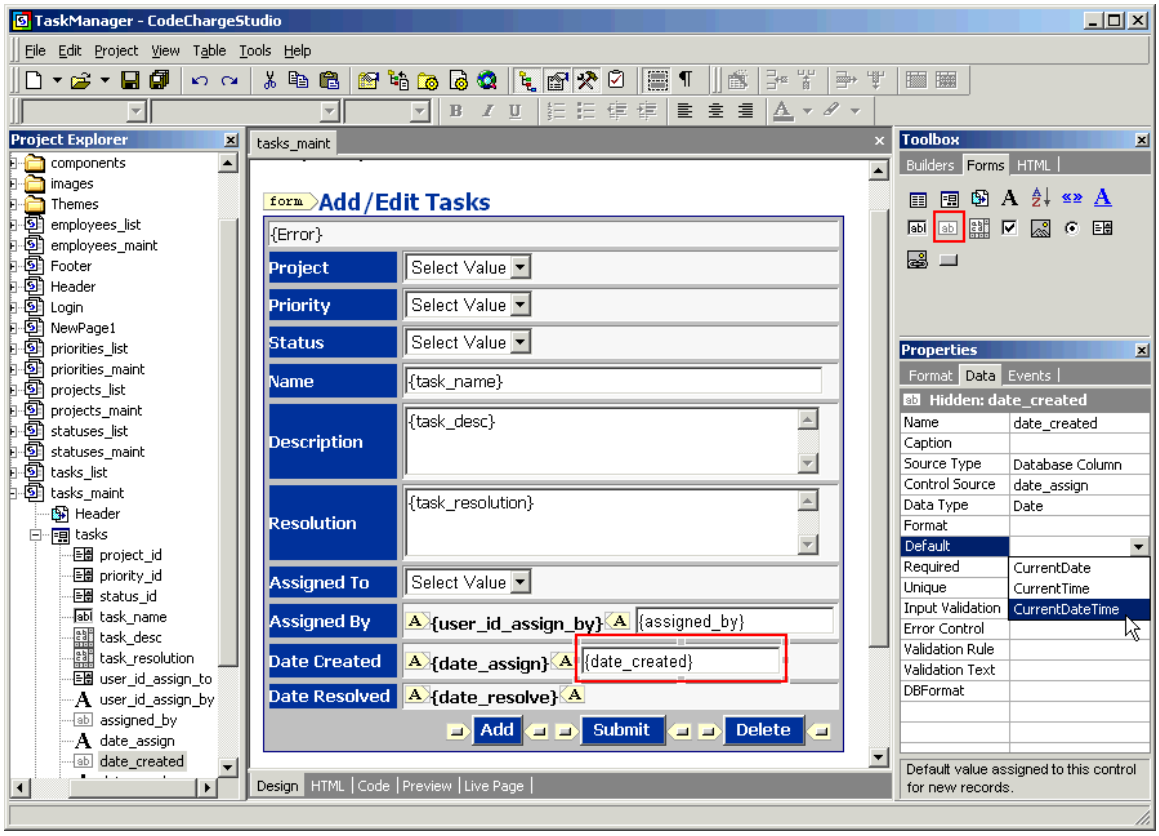
Default: *DBUtility.UserID* – default value for this field if empty. *DBUtility.UserID* is a CodeCharge property that retrieves the user id of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
askManager
Common Files
Pages
components
images
Themes
employees_list
employees_maint
Footer
Header
Login
NewPage1
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolution
user_id_assign_to
user_id_assign_by
assigned_by
tasks_maint
Common Files
form
Add/Edit Tasks
{Error}
Project
Select Value
Priority
Select Value
Status
Select Value
Name
{task_name}
Description
{task_desc}
Resolution
{task_resolution}
Assigned To
Select Value
Assigned By
{user_id_assign_by}
{assigned_by}
Date Created
{date_assign}
Date Resolved
{date_resolve}
Add
Submit
Delete
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Hidden: assigned_by
Name assigned_by
Caption
Source Type Database Column
Control Source user_id_assign_by
Data Type Integer
Format
Default DBUtility.UserId
Required
Unique
Input Validation
Error Control
Validation Rule
Validation Text
DBFormat
Default value assigned to this control for new records.

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.
Configure the new field as follows:

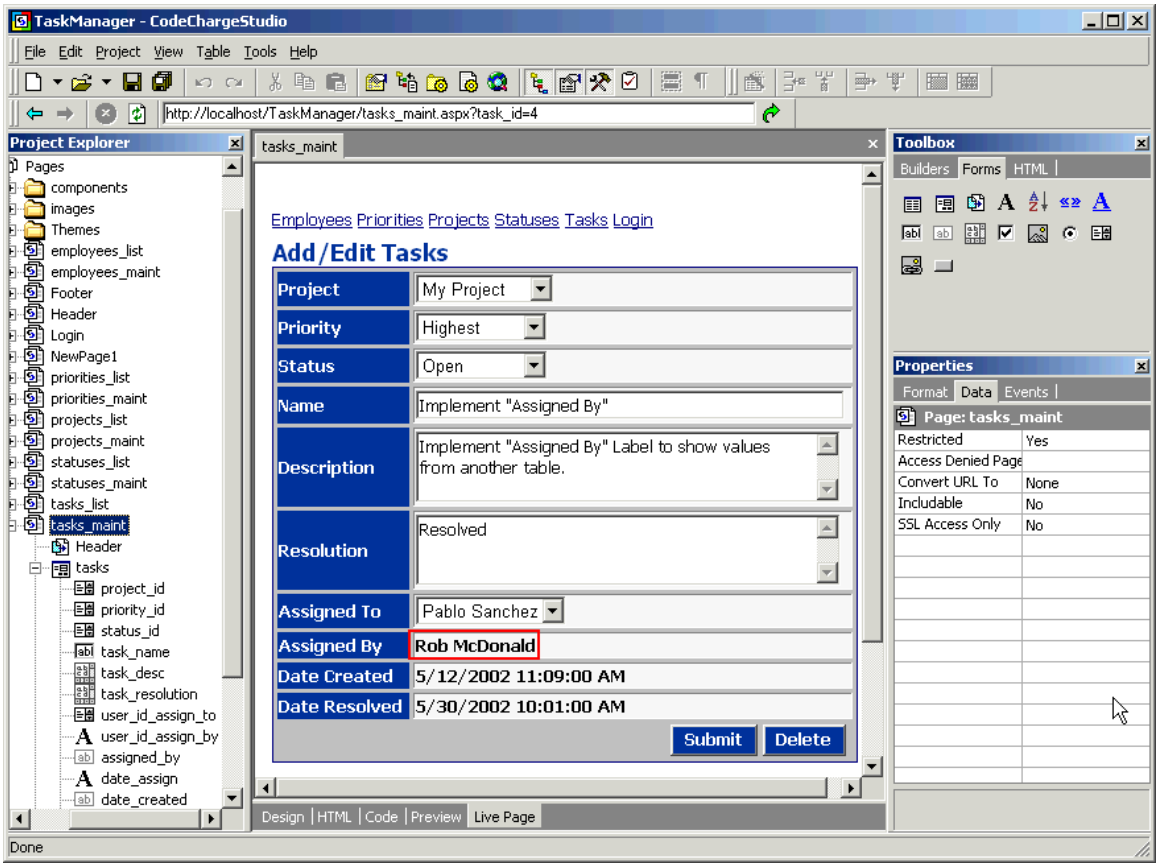
Name: *date_created*
Control Source: *date_assign*
Data Type: *Date*
Default: *CurrentDateTime* – The "CurrentDateTime" property allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.



Test the Label and Hidden Fields

Finally, you can switch to Live Page mode, select a Task, then login and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.
Don't forget to save it!



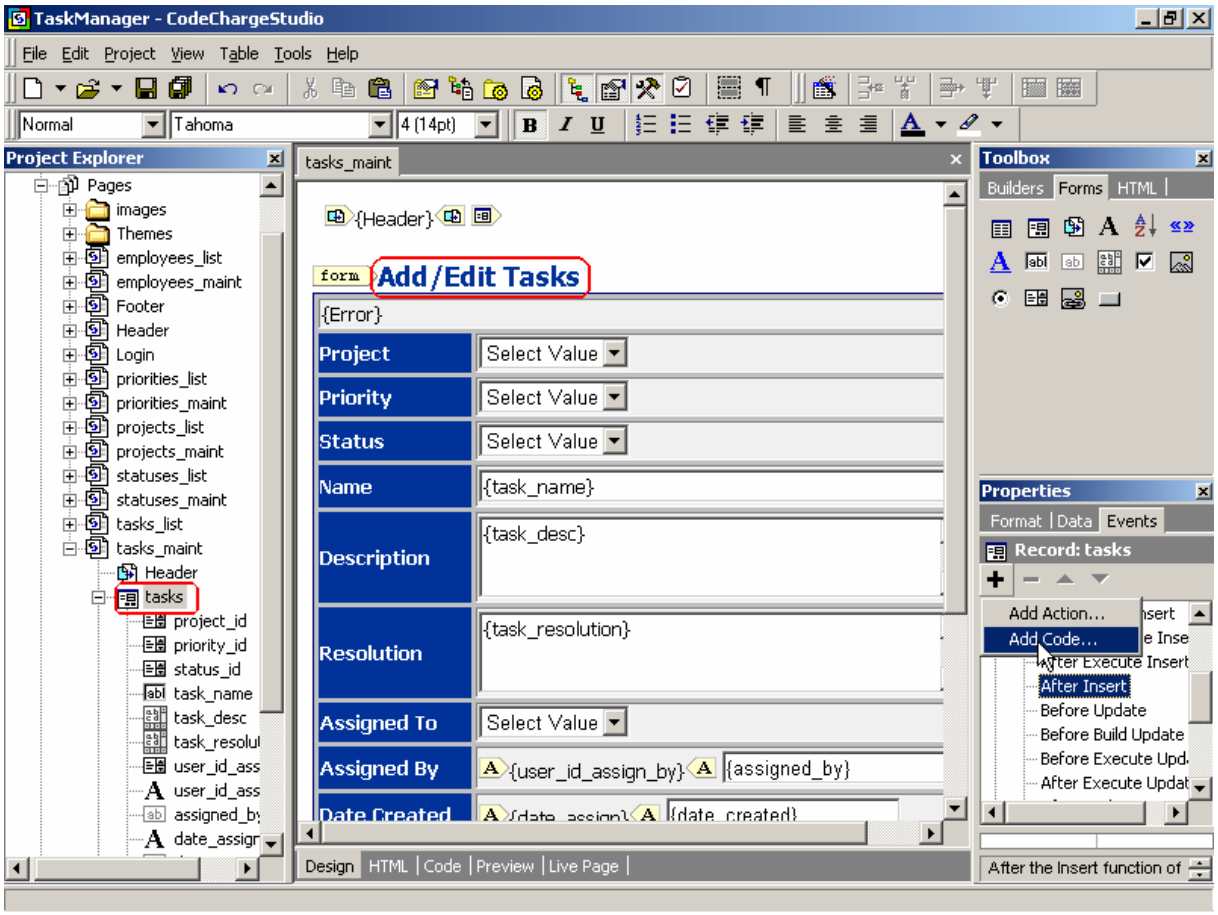
Programming the Record Form

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

Add Code in the "After Insert" Event to Send Emails

Select the "tasks" form by selecting it in the "Project Explorer", or clicking anywhere within the form's caption. Then in the "Properties" window click on the "Events" tab and select the "After Insert" event.
Click on the [+] button, then select "Add Code…"



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

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

with the code below:

```
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.Send( newMessage );
```

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:

c) The above code use the classes provided by the .NET Framework to send e-mails, so you do not need to install extra components.

d) 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.

```
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();
```

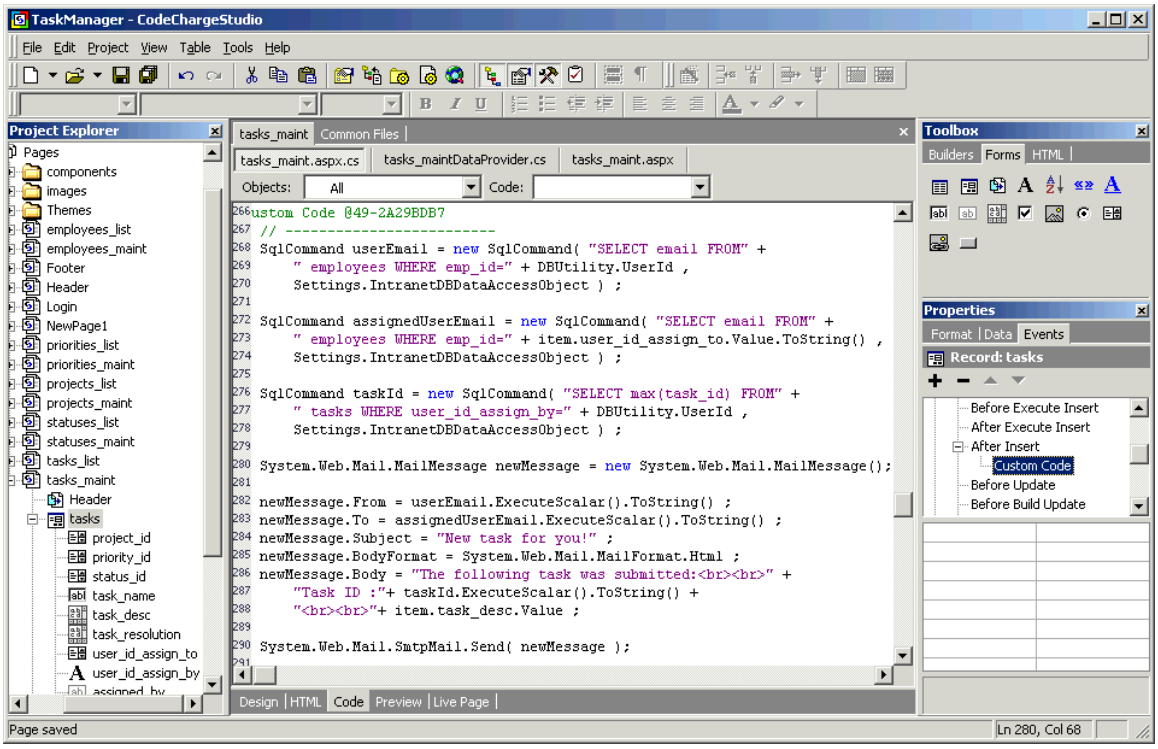
Creates a *MailMessage* object, which is under the namespace *System.Web.Mail*.

```
newMessage.From = userEmail.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 ;
```

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.Send( newMessage );
```

Finally the static “Send” method of the class *SmtpMail* is called to send the email.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
Pages
components
images
Themes
employees_list
employees_maint
Footer
Header
Login
NewPage1
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolution
user_id_assign_to
user_id_assign_by
tasks_maint
Common Files
tasks_maint.aspx.cs
tasks_maintDataProvider.cs
tasks_maint.aspx
Objects:
All
Code:
ustom Code @49-2A29BDB7
// -------------------------
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:

" +
"Task ID :"+ taskId.ExecuteScalar().ToString() +
"

"+ item.task_desc.Value ;
System.Web.Mail.SmtpMail.Send(newMessage);
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Record: tasks
Before Execute Insert
After Execute Insert
After Insert
Custom Code
Before Update
Before Build Update
Page saved
Ln 280, Col 68

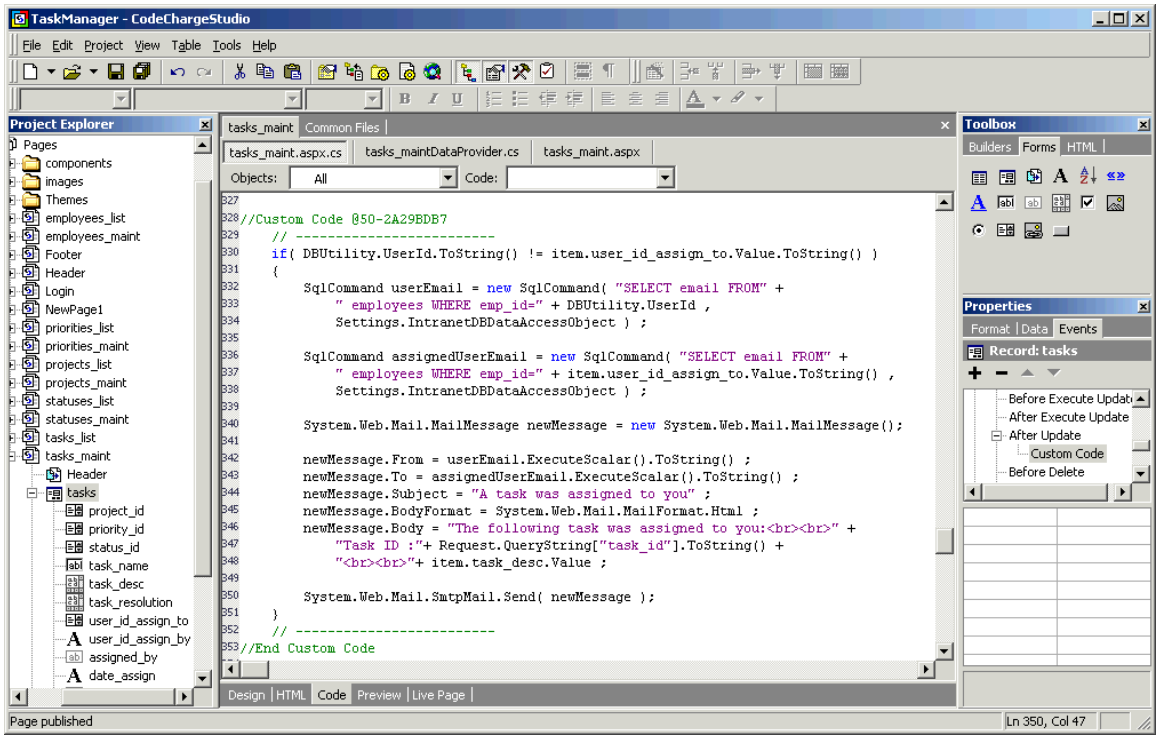
Use the "After Update" Event to Send Emails

You've previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. Now implement similar functionality in "After Update" Event to notify assignee when an existing task is updated and reassigned to someone.
Click on the "tasks" form in the "Project Explorer", then in the "Properties" window, select the "Events" tab, and then add the following "Custom Code" in "After Update" event:

```
if( 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.Send( newMessage );

}
```

The main differences between the above code and the one you used in "After Insert" event are as follows:

- An "if" condition was added to send an email only if a user assigns a task to someone other than himself/herself
- *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***


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
Pages
components
images
Themes
employees_list
employees_maint
Footer
Header
Login
NewPage1
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolution
user_id_assign_to
user_id_assign_by
assigned_by
date_assign
tasks_maint Common Files
tasks_maint.aspx.cs
tasks_maintDataProvider.cs
tasks_maint.aspx
Objects: All
Code:
//Custom Code @50-2A29BDB7
// -------------------------
if(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:

" +
"Task ID :"+ Request.QueryString["task_id"].ToString() +
"

"+ item.task_desc.Value ;
System.Web.Mail.SmtpMail.Send(newMessage);
}
// -------------------------
//End Custom Code
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Record: tasks
Before Execute Update
After Execute Update
After Update
Custom Code
Before Delete
Page published
Ln 350, Col 47

Test Email Delivery

Before testing the system, you should add new users to your database with correct email addresses, or modify the existing test users by changing their email address. You can do this by opening the Intranet.mdb database that is located in your project directory.
(Note: You will need at least MS Access 2000 to manually edit the database file.)
Alternatively, you may use the Task Manager itself and go to the Employees page to view and modify user emails there.

Once you have users configured with test emails, save your project and switch to Live Page mode to test your system. 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.

Implement Record Security in "Before Show" 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 explain several of them.

First, click on the "tasks_maint" page in the "Project Explorer", then select "Events" tab in the "Properties" window. Add "Custom Code" to the "Before Show" event of the page. This method directly maps to the *Page_Load* method of ASP.NET.

Once in the Code view, replace the generated comment:

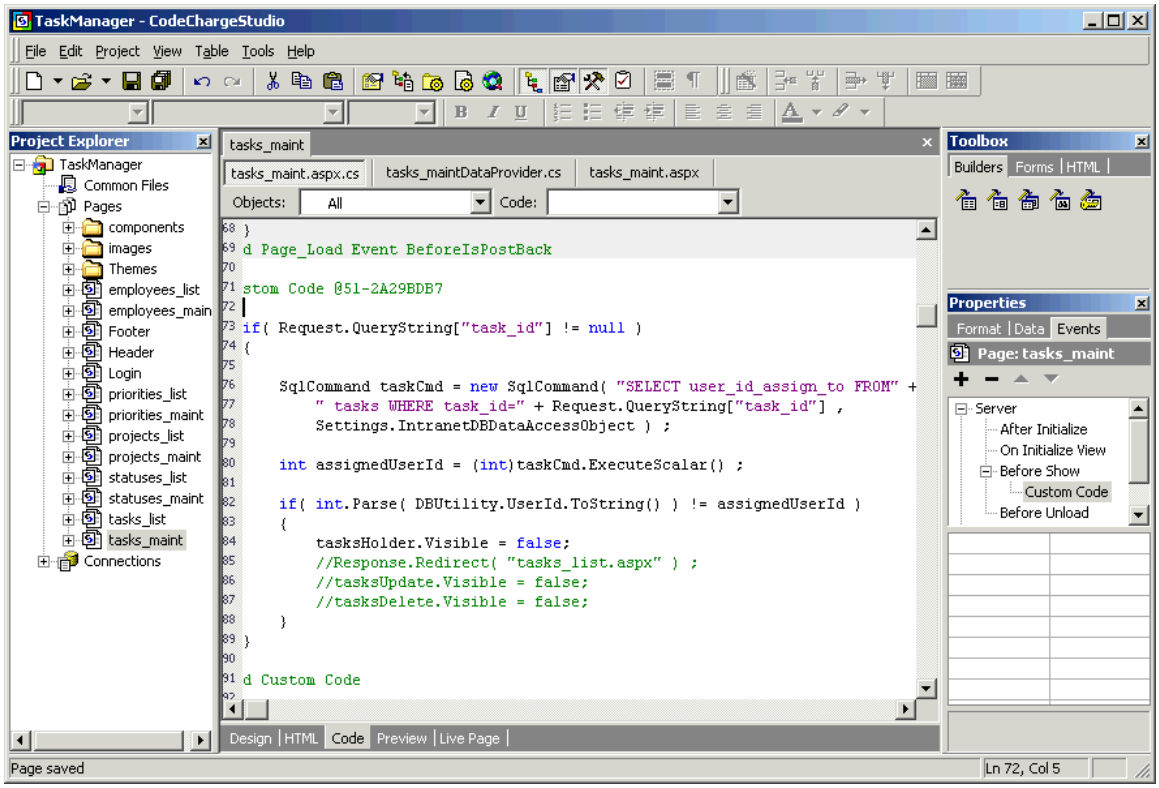
```
// -------------------------
// Write your own code here.
// -------------------------
```

with the code below:

```
if( Request.QueryString["task_id"] != null )
{

   SqlCommand taskCmd = new SqlCommand( "SELECT user_id_assign_to FROM" +
      " tasks WHERE task_id=" + Request.QueryString["task_id"] ,
      Settings.IntranetDBDataAccessObject ) ;

   int assignedUserId = (int)taskCmd.ExecuteScalar() ;

   if( int.Parse( DBUtility.UserId.ToString() ) != assignedUserId )
   {
      tasksHolder.Visible = false;
      //Response.Redirect( "tasks_list.aspx" ) ;
      //tasksUpdate.Visible = false;
      //tasksDelete.Visible = false;
   }
}
```

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

1. **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 assures that all users can create new tasks. You can test this functionality by inserting the 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 opposite, mutually exclusive criteria.



2. **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:

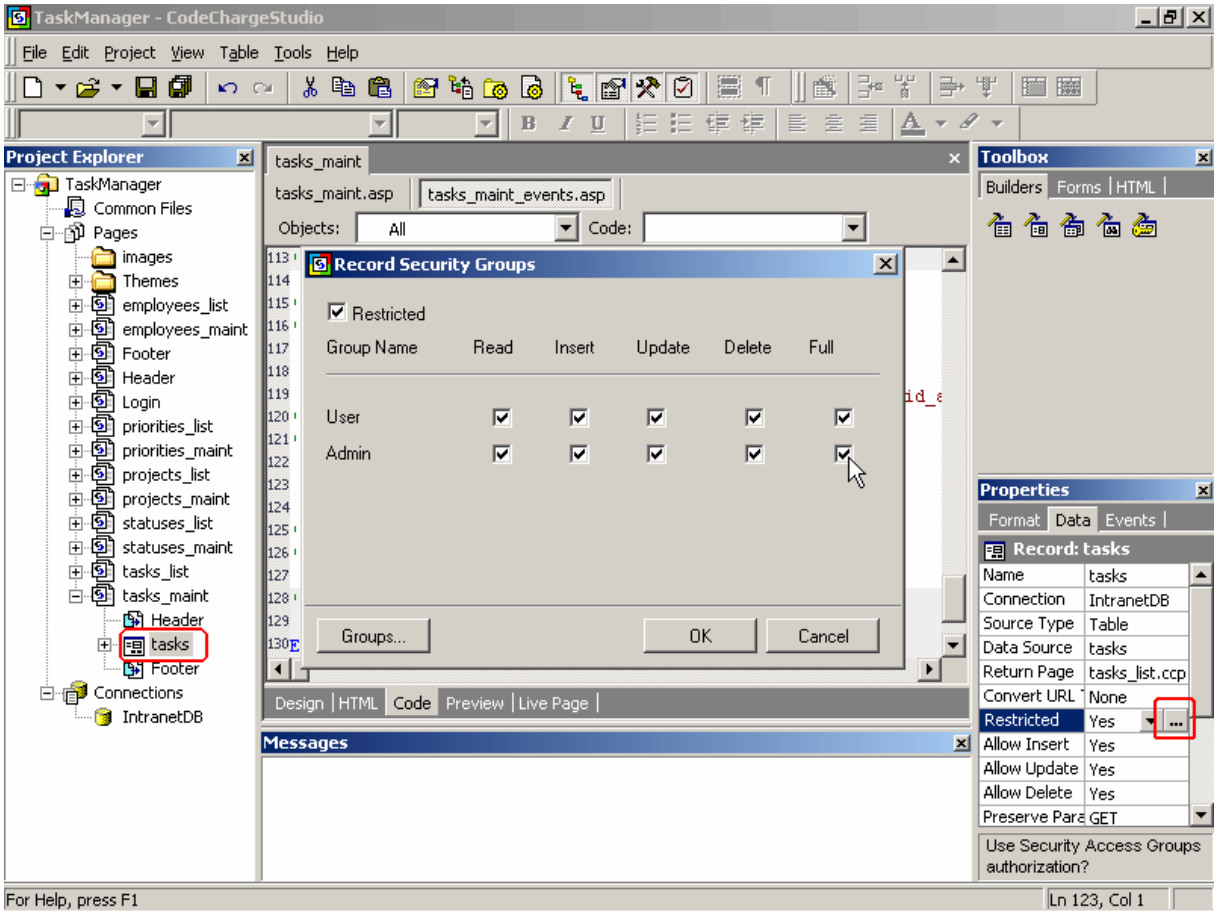
```
if( Request.QueryString["task_id"] != null )
{

   SqlCommand taskCmd = new SqlCommand( "SELECT user_id_assign_to FROM" +
      " tasks WHERE task_id=" + Request.QueryString["task_id"] ,
      Settings.IntranetDBDataAccessObject ) ;

   int assignedUserId = (int)taskCmd.ExecuteScalar() ;

   if( int.Parse( DBUtility.UserId.ToString() ) != assignedUserId )
   {
      // tasksHolder.Visible = false;
      Response.Redirect( "tasks_list.aspx" ) ;
      //tasksUpdate.Visible = false;
      //tasksDelete.Visible = false;
   }
}
```

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.

3. **Disable Update/Submit and Delete buttons for unauthorized users.**
Comment out the “Response.Redirect” statement and uncomment the two lines of code below it, as shown here:

```
if( Request.QueryString["task_id"] != null )
{
   SqlCommand taskCmd = new SqlCommand( "SELECT user_id_assign_to FROM" +
      " tasks WHERE task_id=" + Request.QueryString["task_id"] ,
      Settings.IntranetDBDataAccessObject ) ;
   int assignedUserId = (int)taskCmd.ExecuteScalar() ;

   if( int.Parse( DBUtility.UserId.ToString() ) != assignedUserId )
   {
      // tasksHolder.Visible = false;
      // Response.Redirect( "tasks_list.aspx" ) ;
      tasksUpdate.Visible = false;
      tasksDelete.Visible = false;
   }
}
```

The above code shows how you can manipulate the *Visible* property of the *tasksUpdate* and *tasksDelete* buttons on the record form. This property controls the appearance of the “Update” and “Delete” buttons on the page. If set to *false*, the buttons will not appear. Additional security needs to be implemented to make it not possible to update the record even if someone saves the page, adds the missing buttons and submits the page externally. You also set the “Restricted” property of your form to “Yes” and assign permissions to everyone from the “Properties Explorer”. Restricting page access indicates that certain users are not allowed to add, update or delete records.



Enhancing Application Functionality with Programming Events (JSP)

You've probably noticed that until now you've built your Task Management application without having to deal with the programming code. CodeCharge Studio can help you build fairly functional systems without any programming, however creating more sophisticated systems will require a certain amount of programming code. Fortunately, CodeCharge Studio makes programming easy by providing you with a first-rate code editor, in addition to Events and Actions that aid you in inserting pre-programmed snippets of code into the right place within the program.

Here are the definitions of Action and Event:

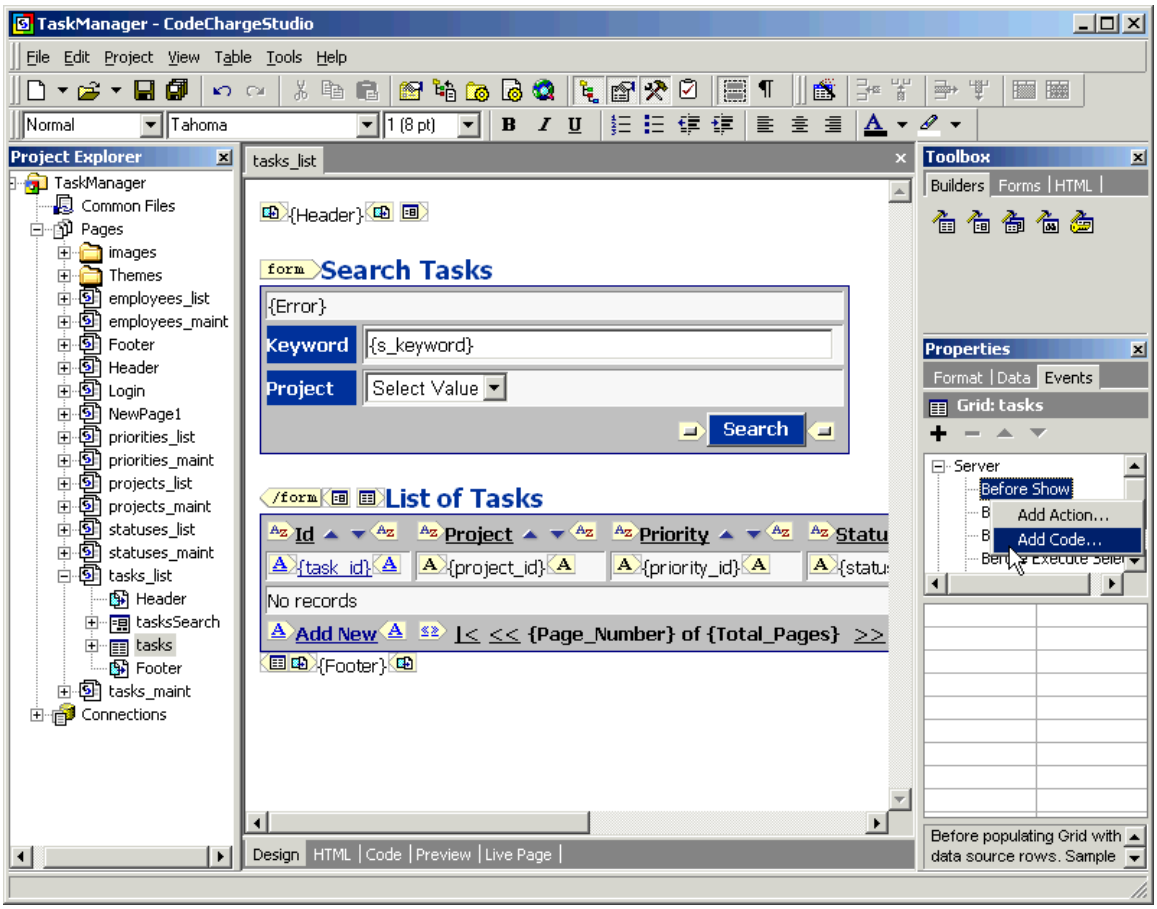
Action
User-definable code generation component, which inserts block of code into an event procedure. CodeCharge Studio comes with several predefined Actions, which are installed into the following folder:
(CCS folder)\Components\Actions
Internally, actions consist of XML and XSL code that can be easily customized.
For example an action can be set on a TextBox to validate the e-mail address.

Event Procedure
A procedure automatically executed in response to an event initiated by the program code during execution. Events are the best place for putting custom programming code.

Use "Before Show" Event to Build Custom Variables

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, you will mark the listed tasks assigned to you (the current user) by showing your name in *blue* color within the grid.
The logic for this task will be first retrieving the name of the user based on his user id from the database and storing it in a page level variable. Next you will check the *assigned_to* field of the grid for equality with the earlier retrieved name and modify the HTML generated respectively.

Expand the *tasks_list* page in *Project Explorer* window, right-click on the *tasks* grid and select "Properties" to display the properties for the *tasks* grid. Select the "Events" tab in the *Properties* window, then right-click on the "Before Show" event and select "Add Code…". By adding code into this event, you can build the variables you might require while manipulating the display of the grid.



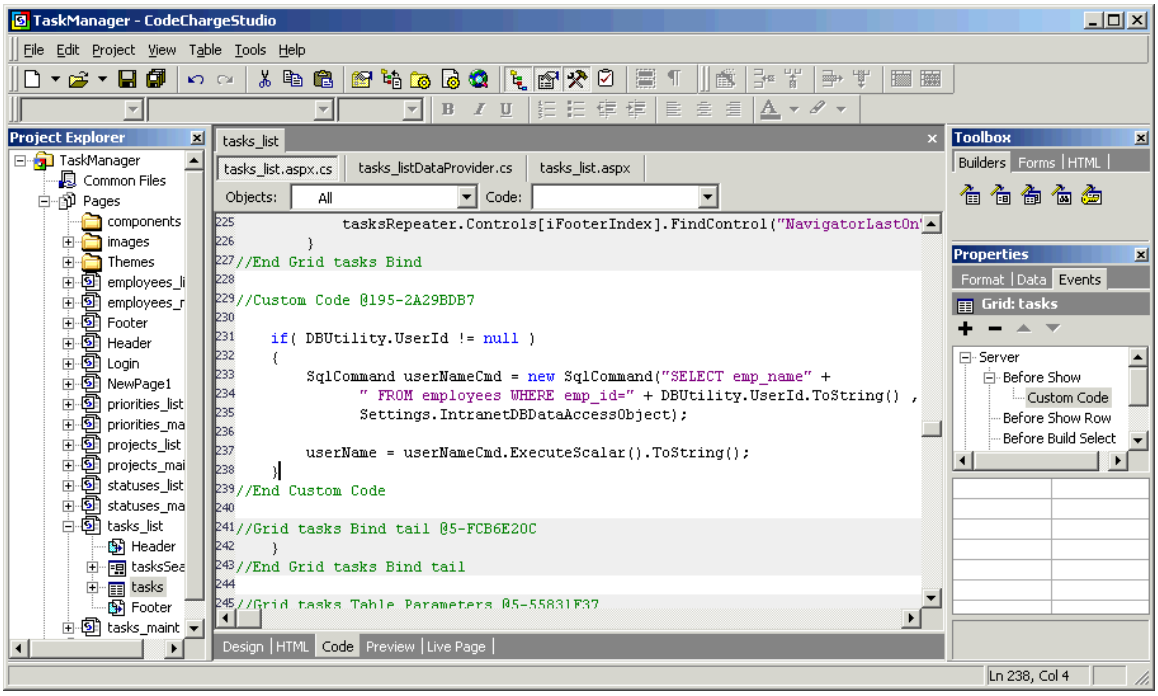
Retrieve a Custom Field From the Database

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

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

with the following lines (JSP):

```
SessionStorage s = SessionStorage.getInstance(e.getPage().getRequest());
String uid = s.getAttributeAsString("UserID");
if (uid != null && uid.length() != 0) {
   PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
   Hashtable result = ds.getOneRow("select emp_name from employees where emp_id=" + uid);
   ds.closeConnection();
   if (result != null) {
      s.setAttribute("UserName", result.get("emp_name"));
   }
}
```



Let's see how the code above works :-

```
SessionStorage s = SessionStorage.getInstance(e.getPage().getRequest());
String uid = s.getAttributeAsString("UserID");
if (uid != null && uid.length() != 0)
```

This is an “if” statement which is *true* only if the variable “uid” is not *null*. The variable “uid” holds the user id of the current user and it is set automatically when a user logs in. The following are all default session variables created by CodeCharge Studio:

UserID – the primary key field value of the logged in user
GroupID – security level/group of the user currently logged into the system.

Hence this statements below the “if” block will only execute if the user has been authenticated.

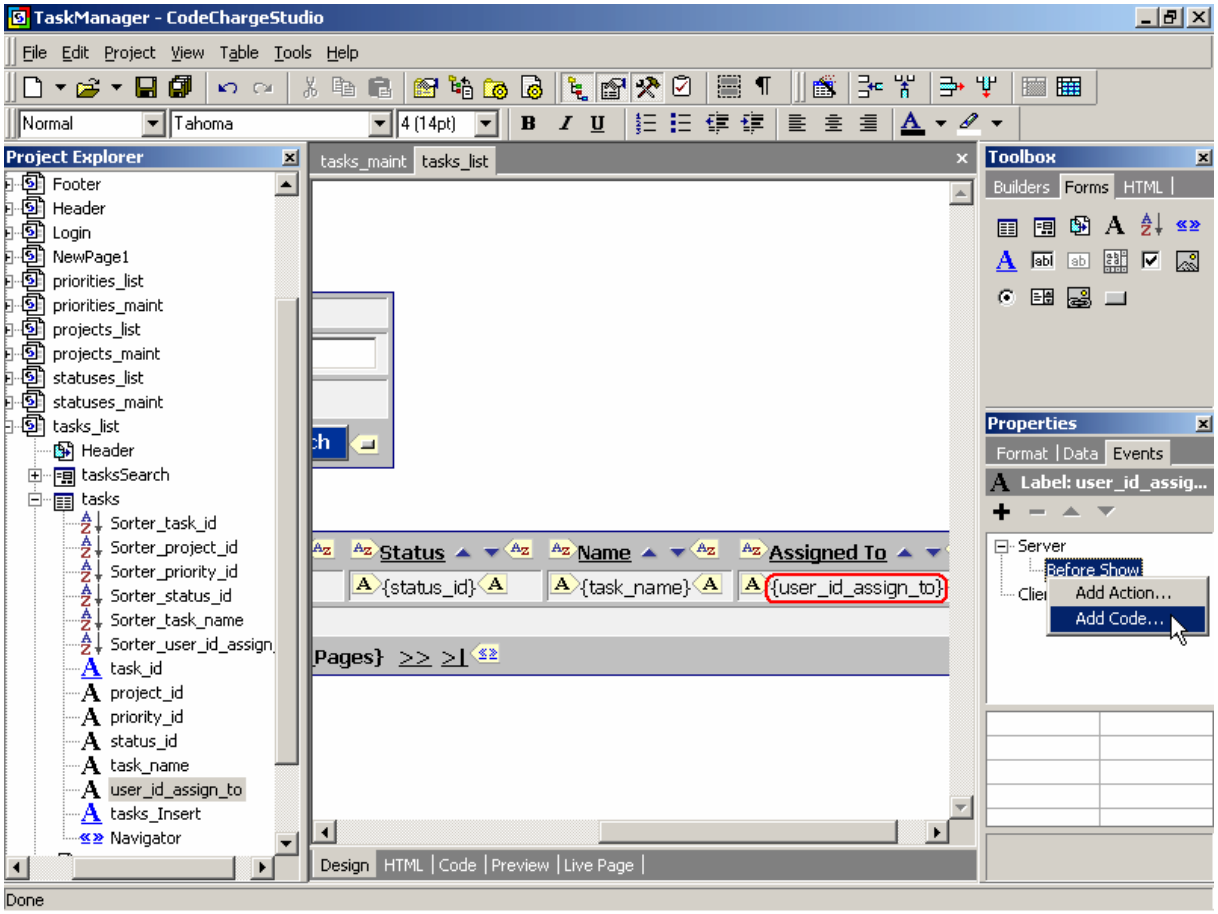
```
PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
Hashtable result = ds.getOneRow("select emp_name from employees where emp_id=" + uid);
ds.closeConnection();
if (result != null) {
   s.setAttribute("UserName", result.get("emp_name"));
}
```

The above statement uses the existing connection pool of the Intranet connection to retrieve the name of the employee identified by “uid”. Then the “setAttribute” method is used to create a session variable named “UserName” and the employee’s name is assigned to it.

Use Field's "Before Show" Event to Alter Grid's Output

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.

Open the *tasks_list* page in the "Project Explorer", expand the *tasks* grid, and right-click on the *user_id_assign_to* field and select "Properties". Under the "Data" tab, set the value of the "Content" property to "HTML". Next, select the "Events" tab in the Properties window, then 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.



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.
Replace the following line of code:

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

with the following:

```
SessionStorage s = SessionStorage.getInstance(e.getPage().getRequest());
String uname = s.getAttributeAsString("UserName");
Control cntrl = (Control)e.getSource();
String value = cntrl.getFormatedValue();
if (value.equals(uname)) {
   cntrl.setFormatedValue("<b><font color=blue>" + value + "</font></b>");
}
```

Let's explain how the above JSP code works:

```
SessionStorage s = SessionStorage.getInstance(e.getPage().getRequest());
String uname = s.getAttributeAsString("UserName");
Control cntrl = (Control)e.getSource();
String value = cntrl.getFormatedValue();
if (value.equals(uname))
```

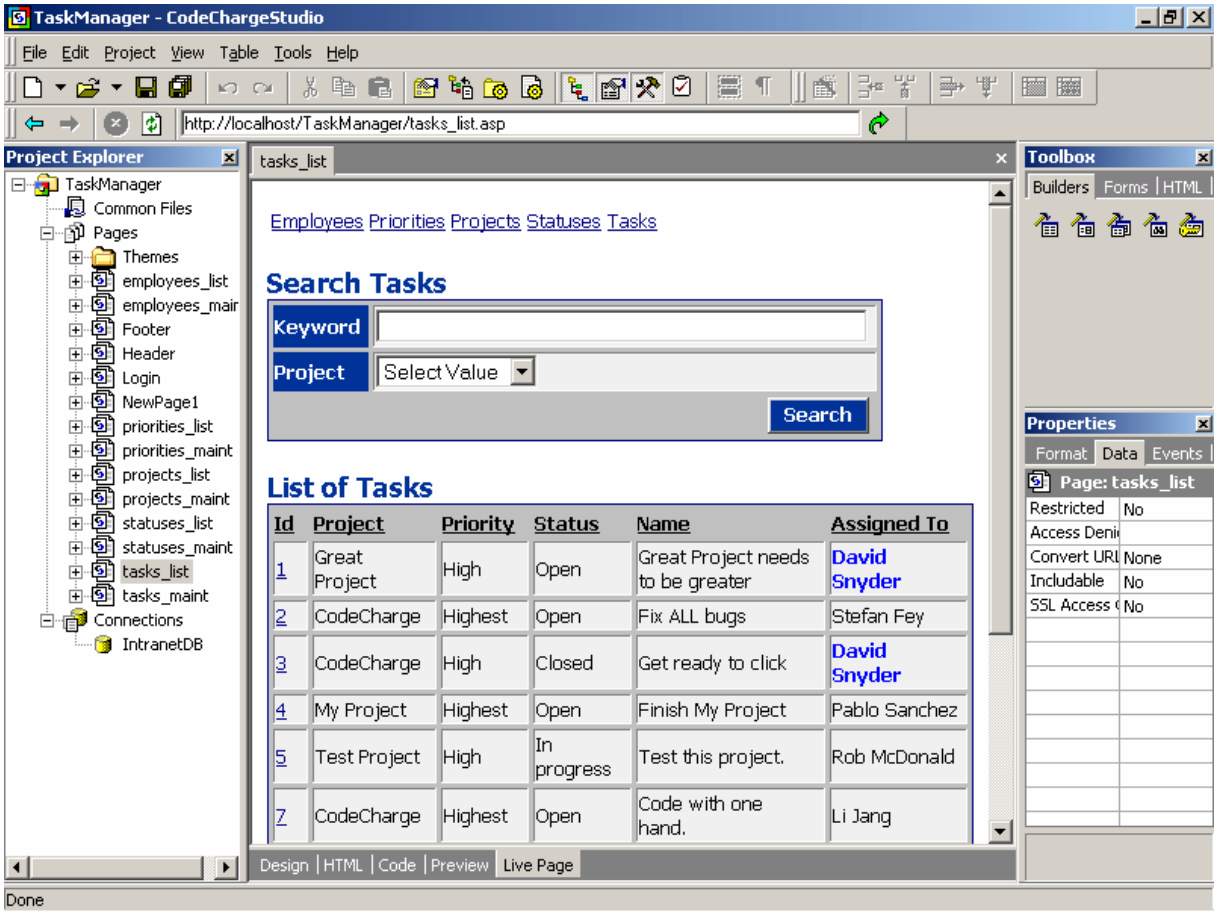
The above lines prepare and then execute the "if" condition that checks if the value of the current control (*user_id_assign_to)* identified as "e" is equal to the user name stored in the session variable "UserName".

```
cntrl.setFormatedValue("<b><font color=blue>" + value + "</font></b>");
```

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 name value wrapped within HTML code that specifies the font color as blue, and adds HTML <*b*> tag to make the font bold as well.

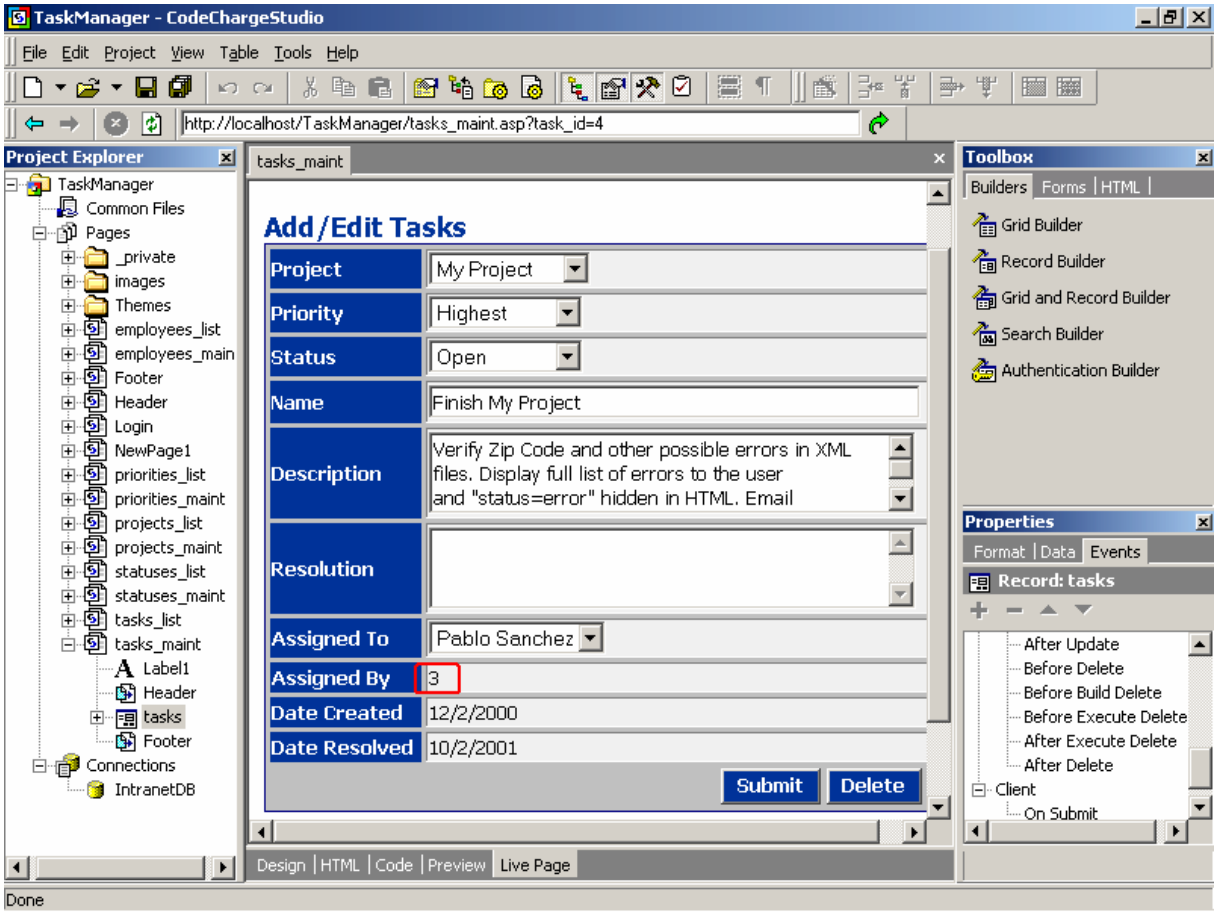
Preview the Tasks List Page

Save your project and 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.
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. Click on any of the project Ids and you should see the login page.
Login as davids / davids , then 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.



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.



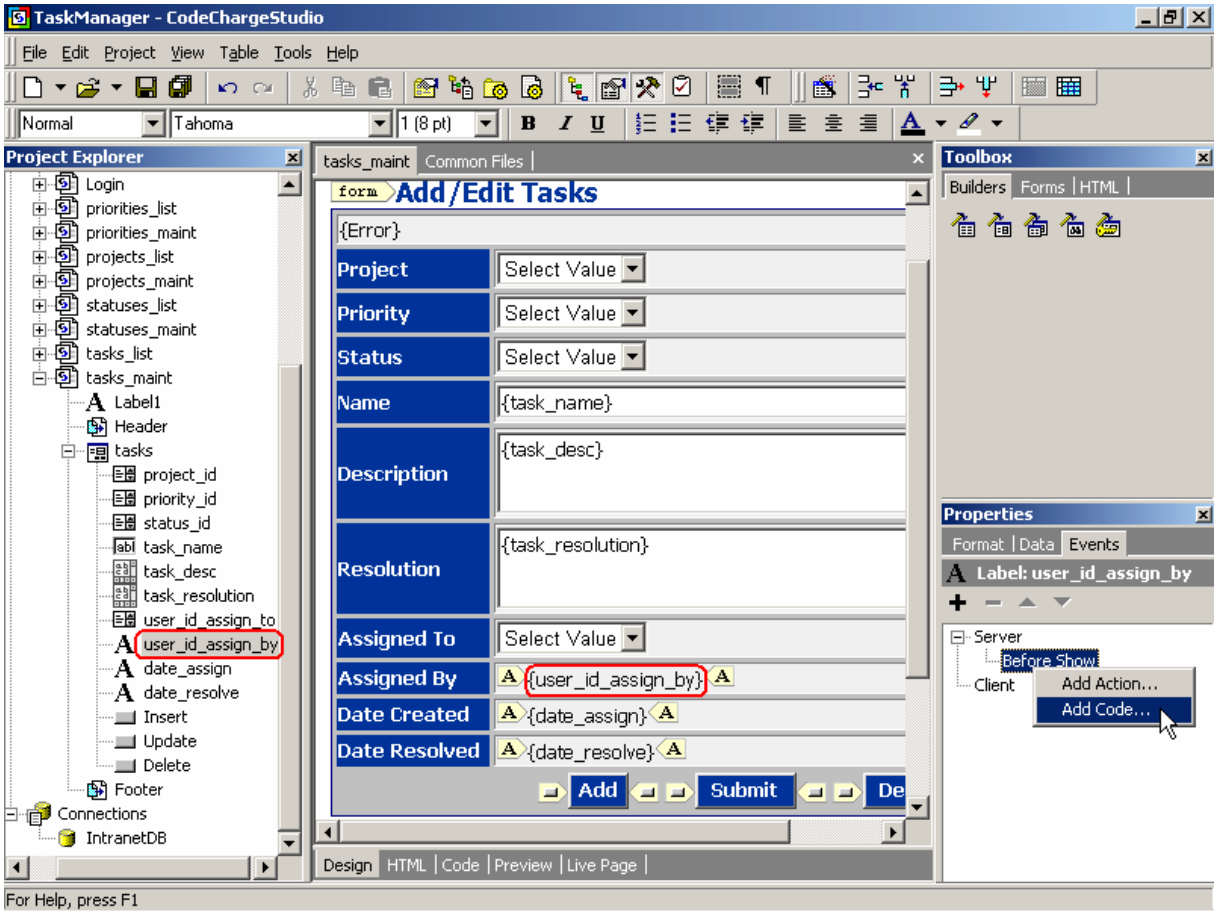
There are several potential methods of dealing with the above issue and let's explain each one in detail:

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.

Create “Before Show” Event to Alter Label’s Value

Select the Label *user_id_assign_by* in the Design mode, then in the “Properties” window click on “Events” tab.
Right-click on “Before Show” event and select “Add Code…”.
CodeCharge Studio should automatically switch to Code view at that time.
“Before Show” event is a place in the program that is executed after a value is assigned to the control, but before such value is displayed.



Use "Before Show" Event to Alter Label's Value

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

```
//user_id_assign_by Label Handler Head @13-91DFA71C-->
  public class tasksuser_id_assign_byLabelHandler implements LabelListener {
    public void beforeShow(CCSEvent e) {
//End user_id_assign_by Label Handler Head-->

//Event BeforeShow Action Custom Code @13-4EFC2132-->
/* Write your own code here. */
//End Event BeforeShow Action Custom Code-->

//user_id_assign_by Label Handler Tail @13-F5FC18C5-->
    }
  }
//End user_id_assign_by Label Handler Tail-->
```

Replace the text:

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

with the following:

```
Control cntrl = (Control)e.getSource();
if (((Record)cntrl.getParent()).getMode() == com.codecharge.components.Record.UPDATE_MODE) {
   String value = cntrl.getFormatedValue();
   PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
   Hashtable result = ds.getOneRow("select emp_name from employees where emp_id=" + value);
   if (result != null) {
      cntrl.setFormatedValue((String)result.get("emp_name"));
   }
   ds.closeConnection();
   } else {
      String value = (String)SessionStorage.getInstance(e.getPage().getRequest()).getAttribute("UserID");
      PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
      Hashtable result = ds.getOneRow("select emp_name from employees where emp_id=" + value);
      if (result != null) {
         cntrl.setFormatedValue((String)result.get("emp_name"));
      }
   ds.closeConnection();
}
```

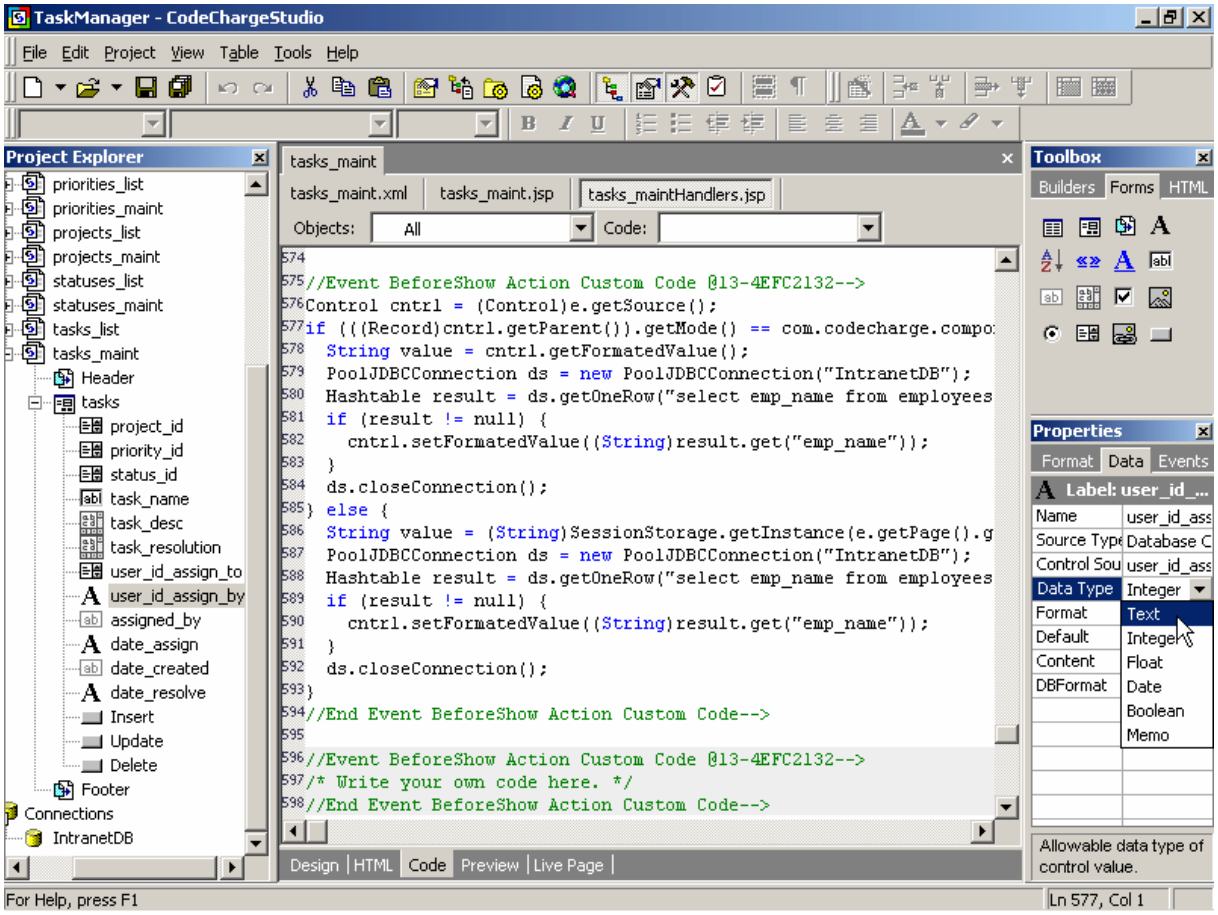
In the above code, the first "if" block checks if the current record form is in the update mode and if so, it retrieves the name of of the appropriate employee and assigns it to the current control's value. If the form is not in the update mode, the name of the currently logged in user is assigned as 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**.
getFormatedValue and ***setFormatedValue*** – 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).
getMode – 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).
getOneRow –a method that retrieves a database value based on an sql query. Here, this method retrieves the Employee Name (*emp_name*) from the *employees* table under 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.
SessionStorage.getInstance(e.getPage().getRequest()).getAttribute("UserID") – 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.
"

Refer to the CodeCharge Studio Programming Reference for more information about functions and properties available in CodeCharge-generated programs.

Now that you programmatically modified the value of the *tasksuser_id_assign_by* Label to output employee name instead of the id, you will also need to specify that this field is now a Text field, instead of Numeric. Click on “Data” tab in “Properties” window, and select “Text” as the Data Type.



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

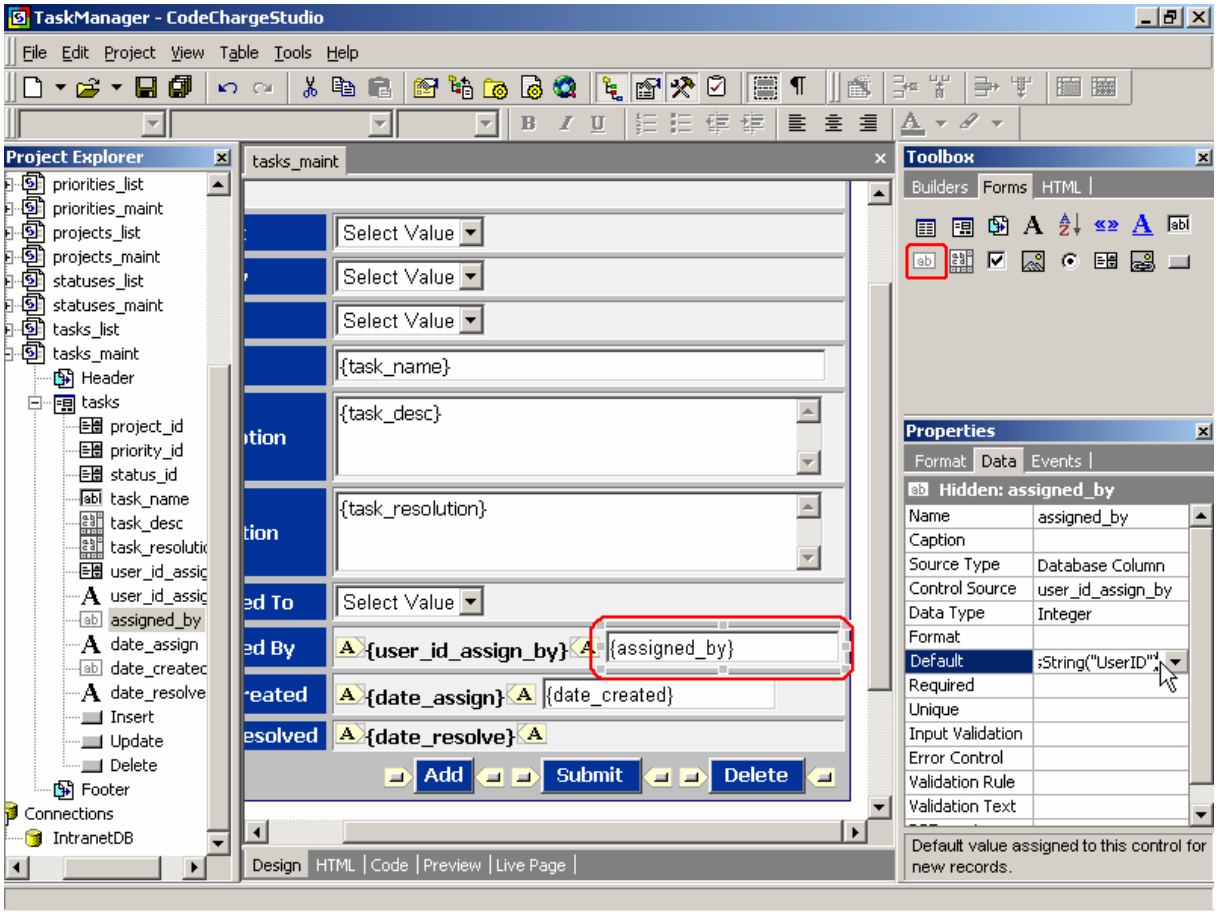
You've previously used the "Before Show" event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though employee's name is displayed on the page; it is not written to the database. Since we want the database to record the name or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

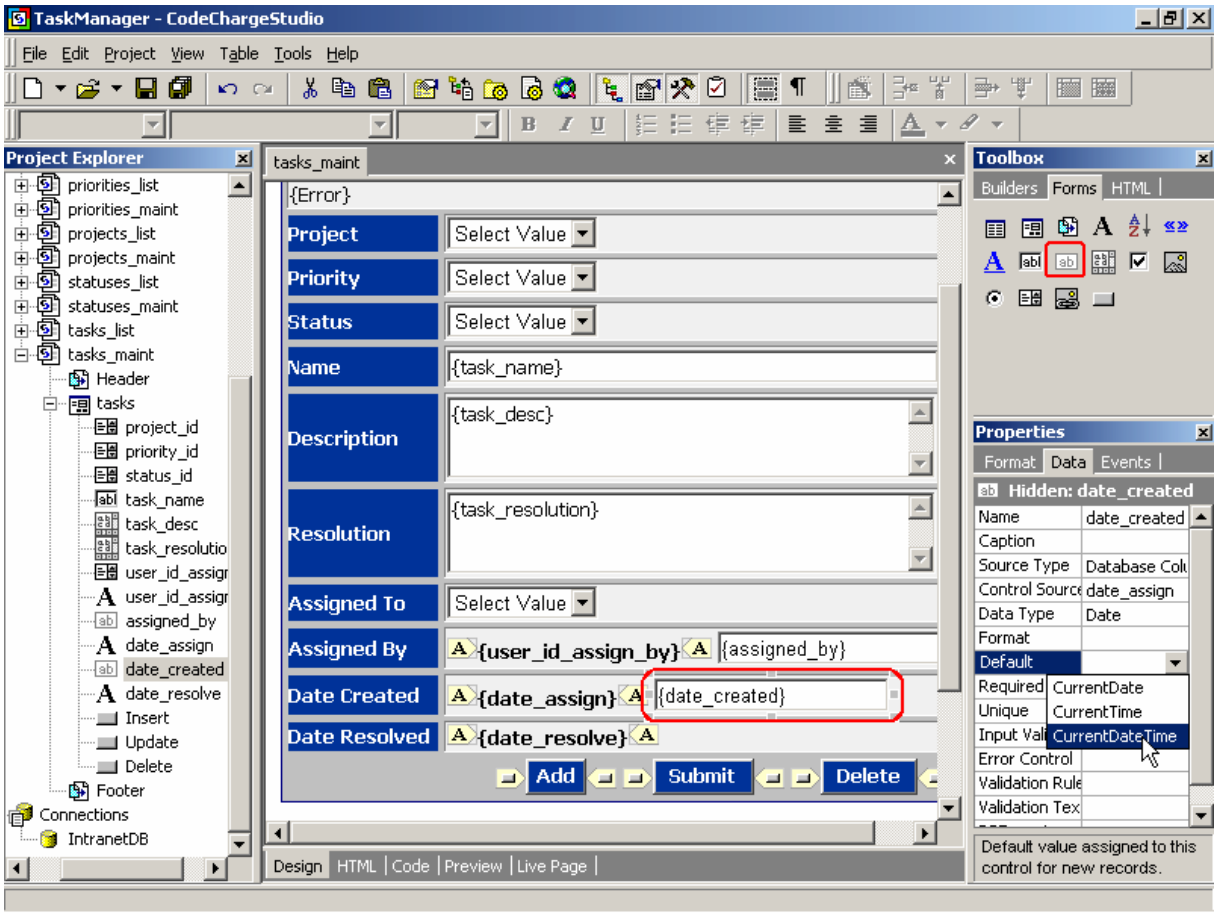
Default: *SessionStorage.getInstance(e.getPage().getRequest()).getAttributeAsString("UserID")* – default value for this field if empty. This code retrieves the user id of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
user_id_assig
user_id_assig
assigned_by
date_assign
date_created
date_resolve
Insert
Update
Delete
Footer
Connections
IntranetDB
tasks_maint
Select Value
Select Value
Select Value
{task_name}
{task_desc}
{task_resolution}
Select Value
{user_id_assign_by}
{assigned_by}
{date_assign}
{date_created}
{date_resolve}
Add
Submit
Delete
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Hidden: assigned_by
Name assigned_by
Caption
Source Type Database Column
Control Source user_id_assign_by
Data Type Integer
Format
Default ;String("UserID"
Required
Unique
Input Validation
Error Control
Validation Rule
Validation Text
Default value assigned to this control for new records.

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.
Configure the new field as follows:

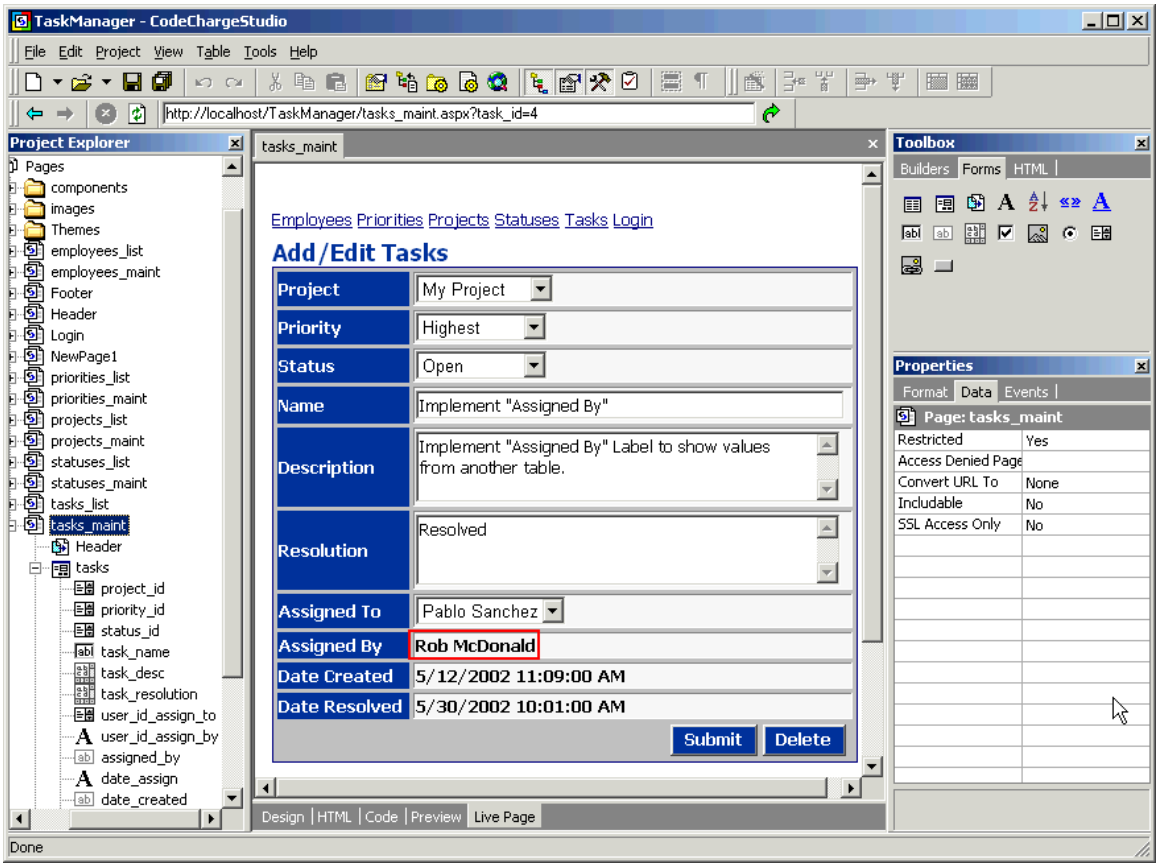
Name: *date_created*
Control Source: *date_assign*
Data Type: *Date*
Default: *CurrentDateTime* – The "CurrentDateTime" property allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.



Test the Label and Hidden Fields

Finally, you can switch to Live Page mode, select a Task, Login, and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.
Don't forget to save it!



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 explain several of them.

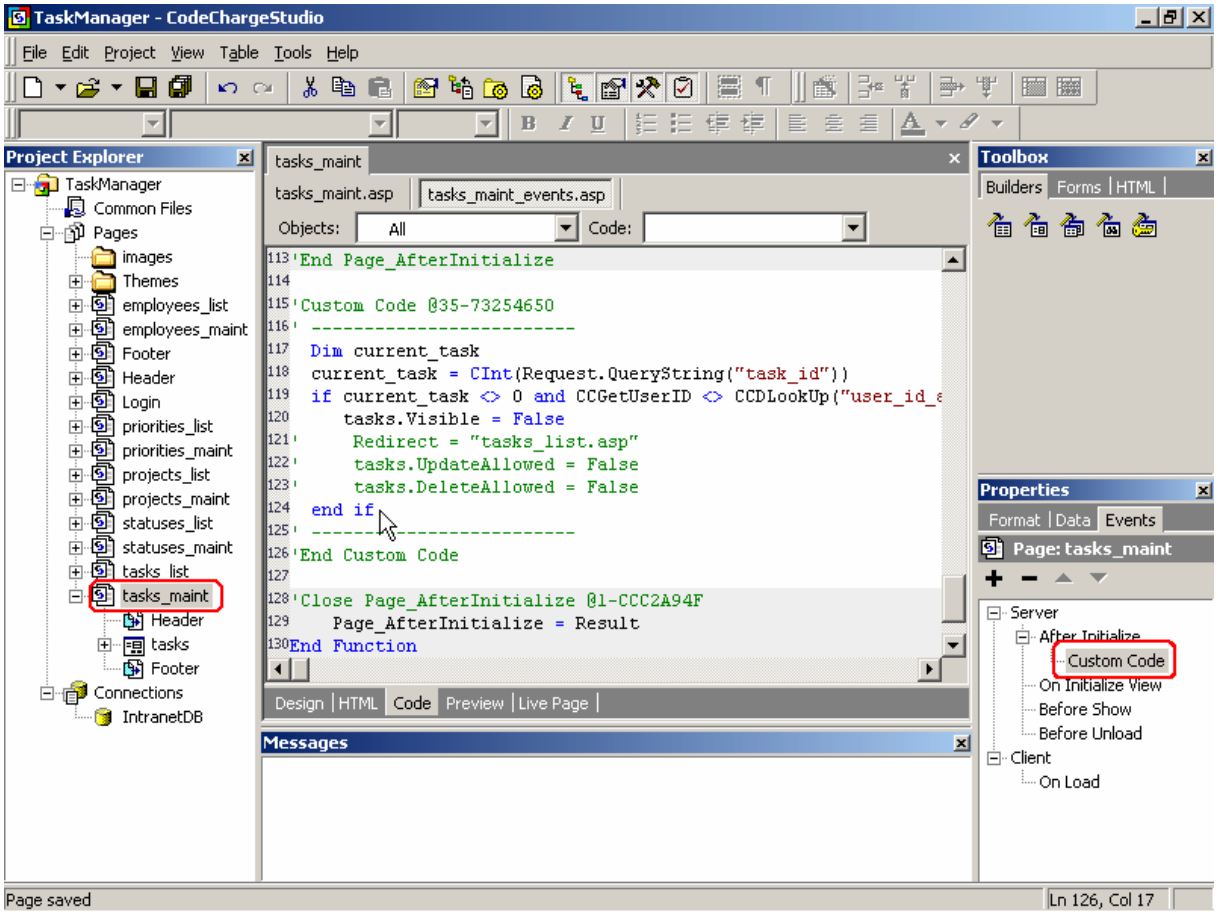
First, click on the "tasks_maint" page in the Project Explorer, then select "Events" tab in the Properties window. Add "Custom Code" to the "After Initialize" event of the page.

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 = (String)SessionStorage.getInstance(e.getPage().getRequest()).getAttribute("UserID");
   PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
   int task_uid = ds.count("select user_id_assign_to from tasks where task_id=" + currentTask);
   if (!uid.equals( String.valueOf(task_uid) )) {
      ((Record)e.getSource()).setVisible( false );
//   ((Page)e.getPage()).setRedirect("tasks_list.jsp");
//   ((Record)e.getSource()).setAllowUpdate( false);
//   ((Record)e.getSource()).setAllowDelete( false);
   }
   ds.closeConnection();
}
```



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

1. **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.

2. **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 = (String)SessionStorage.getInstance(e.getPage().getRequest()).getAttribute("UserID");
   PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
   int task_uid = ds.count("select user_id_assign_to from tasks where task_id=" + currentTask);
   if (!uid.equals( String.valueOf(task_uid) )) {
//   ((Record)e.getSource()).setVisible( false );
      ((Page)e.getPage()).setRedirect("tasks_list.jsp");
//   ((Record)e.getSource()).setAllowUpdate( false);
//   ((Record)e.getSource()).setAllowDelete( false);
   }
   ds.closeConnection();
}
```

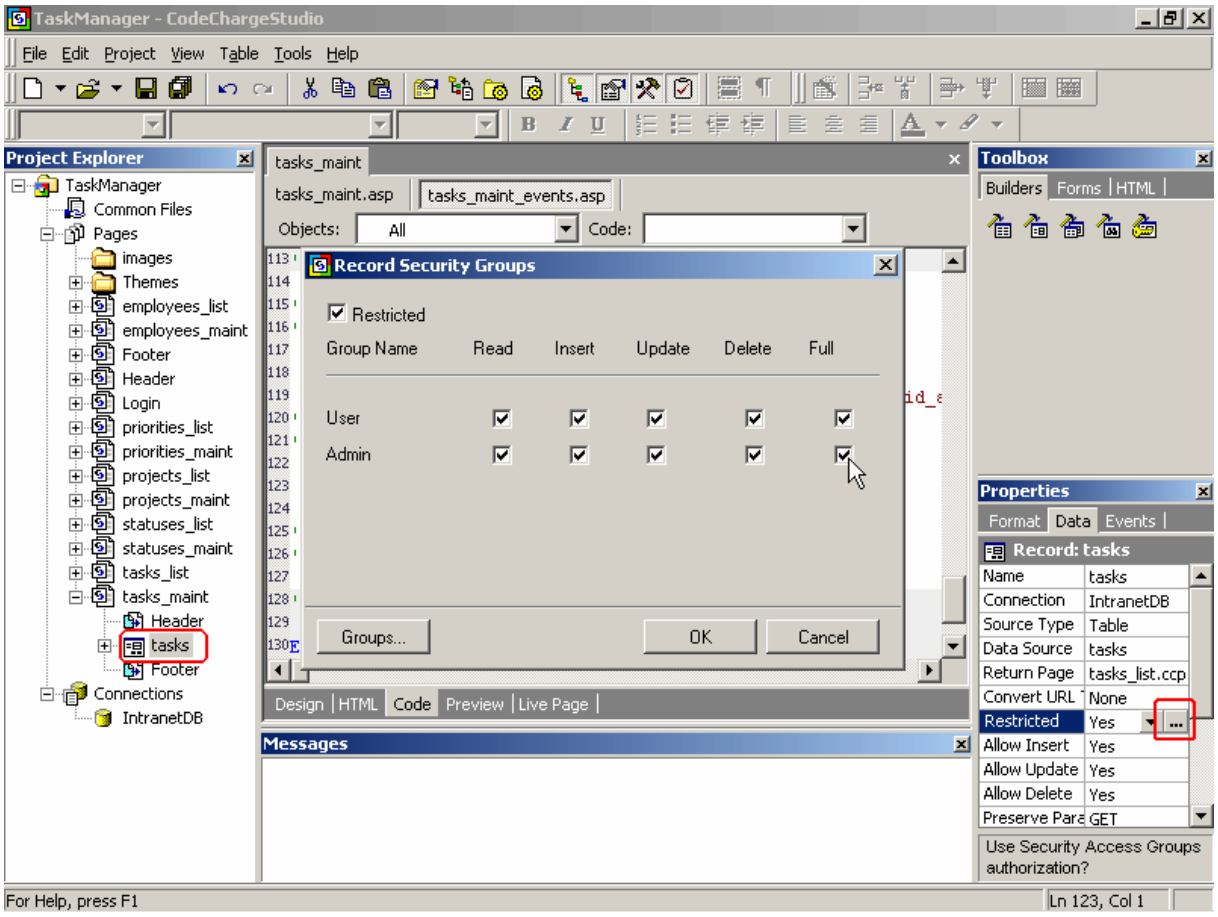
The above code shows that you should comment out the previously active line, and uncomment the line that contains “setRedirect”.
set*Redirect* is a method used by CodeCharge Studio to determine if the current page should be redirected to another page, for example if a user is not logged in. This variable can be used only on pages that have restricted access and require users to login. You can simply assign the destination page using the *setRedirect* method 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.

3. **Disable Update/Submit and Delete buttons for unauthorized users.**
Comment out the “Redirect” statement and uncomment the two lines of code below it, as shown here:

```
String currentTask = e.getPage().getHttpGetParams().getParameter("task_id");
if (!StringUtils.isEmpty(currentTask)) {
   String uid = (String)SessionStorage.getInstance(e.getPage().getRequest()).getAttribute("UserID");
   PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
   int task_uid = ds.count("select user_id_assign_to from tasks where task_id=" + currentTask);
   if (!uid.equals( String.valueOf(task_uid) )) {
//   ((Record)e.getSource()).setVisible( false );
//   ((Page)e.getPage()).setRedirect("tasks_list.jsp");
      ((Record)e.getSource()).setAllowUpdate( false);
      ((Record)e.getSource()).setAllowDelete( false);
   }
   ds.closeConnection();
}
```

This code shows how you can manipulate the *AllowUpdate* and *AllowDelete* properties of a record form. These properties control the appearance of the “Update” and “Delete” buttons on the page. If set to False, the buttons will not appear. Additional security is implemented to make it impossible to update the record even if someone

saves the page, adds the missing buttons and submits the page externally. However, the above code won't work unless you also set the "Restricted" property of your form to "Yes" and assign permissions to everyone. That's because CodeCharge Studio generates the code appropriate for hiding and disabling buttons only when there is a need to do so, and restricting page access indicates that certain users are not allowed to add, update or delete records.



4. One more, although less secure method of disabling buttons on the record form is to hide the Update and Delete buttons on the page by adding the following custom code to the "Before Show" event of the form:

```
String currentTask = e.getPage().getHttpGetParams().getParameter("task_id");
if (currentTask != null) {
  String uid = (String)SessionStorage.getInstance(e.getPage().getRequest()).getAttribute("UserID");
  PoolJDBCConnection ds = new PoolJDBCConnection("IntranetDB");
  int task_uid = ds.count("select user_id_assign_to from tasks where task_id=" + currentTask);
     if (!uid.equals( String.valueOf(task_uid) )) {
       ((Record)e.getSource()).getChild("Update").setVisible(false);
       ((Record)e.getSource()).getChild("Delete").setVisible(false);
     }
  ds.closeConnection();
}
```

The above code shows how you to manipulate the *Visible* and property of the Buttons on a record form, thus hiding them if needed. If *Visible* is set to False, the buttons will not appear on the page.
You may also add additional lines of code that will prevent updates from being executed even if someone saves the page on their local harddrive and manually adds the "hidden" buttons: Here is the code that achieves this:

```
((Record)e.getSource()).setAllowUpdate( false);
((Record)e.getSource()).setAllowDelete( false);
```

Enhancing Application Functionality with Programming Events (PHP)

You've probably noticed that until now you've built your Task Management application without having to deal with the programming code. CodeCharge Studio can help you build fairly functional systems without any programming, however creating more sophisticated systems will require a certain amount of programming code. Fortunately, CodeCharge Studio makes programming easy by providing you with a first-rate code editor, in addition to Events and Actions that aid you in inserting pre-programmed snippets of code into the right place within the program.

Here are the definitions of Action and Event:

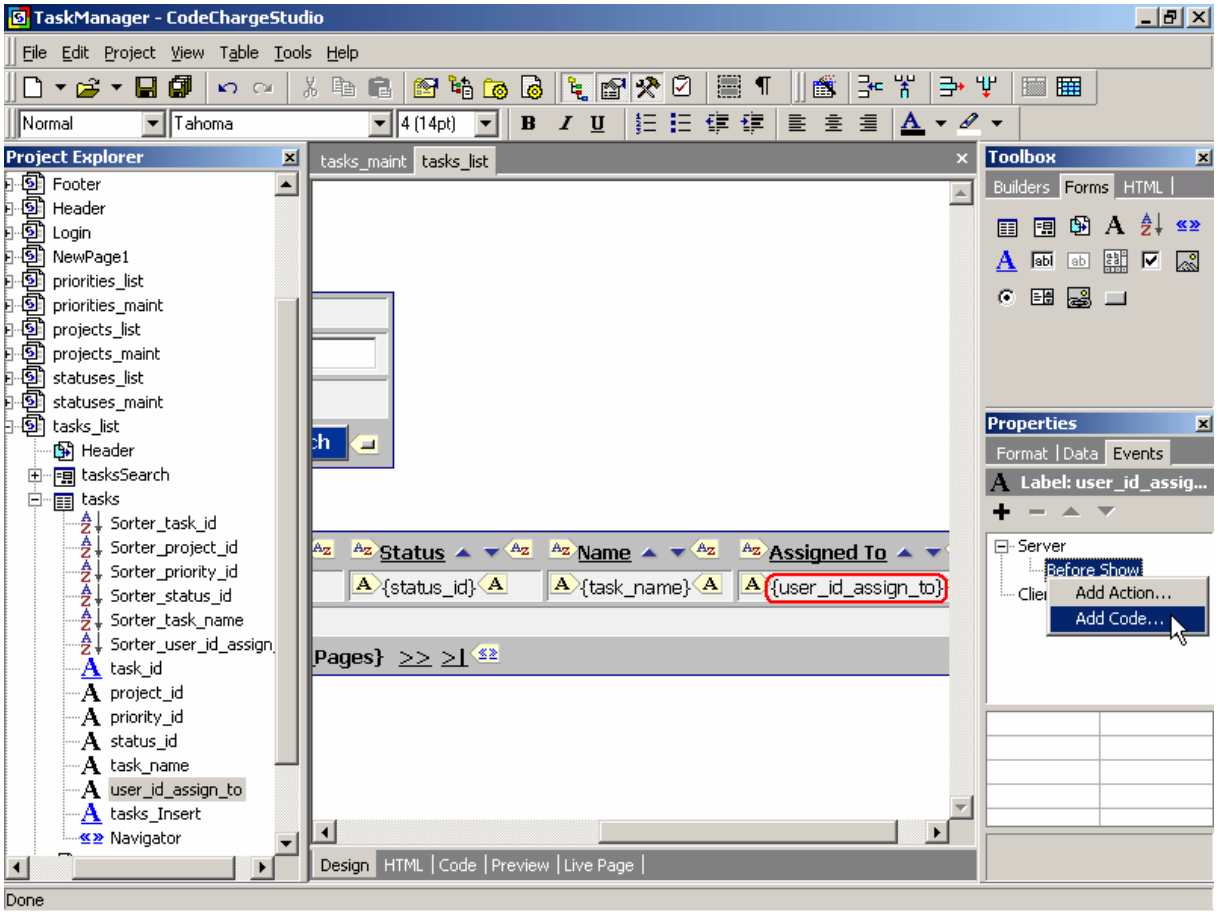
Action
User-definable code generation component, which inserts block of code into an event procedure. CodeCharge Studio comes with several predefined Actions, which are installed into the following folder:
(CCS folder)\Components\Actions
Internally, actions consist of XML and XSL code that can be easily customized.
For example an action can be set on a TextBox to validate the e-mail address.

Event Procedure
A procedure automatically executed in response to an event initiated by the program code during execution. Events are the best place for putting custom programming code.

Use "Before Show" Event to Alter Grid's Output

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.

Open the *tasks_list* page in the "Project Explorer", expand the *tasks* grid, and right-click on the *user_id_assign_to* field and select "Properties". Under the "Data" tab, set the value of the "Content" property to "HTML". Next, select the "Events" tab in the Properties window, then 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.



Programmatically Control Field's Value

After selecting the "Add Code…" option, you will see the code-editing window with the appropriate place to enter the new code.
Replace this line of code:

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

with the following lines (PHP):

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

The following is how the above PHP code works:

```
if ( $tasks->ds->f("user_id_assign_to") == CCGetSession("UserID") && strlen(CCGetSession("UserID")) )
```

This is an "if" condition that is true only if the database field *user_id_assign_to* is equal to the id 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 id to the id of the person that a task is assigned to. "UserID" is one of session variables used by CodeCharge-generated programs, and it holds the ID of the currently logged in user until the session expires. 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->user_id_assign_to->HTML = true;
```

This code sets "HTML" control property to true to indicate that the field content is in HTML markup.

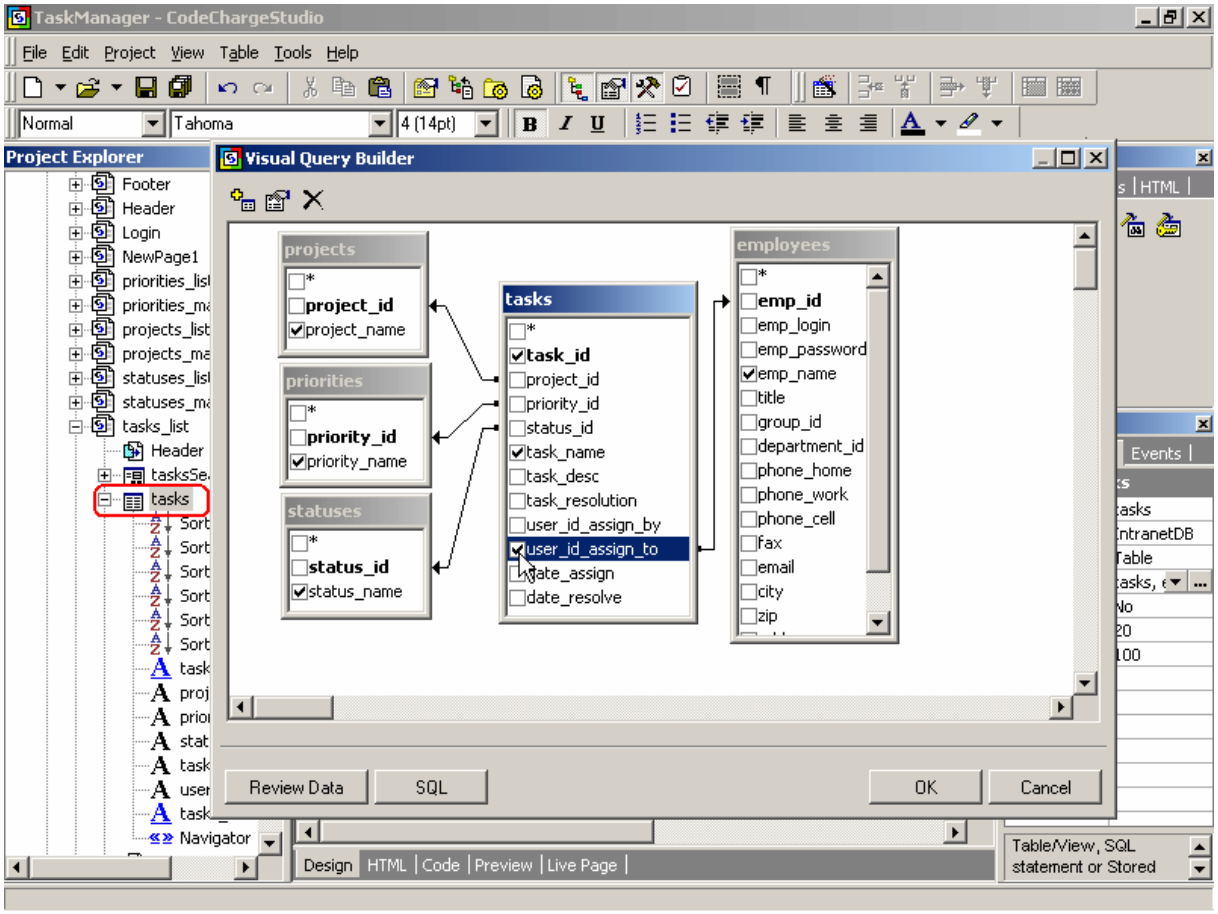
```
    $tasks->user_id_assign_to->SetValue("<b><font color=\"blue\">"
    . $tasks->user_id_assign_to->GetValue() . "</font></b>");
```

This code is executed if the previous "if" condition is met. It modifies the value of the *user_id_assign_to* field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML *<b>* 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 *user_id_assign_to* field in the *tasks* grid. *SetValue* is a method of an object, which can be used to modify the object's value.

Although the code you've inserted is complete, it may not work if the data source of your grid doesn't contain the value of the *user_id_assign_to*, which is compared against user's id. The next step is to add this database field to the grid's data source.

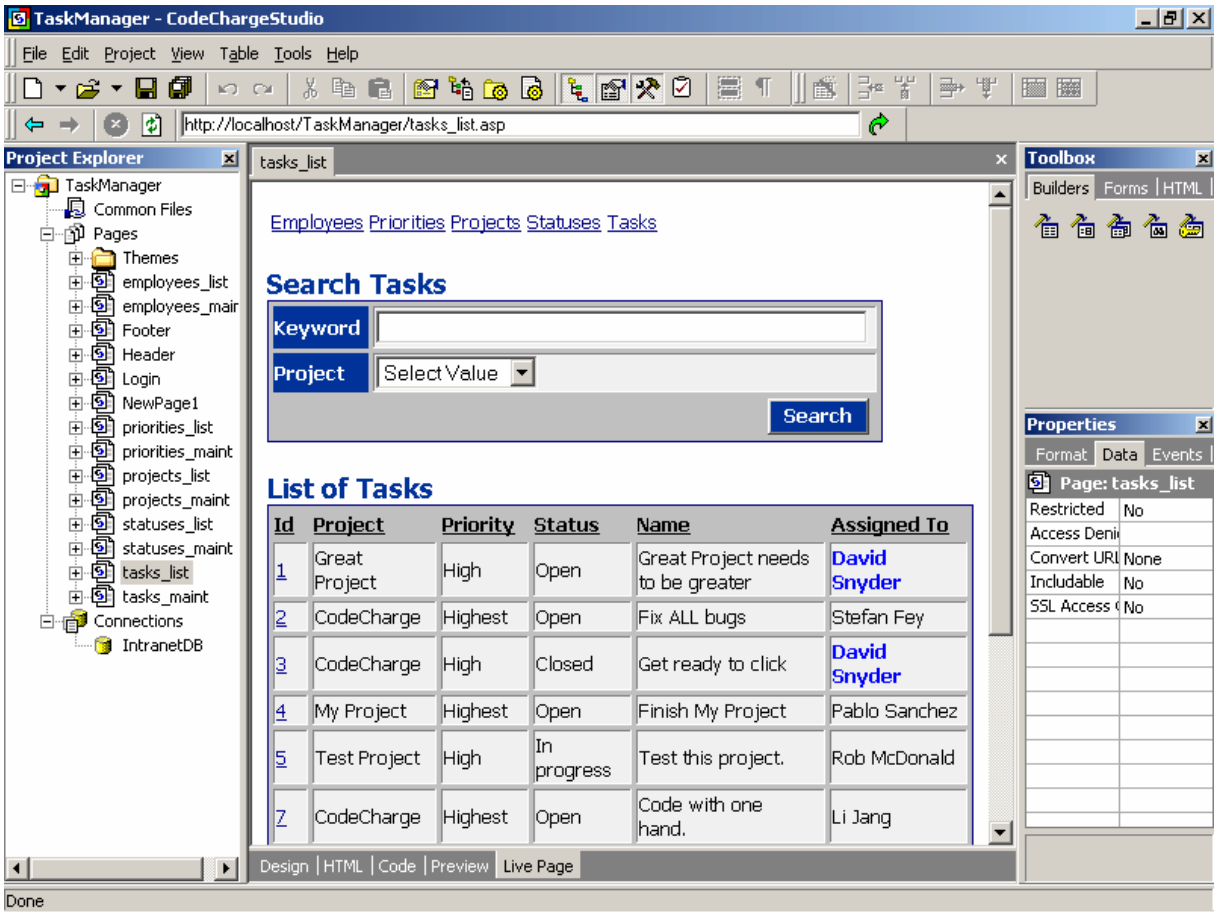
Specify Additional Database Fields for the Grid

Switch to Design mode, select the *tasks* grid in Project Explorer (or click on the grid's caption on the screen), then click on the […] button next to the grid's "Data Source" property, then click "Build Query" to open the Visual Query Builder window.
Once in Visual Query Builder, add the *user_id_assign_to* field to the data source by selecting its checkbox. Notice that the *tasks* table is connected to *employees* table because both *user_id_assign_to* and *emp_id* fields are related. You will be ale to assign a task to someone by specifying an employee and his id.



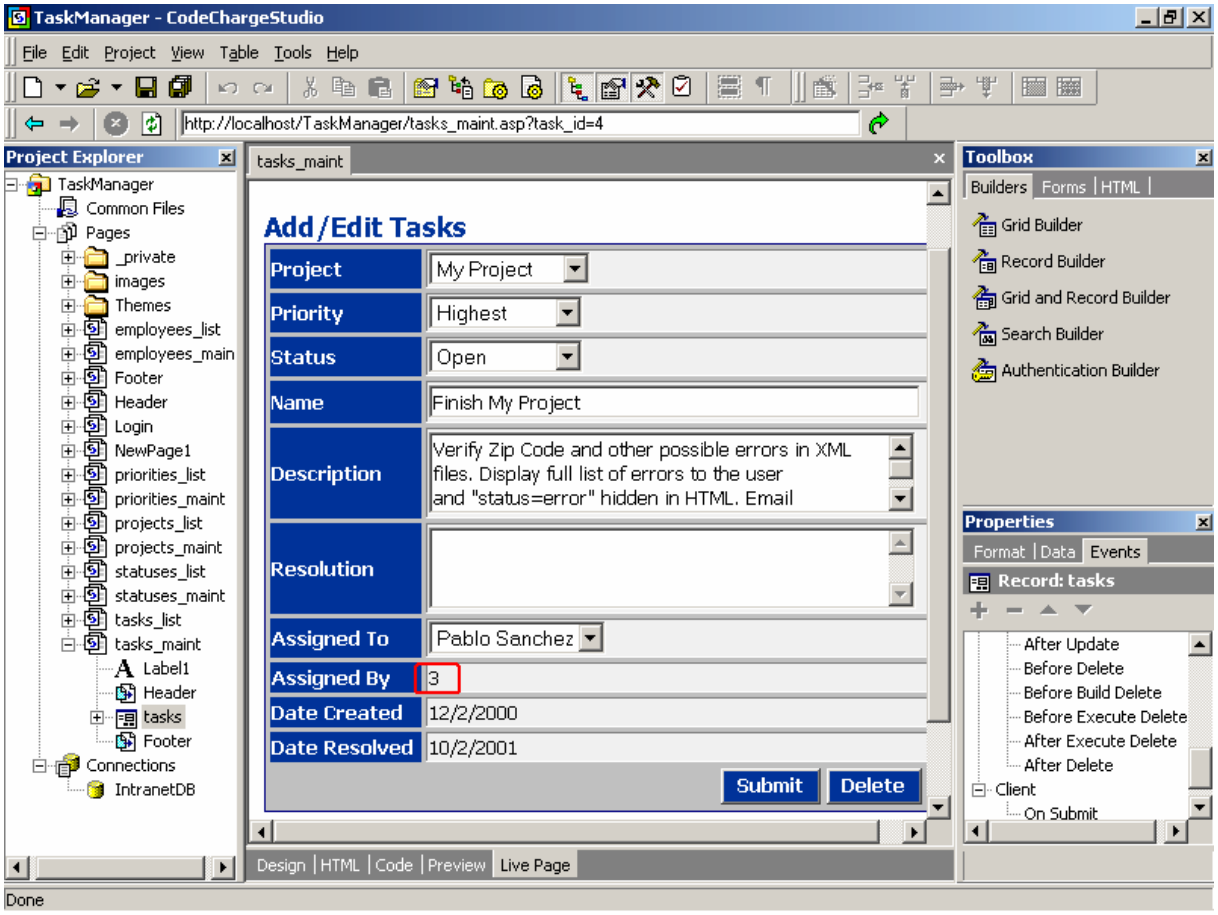
Preview Tasks List Page

Save your project and 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.
Since the menu doesn’t contain a link to the Login page at this time, you can access the Login page by trying to access one of the restricted pages, such as the Task Maintenance page. Click on any of the project Ids and you should see the login page. Login as davids / davids , then 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.



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 user's name, just like the "Assigned To" ListBox displays employee names from another table.



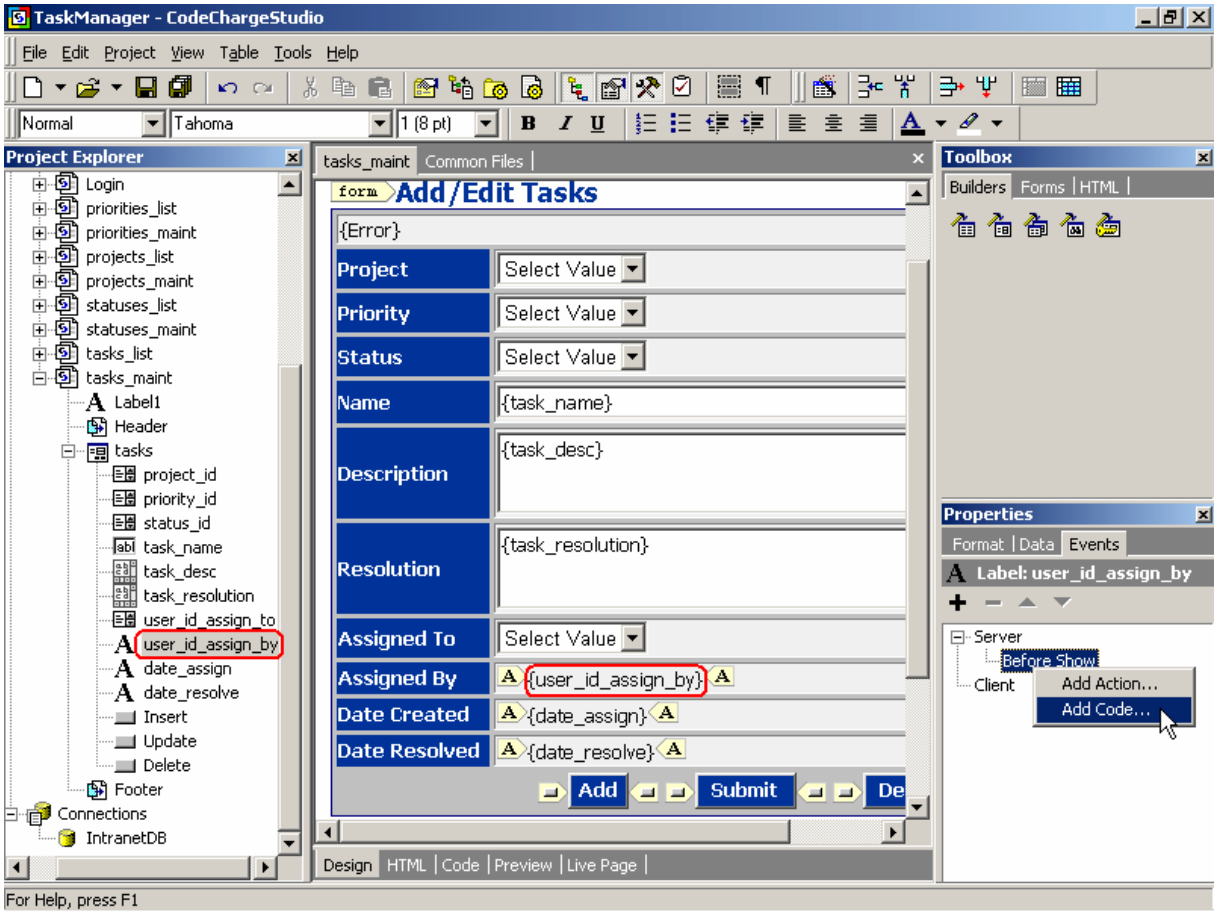
There are several potential methods of dealing with the above issue and let's explain each one in detail:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, just like you did with the grid on the Task List page.
   Unfortunately, queries that contain multiple tables cannot be updateable by their nature, and thus your whole record form would 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 the record form's properties, to specify which database fields should be updated with corresponding values entered on the page.
   This approach looks is 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.

Create a "Before Show" Event to Alter the Label's Value

Select the Label *user_id_assign_by* in the Design mode, then in the Properties window click on the "Events" tab. Right-click on the "Before Show" event and select "Add Code…".
CodeCharge Studio should automatically switch to Code view.
The "Before Show" event is a place in the program that is executed after a value is assigned to the component, but before such value is displayed.



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

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

```
//Custom Code @28-73254650
//-------------------------
// Write your own code here.
//-------------------------
//End Custom Code
```

Replace the text:

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

with the following:

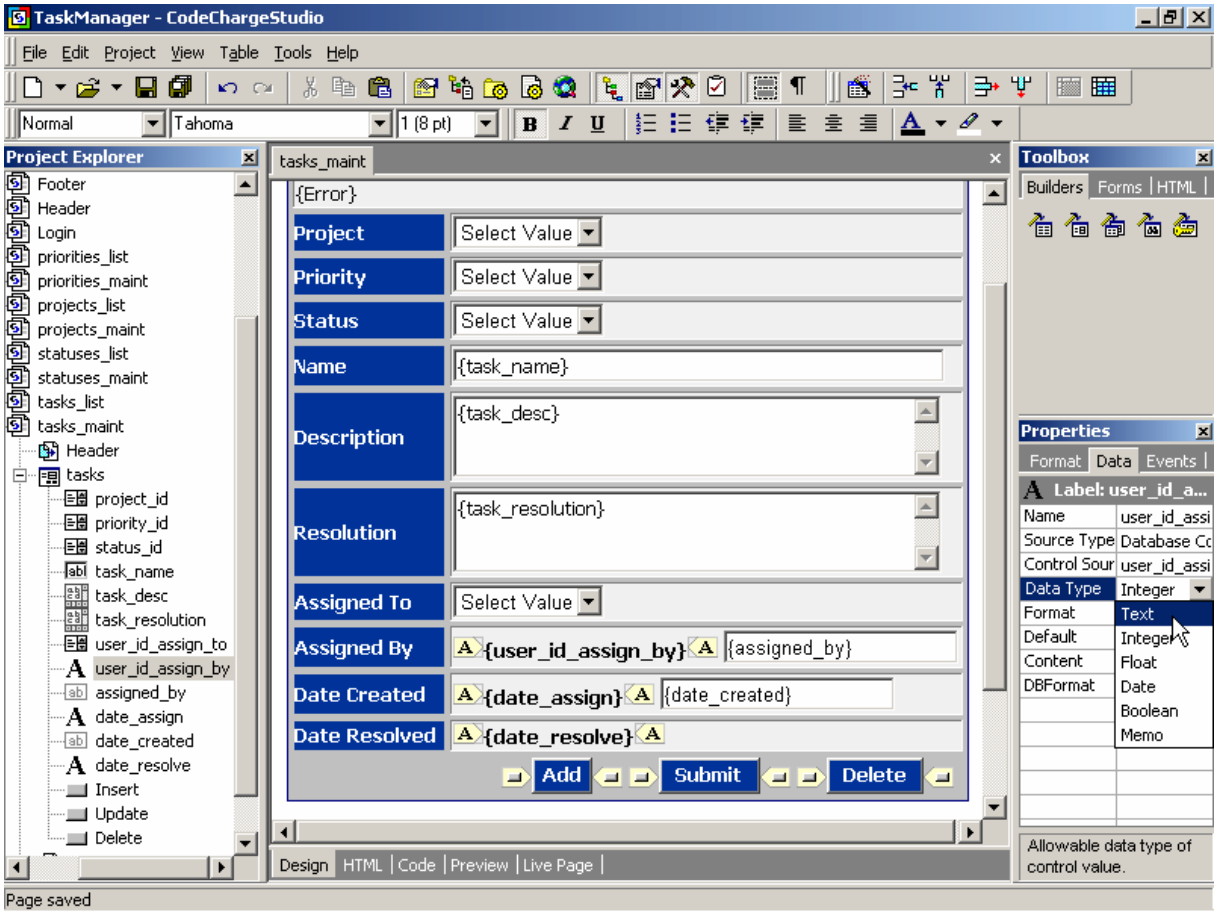
```
    global $tasks;
    global $DBIntranetDB;

    $tasks->user_id_assign_by->SetValue(CCDLookUp("emp_name", "employees", " emp_id="
    . $DBIntranetDB->ToSQL($task->assigned_by->GetValue(), 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 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 *tasks* grid.
CCDLookup –CodeCharge function that supports retrieving database value based on field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table under 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 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 *IntranetDB* connection and the value of the *assigned_by* Hidden Field.
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 CCDLookup function that uses *IntranetDB* connection and the value of the *assigned_by* Hidden Field that use as default value *CCGetUserID()* function that obtains current user’s ID.
”

Refer to the CodeCharge Studio Programming Reference for more information about functions and properties available in CodeCharge-generated programs.

Now that you’ve programmatically modified the value of the *user_id_assign_by* Label to output Employee Name instead of the ID, you will also need to specify that this field is now a Text field, instead of Numeric.
Click on “Data” tab in Properties window, select “Text” as the Data Type,
select ‘Code Expression’ as Source Type and leave Control Source property empty.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Normal Tahoma 1 (8 pt)
Project Explorer
Footer
Header
Login
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolution
user_id_assign_to
user_id_assign_by
assigned_by
date_assign
date_created
date_resolve
Insert
Update
Delete
tasks_maint
{Error}
Project
Select Value
Priority
Select Value
Status
Select Value
Name
{task_name}
Description
{task_desc}
Resolution
{task_resolution}
Assigned To
Select Value
Assigned By
{user_id_assign_by}
{assigned_by}
Date Created
{date_assign}
{date_created}
Date Resolved
{date_resolve}
Add
Submit
Delete
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Label: user_id_a...
Name user_id_assi
Source Type Database Co
Control Sour user_id_assi
Data Type Integer
Format Text
Default Integer
Content Float
DBFormat Date
Boolean
Memo
Allowable data type of control value.
Page saved

Create "Before Show" Event to Alter Label's Value for Label date_assign

Select the Label *date_assign* in the Design mode, then in the Properties window click on "Events" tab. Right-click on "Before Show" event and select "Add Code…".
CodeCharge Studio should automatically switch to Code view at that time.
"Before Show" event is a place in the program that is executed after a value is assigned to the component, but before such value is displayed.

Use "Before Show" Event to Alter Label's Value

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

```
//Custom Code @28-73254650
//-------------------------
// Write your own code here.
//-------------------------
//End Custom Code
```

Replace the text:

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

with the following:

```
    global $tasks;

    $tasks->date_assign->SetValue($tasks->date_created->GetValue());
```

The whole code reads approximately as follows:
"If a record is being edited:
Assign the date to the *date_assign* Label, by set the value from *date_created* Hidden Field.
If new record is being created it assign date to the *date_assign* Label by retrieving the value from the *date_created* Hidden Field that use as default value *time()* function that obtains current Date.
"

You will also need to specify that this field is now a Date field,
Click on "Data" tab in Properties window, select "Data" as the Data Type,
select 'Code Expression' as Source Type and leave Control Source property empty.

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

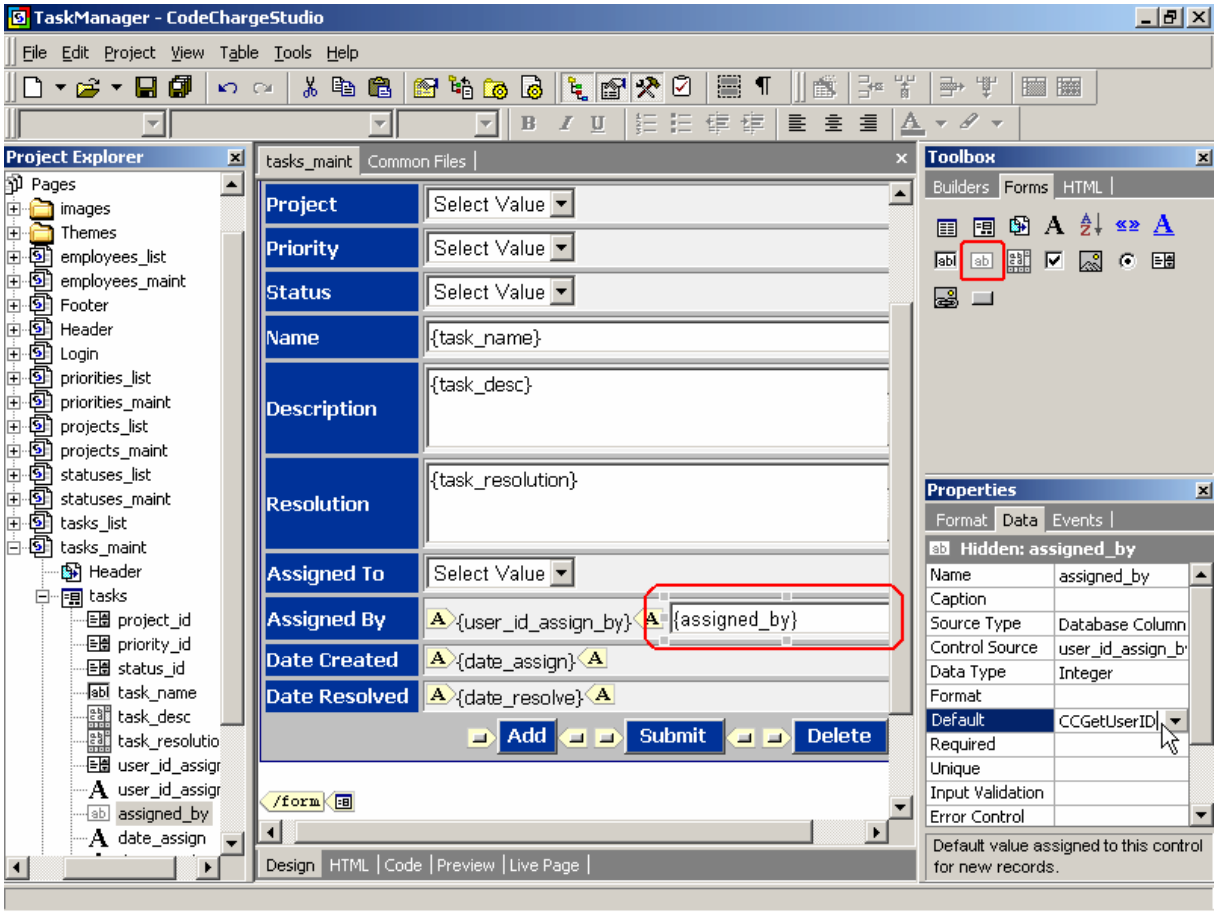
You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.
First, add a Hidden field to your page from the "Forms" tab on the "Toolbox" window. This field type isn't visible in the browser, but will be used to store new values and update the database.
Then 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.

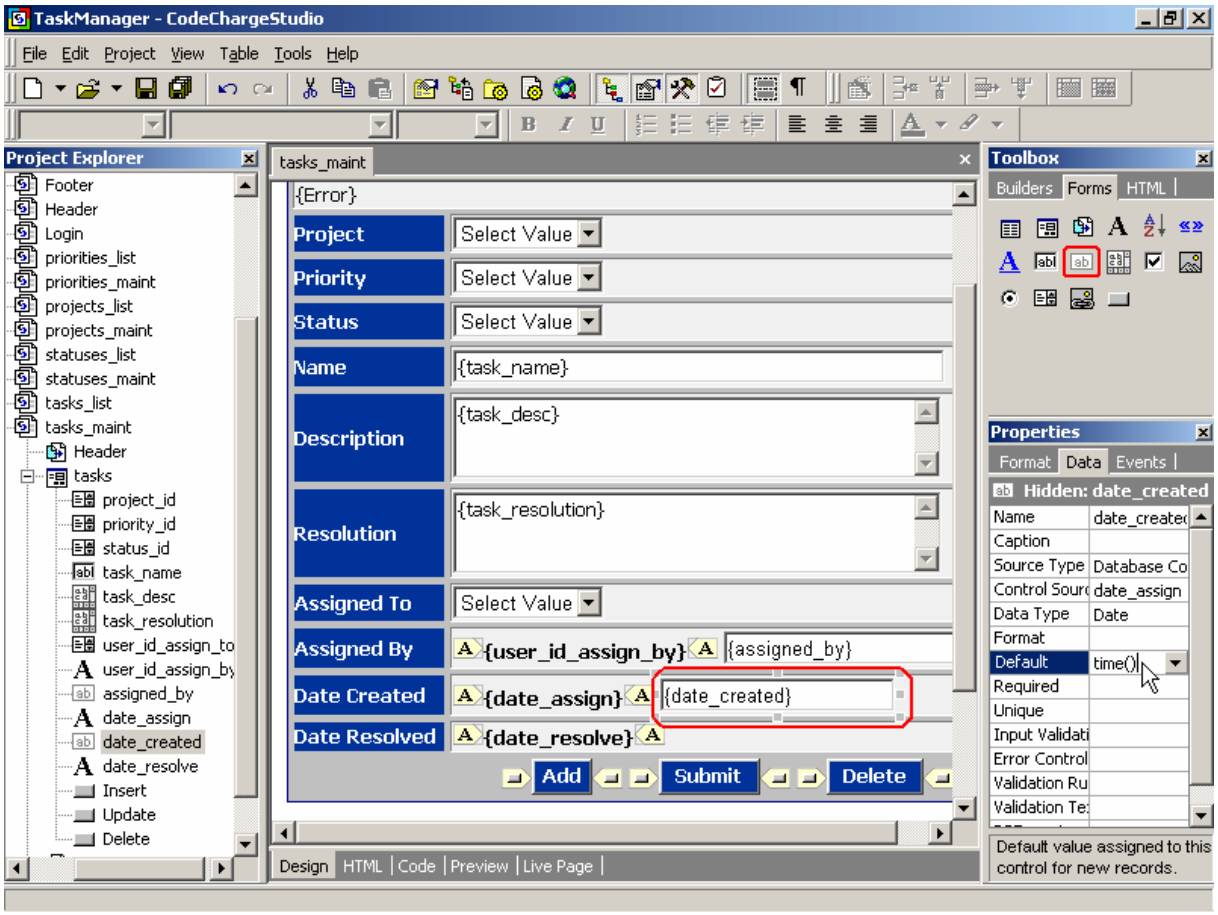
Default: *CCGetUserID()* – default value for this field if empty. CCGetUserID() is a CodeCharge function that retrieves the ID of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
Pages
images
Themes
employees_list
employees_maint
Footer
Header
Login
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolutio
user_id_assigr
user_id_assigr
assigned_by
date_assign
tasks_maint
Common Files
Project
Select Value
Priority
Select Value
Status
Select Value
Name
{task_name}
Description
{task_desc}
Resolution
{task_resolution}
Assigned To
Select Value
Assigned By
{user_id_assign_by}
{assigned_by}
Date Created
{date_assign}
Date Resolved
{date_resolve}
Add
Submit
Delete
/form
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Hidden: assigned_by
Name assigned_by
Caption
Source Type Database Column
Control Source user_id_assign_b
Data Type Integer
Format
Default CCGetUserID
Required
Unique
Input Validation
Error Control
Default value assigned to this control for new records.

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.
Configure the new field as follows:

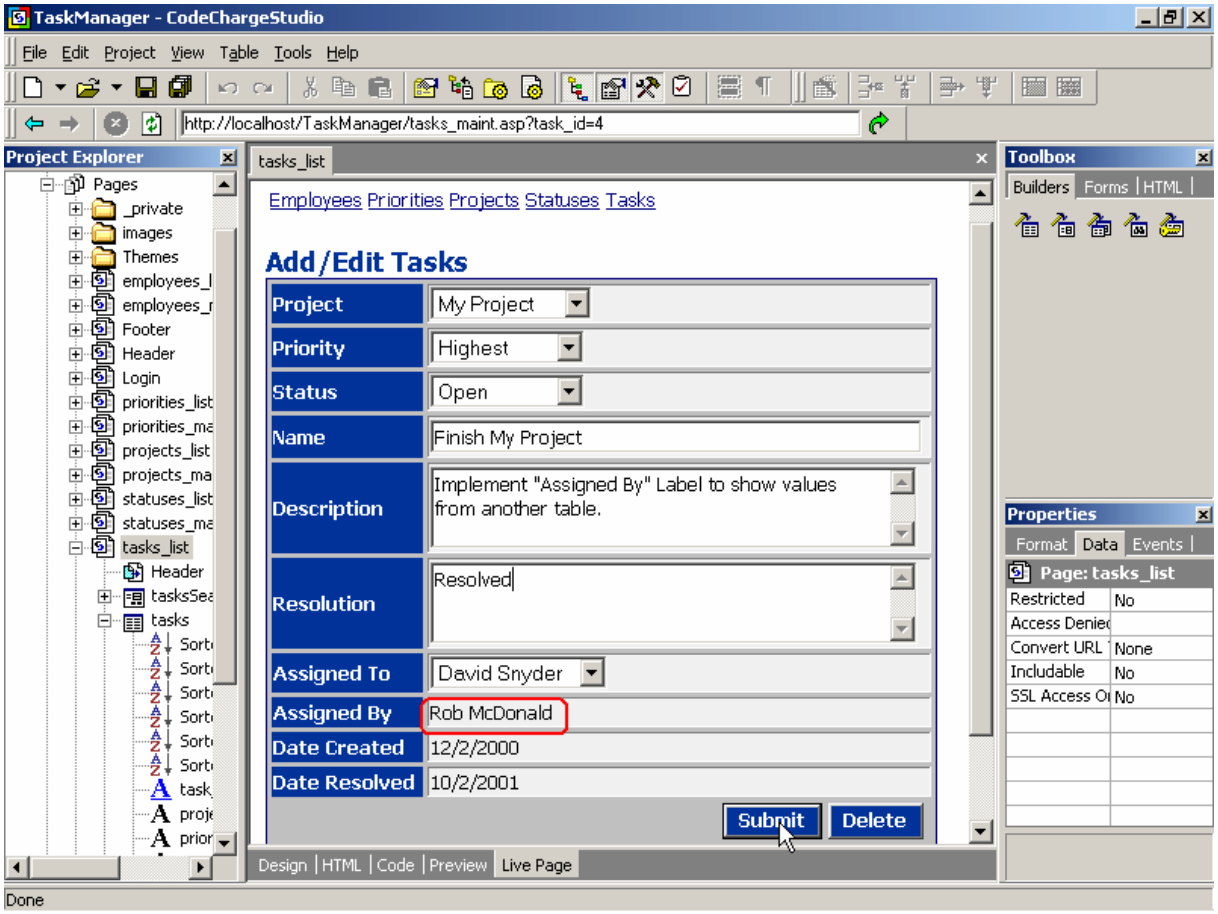
Name: *date_created*
Control Source: *date_assign*
Data Type: *Date*
Default: *time()* – time() is a PHP function that obtains the server's current date and time. Using this function allows you to automatically assign the current date and time to new tasks. The Default property doesn't affect existing records, thus the date of existing tasks won't be modified during updates.



Test the Label and Hidden Fields

Finally, you can switch to Live Page mode, select a Task, Login, and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.
Don't forget to save it!



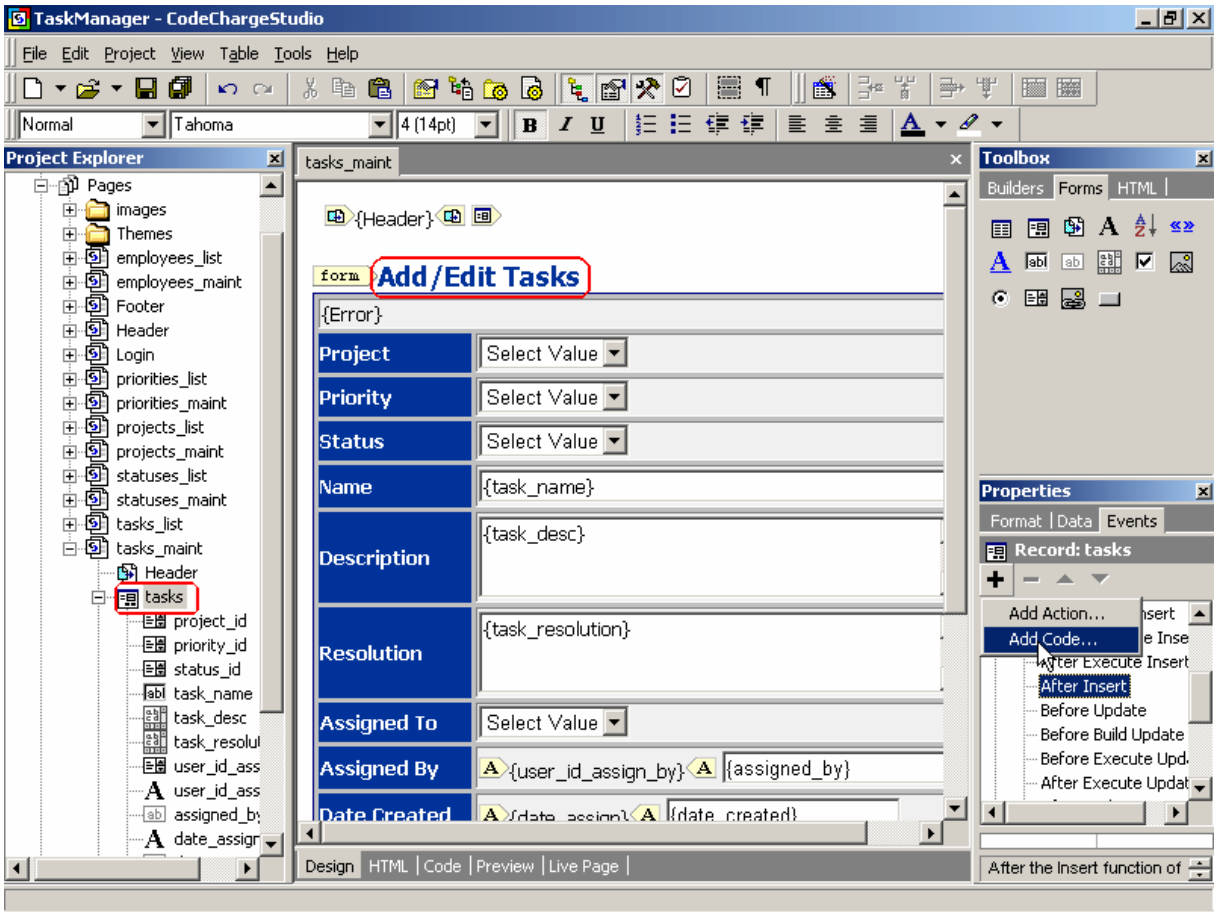
Programming the Record Form

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

Add Code in the “After Insert” Event to Send Emails

Select the “tasks” form by selecting it in the Project Explorer, or clicking anywhere within the form’s caption. Then in the Properties window click on the “Events” tab and select the “After Insert” event.
Click on the [+] button, then select “Add Code…”



Once your in the Code view, 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_name    = CCDLookUp("emp_name", "employees", "emp_id="
   . $DBIntranetDB->ToSQL($tasks->user_id_assign_to->GetValue(), ccsInteger) , $DBIntranetDB);
$to_email   = CCDLookUp("email", "employees", "emp_id="
   . $DBIntranetDB->ToSQL($tasks->user_id_assign_to->GetValue(), ccsInteger) , $DBIntranetDB);

$recipient = $to_name  . "<" . $to_email . ">";

$headers  = "From: " . $from_name . "<" . $from_email . ">\n";
$headers .= "Content-Type: text/html\n";
$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 ($recipient, $subject, $message, $headers);
```

As you may have realized by now, the above code sends emails to users to whom the new tasks are assigned.

```
$from_email = 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.

```
$from_name  = 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.

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

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

```
mail ($recipient, $subject, $message, $headers);
```
Sends the email.

Use the "After Update" Event to Send Emails

You've previously added the necessary code that sends email notification to the assignee upon recording a new task in the system. Now implement similar functionality in "After Update" Event to notify assignee when an existing task is updated and reassigned to someone.
Click on the "tasks" form in the Project Explorer, then in the Properties window, select the "Events" tab, and then add the following "Custom Code" in "After Update" event:

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

  $recipient = $to_name  . "<" . $to_email . ">";

  $headers  = "From: " . $from_name . "<" . $from_email . ">\n";
  $headers .= "Content-Type: text/html\n";
  $subject  = "A task was assigned to you";
  $message  = "The following task was assigned to you:<br><br>"
    . "Task ID: " . CCGetFromGet ("task_id")
    . "<br><br>" . $tasks->task_desc->GetText();

    mail ($recipient, $subject, $message, $headers);
}
```

The main differences between the above code and the one you used in "After Insert" event are as follows:

- An "if" condition was added to send an email only if a user assigns a task to someone other than himself/herself
- *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***

Test Email Delivery

Before testing the system, you should add new users to your database with correct email addresses, or modify the existing test users by changing their email address. You can do this by opening the Intranet.mdb database that is located in your project directory. Alternatively, you may use the Task Manager itself and go to the Employees page to view and modify user emails there.
(Note: You will need Ms Access 2000 to manually edit the database file.)

Once you have users configured with test emails, save your project and switch to Live Page mode to test your system. If your email properties in php.ini was properly installed, 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.

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 explain several of them.

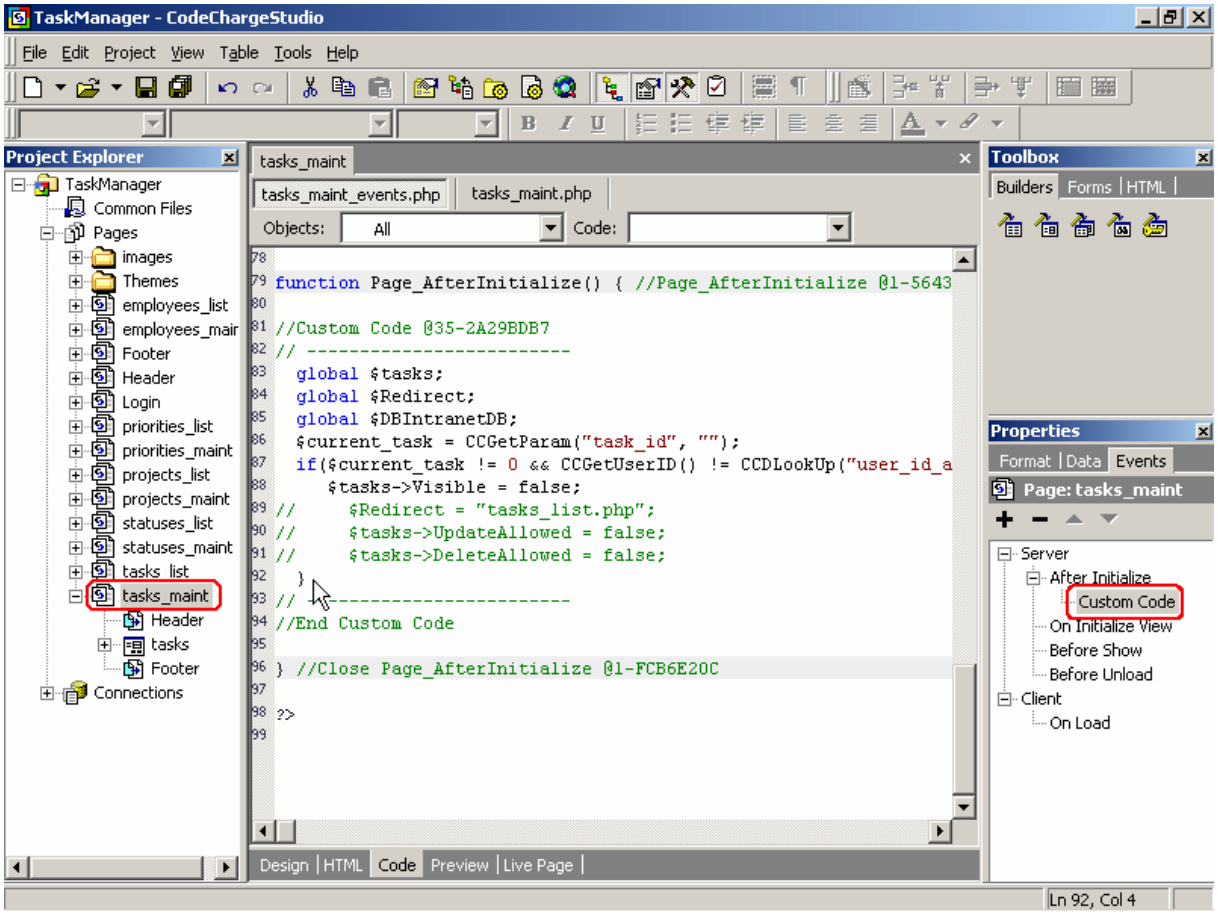
First, click on the "tasks_maint" page in the Project Explorer, then select "Events" tab in the Properties window. Add "Custom Code" to the "After Initialize" event of the page.

Once in the Code view, replace the generated comment:

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

with the code below:

```
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:

1. **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.
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 opposite, mutually exclusive criteria.

2. **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;
$current_task = CCGetParam("task_id", "");
if($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;
}
```

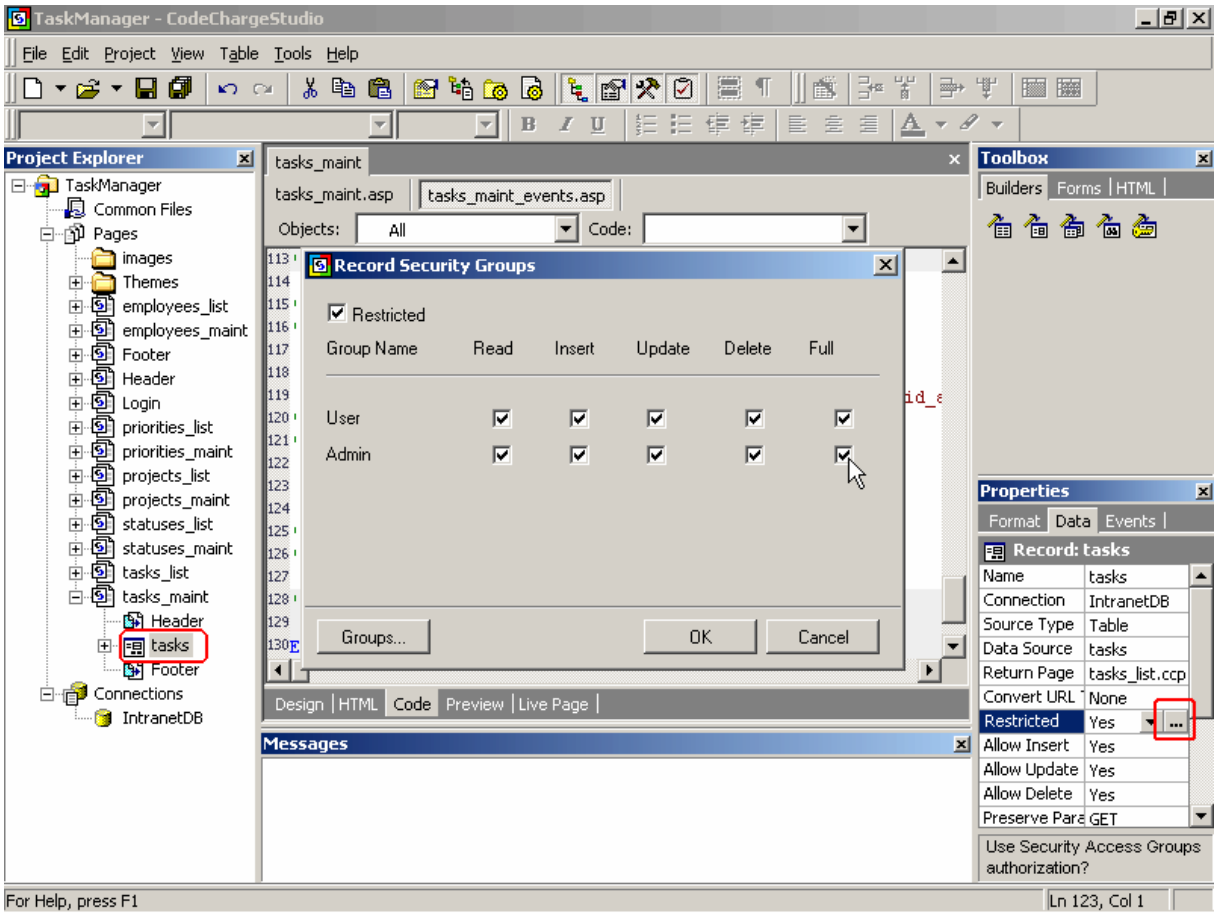
The above code shows that you should comment out the previously active line, and uncomment the line that starts with “Redirect”.
$*Redirect* is a variable used by CodeCharge Studio to determine if the current page should be redirected to another page, for example if a user is not logged in. This variable 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.

3. **Disable Update/Submit and Delete buttons for unauthorized users.**
Comment out the “Redirect” statement and uncomment the two lines of code below it, as shown here:

```
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 the appearance of the “Update” and “Delete” buttons on the page. If set to False, the buttons will not appear. Additional security is implemented to make impossible to update the record even if someone saves the page, adds the missing buttons and submits the page externally. However, the above code won’t work unless you also set the “Restricted” property of your form to “Yes” and assign permissions to everyone. That’s because CodeCharge Studio generates the code appropriate for hiding and disabling buttons

only when there is a need to do so, and restricting page access indicates that certain users are not allowed to add, update or delete records.



4. One more, although less secure method of disabling buttons on the record form is to hide the Update and Delete buttons on the page by adding the following custom code to the “Before Show” event of the form:

```
global $tasks;
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->Update->Visible = false;
   $tasks->Delete->Visible = false;
}
```

Enhancing Application Functionality with Programming Events (ColdFusion)

You've probably noticed that until now you've built your Task Management application without having to deal with the programming code. CodeCharge Studio can help you build fairly functional systems without any programming, however creating more sophisticated systems will require a certain amount of programming code. Fortunately, CodeCharge Studio makes programming easy by providing you with a first-rate code editor, in addition to Events and Actions that aid you in inserting pre-programmed snippets of code into the right place within the program.

Here are the definitions of Action and Event:

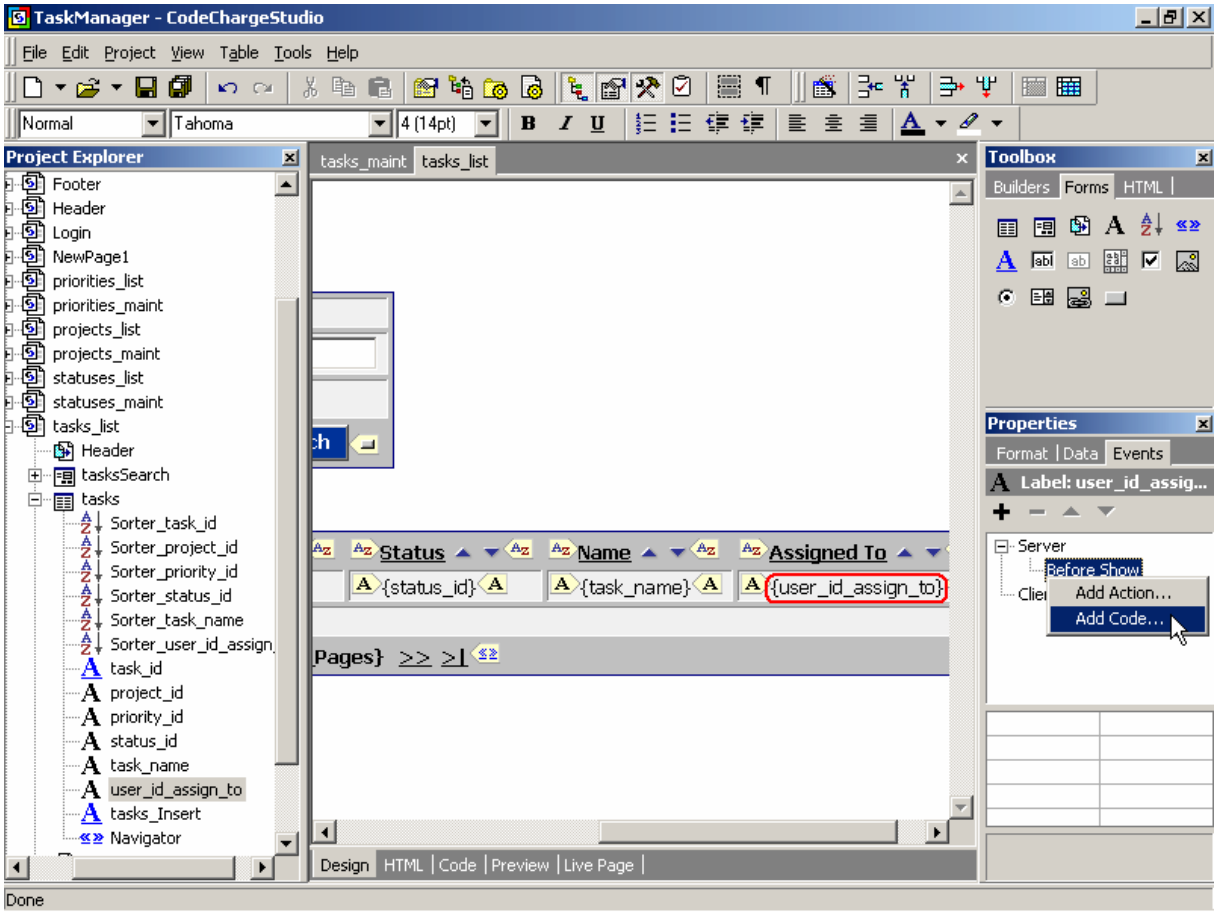
Action
User-definable code generation component, which inserts block of code into an event procedure. CodeCharge Studio comes with several predefined Actions, which are installed into the following folder:
(CCS folder)\Components\Actions
Internally, actions consist of XML and XSL code that can be easily customized.
For example an action can be set on a TextBox to validate the e-mail address.

Events
Events are the best place for putting custom programming code.

Use "Before Show" Event to Alter Grid's Output

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.

Open the *tasks_list* page in the "Project Explorer", expand the *tasks* grid, and right-click on the *user_id_assign_to* field and select "Properties". Under the "Data" tab, set the value of the "Content" property to "HTML". Next, select the "Events" tab in the Properties window, then 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.



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.
Replace this line of code:

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

with the following lines (ColdFusion):

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

Let's explain how the above ColdFusion code works:

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

We are test variable "Session.UserID ".If the variable does not exist, it is created and set to the value of the DEFAULT attribute

```
<CFIF user_id_assign_to EQ Session.UserID AND Session.UserID NEQ "">
```

This is an "if" condition that is true only if the database field *user_id_assign_to* is equal to the id 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 id to the id of the person that a task is assigned to. "UserID" is one of session variables used by CodeCharge-generated programs, and it holds the ID of the currently logged in user until the session expires. 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 flduser_id_assign_to= '<b><font color="blue">'  & flduser_id_assign_to & '</font></b>'>
```

This code is executed if the previous "if" condition is met. It modifies the value of the *user_id_assign_to* field. The field value is replaced with its database value wrapped within HTML code that specifies the font color as blue, and adds HTML <*b*> 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 *user_id_assign_to* field in the *tasks* grid.

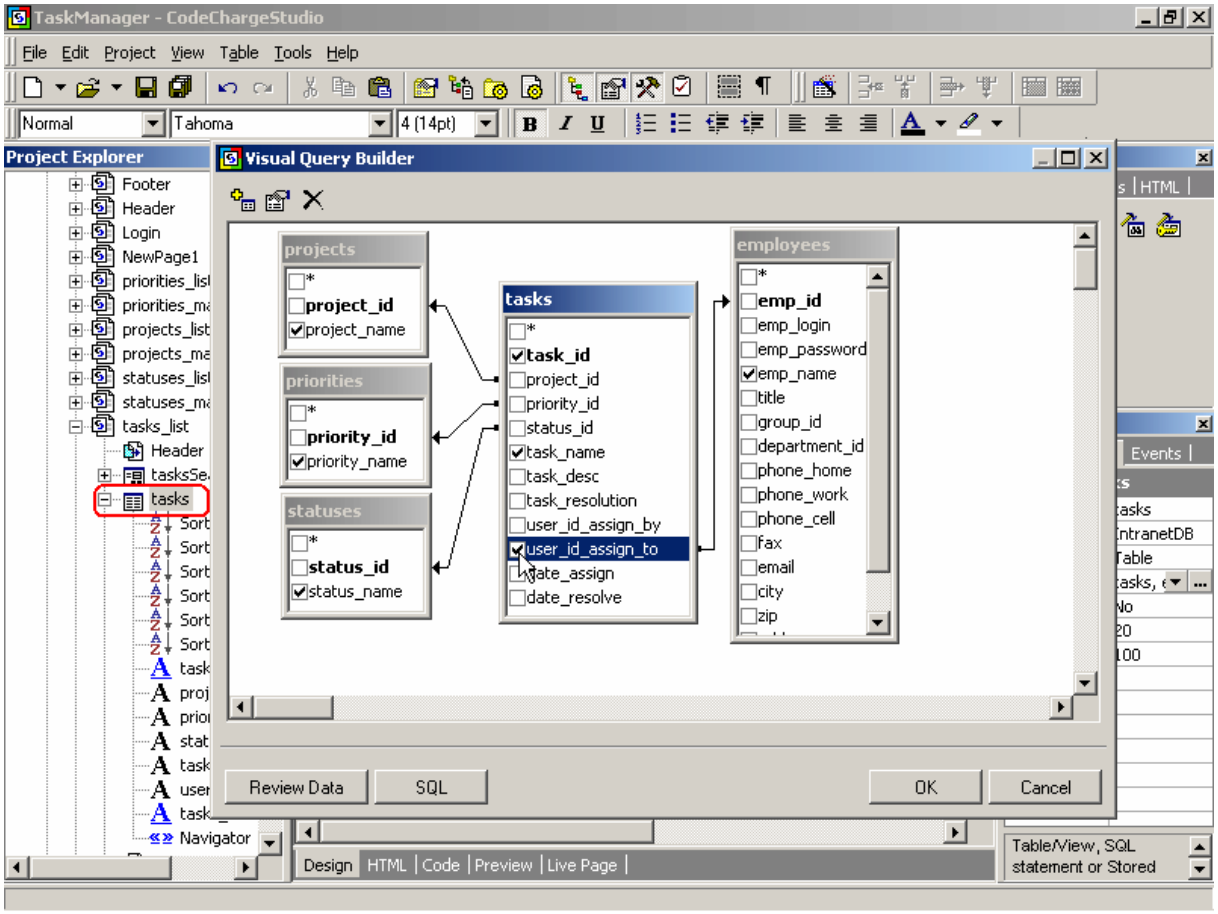
```
</CFIF>
```

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

Although the code you've inserted is complete, it may not work if the data source of your grid doesn't contain the value of the *user_id_assign_to*, which is compared against user's id. The next step is to add this database field to the grid's data source.

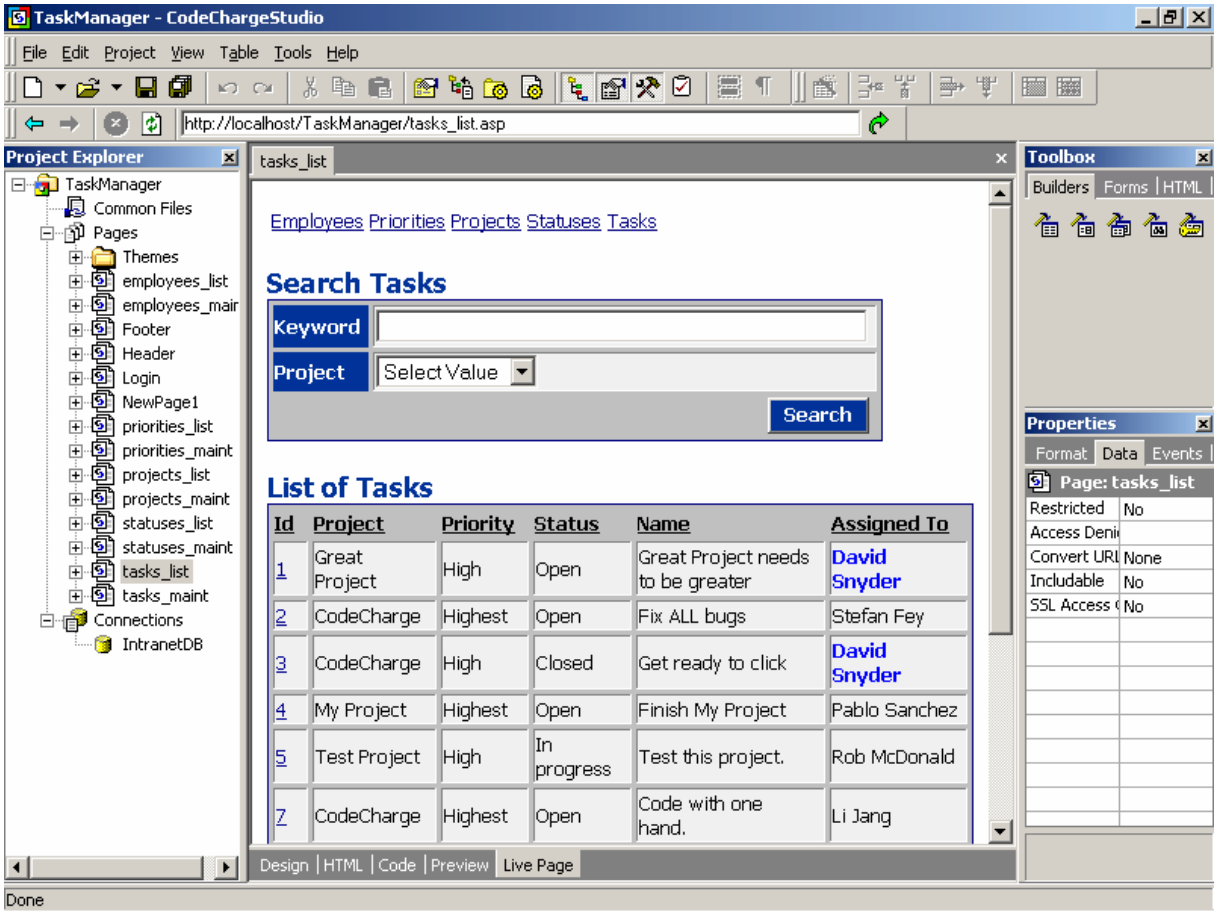
Specify Additional Database Fields for the Grid

Switch to Design mode, select the *tasks* grid in Project Explorer (or click on grid's caption on the screen), then click on the […] button next to grid's "Data Source" property, then click "Build Query" to open the Visual Query Builder window.
Once in Visual Query Builder, add the *user_id_assign_to* field to the data source by selecting its checkbox. Notice that the *tasks* table is connected to *employees* table because both *user_id_assign_to* and *emp_id* fields are related. You will be ale to assign a task to someone by specifying an employee and his id.



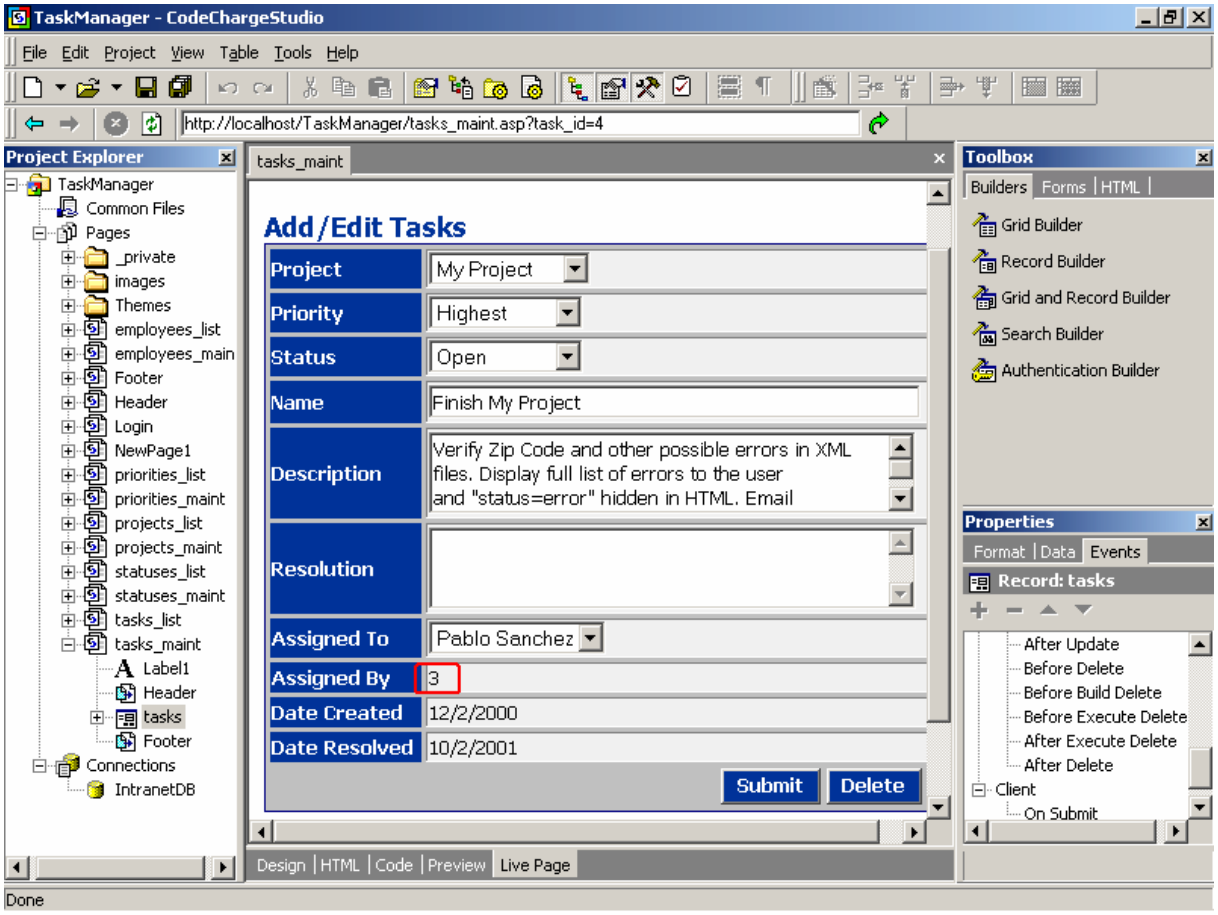
Preview Tasks List Page

Save your project and 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.
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. Click on any of the project Ids and you should see the login page.
Login as davids / davids , then 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.



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 *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.



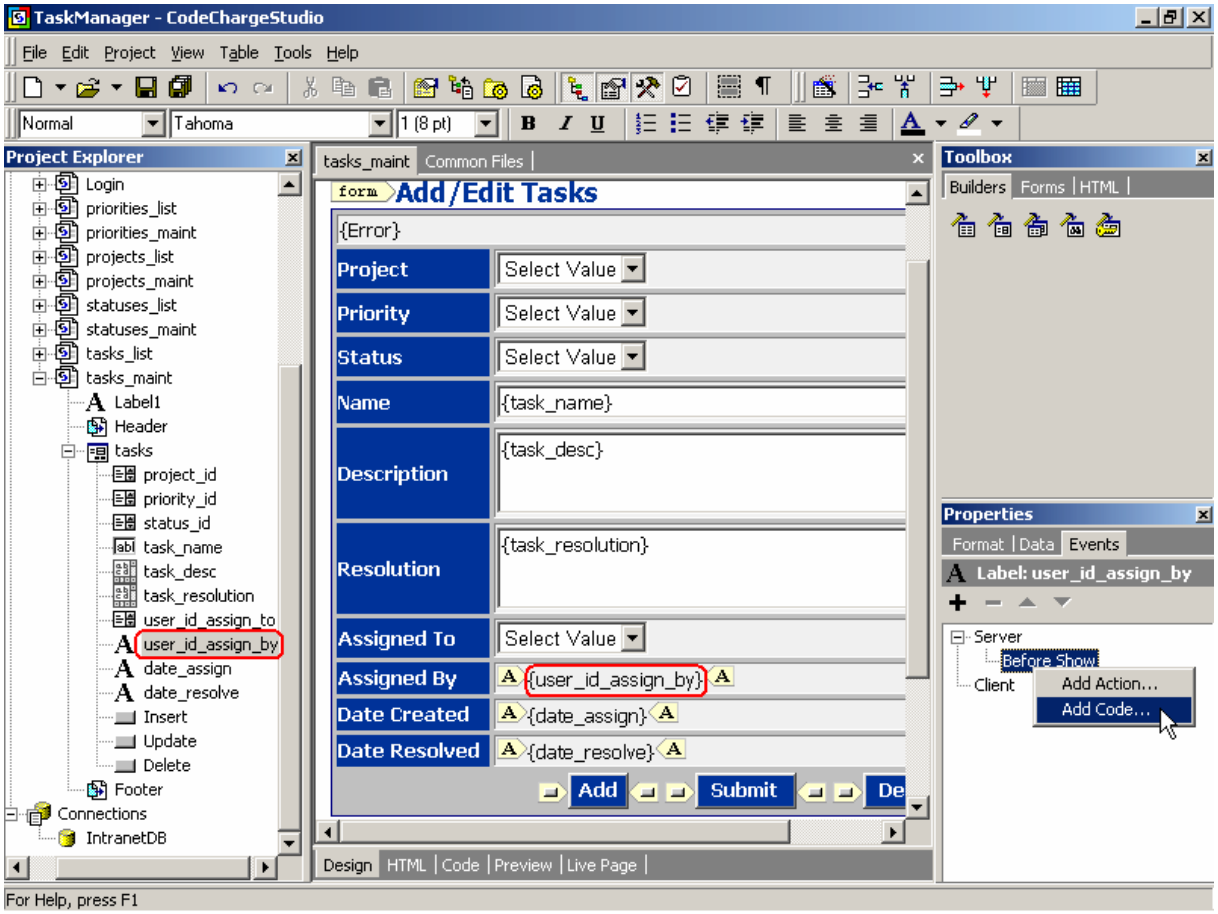
There are several potential methods of dealing with the above issue and let's explain each one in detail:

1. Create a Query that contains multiple tables and can be used as the data source for the record form, 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.

Create “Before Show” Event to Alter Label’s Value

Select the Label *user_id_assign_by* in the Design mode, then in the Properties window click on “Events” tab.
Right-click on “Before Show” event and select “Add Code…”.
CodeCharge Studio should automatically switch to Code view at that time.
“Before Show” event is a place in the program that is executed after a value is assigned to the component, but before such value is displayed.



Use "Before Show" Event to Alter Label's Value

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

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

Replace the text:

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

with the following:

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

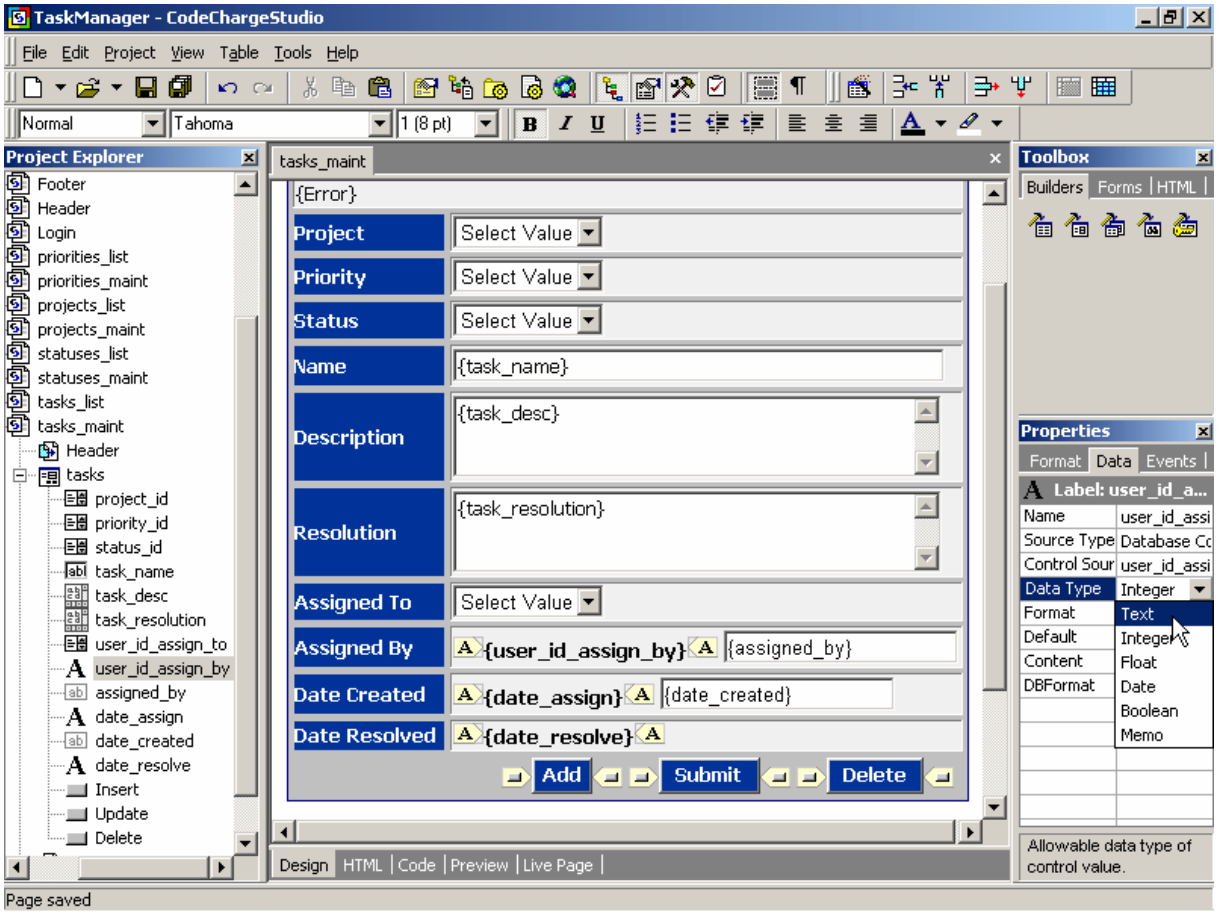
The above code consists of the following elements:
tasks –the name of the record form on the page
blnEditModedTasks – form variable, which specifies if the record is being edited. Depending on the value of this property, the program displays either the name of the person who originally submitted the 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 use by the generated program to refer to the Label and database field for be altered. CodeCharge Studio constructs variable names by adding the "fld" prefix to field's name.
CCDLookup –CodeCharge custom tag that supports retrieving database value based on field name, table name, and a condition. Here, this function retrieves the Employee Name (*emp_name*) from the *employees* table under 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 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 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 *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 CCDLookup function that uses *IntranetDB* connection and Session.UserID variable that obtains current user's ID."

Refer to the CodeCharge Studio Programming Reference for more information about functions and properties available in CodeCharge-generated programs.

Now that you programmatically modified the value of the *user_id_assign_by* Label to output Employee Name instead of the ID, you will also need to specify that this field is now a Text field, instead of Numeric.
Click on “Data” tab in Properties window, and select “Text” as the Data Type.



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

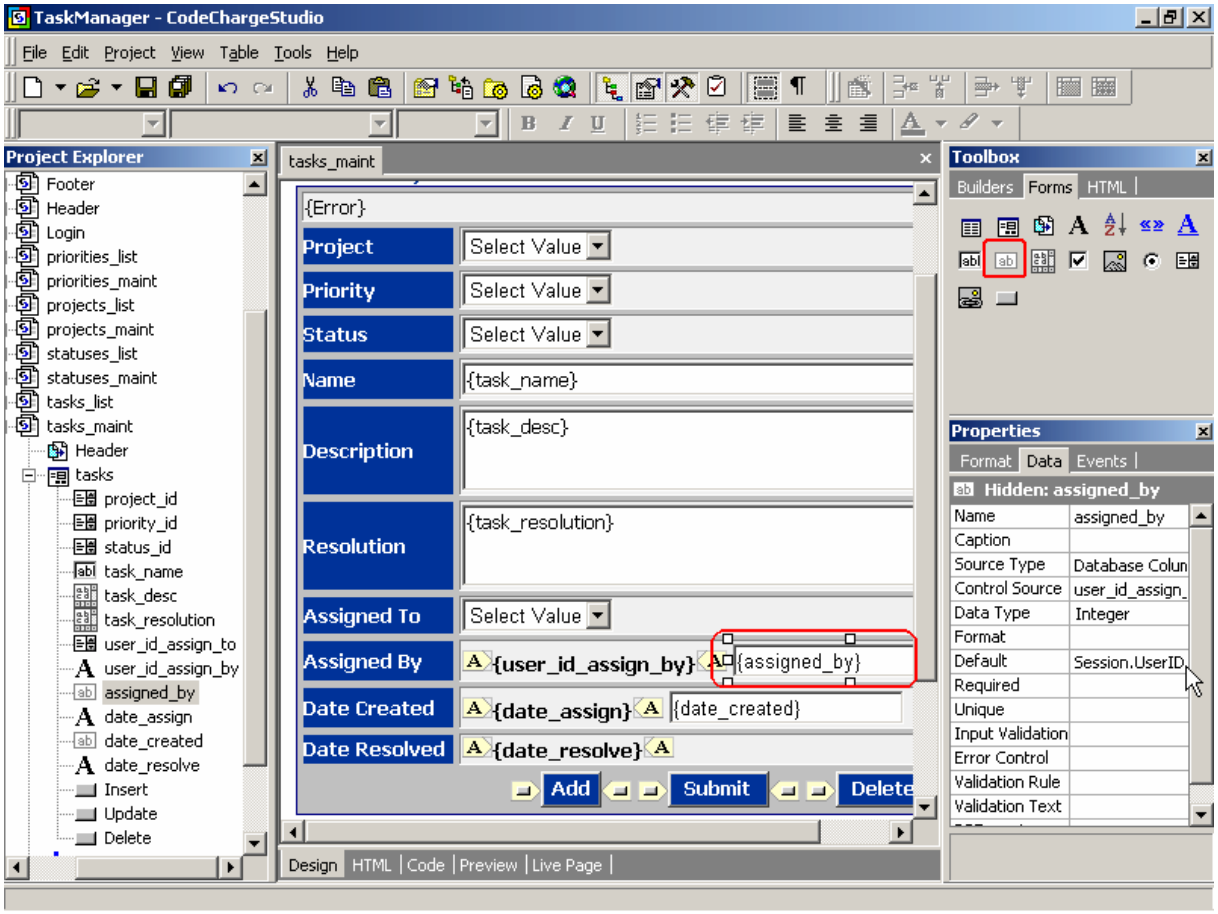
You've previously used the Before Show event to display the name of the person who assigns the task. However, Label fields are not updateable by nature, therefore even though employee's name is displayed on the page, it is not written to the database. Since we want the database to record the name or id of the person who submits a task, we will need to add programming logic to accomplish this.
First, add a Hidden field to your page from the "Forms" tab on the "Toolbox" window. This field type isn't visible in the browser, but will be used to store values and update the database.
Then 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.

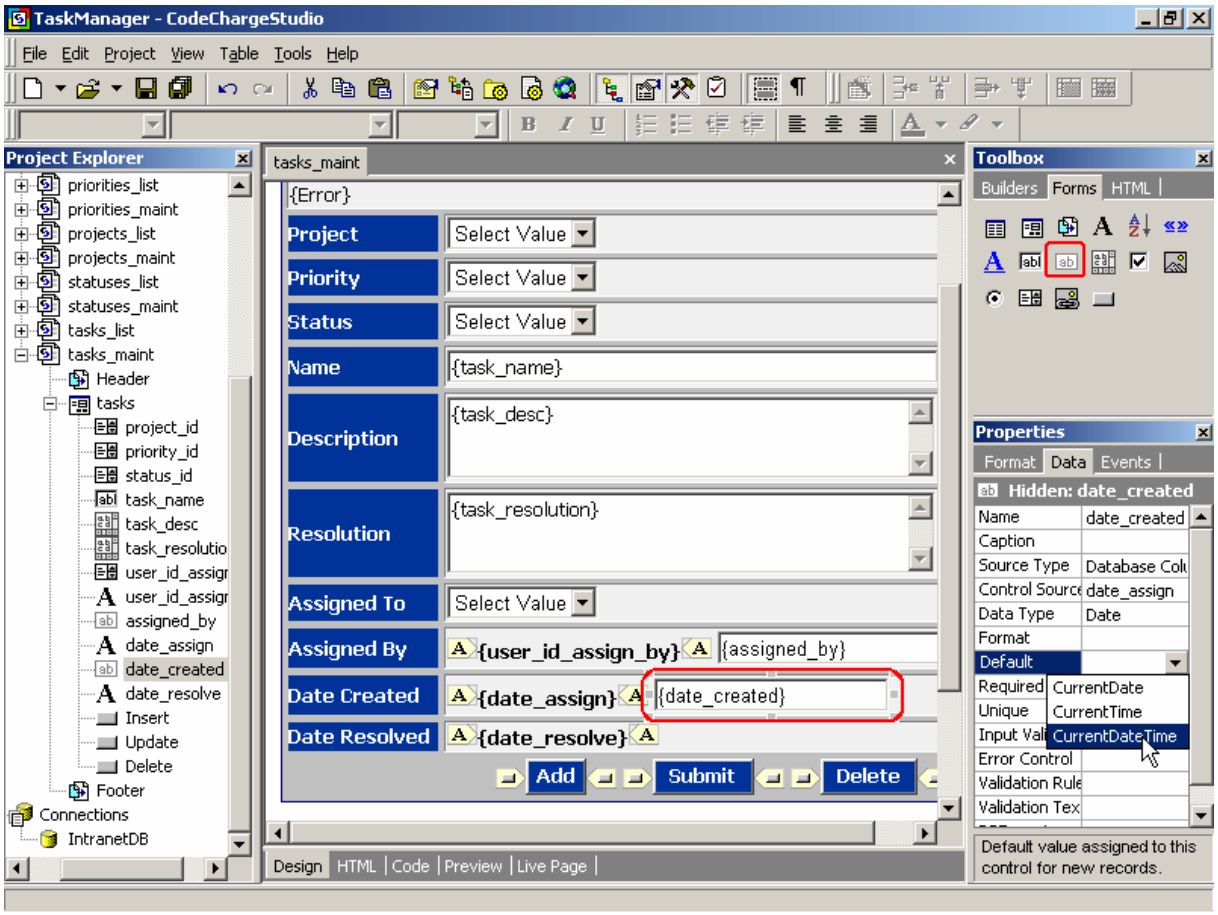
Default: *Session.UserID* – default value for this field if empty. Session.UserID is a function that retrieves the ID of the user that is currently logged in into the system. This way you can simply specify that you want to record the current user's id in the *user_id_assign_by* field for each new task that is being submitted.


TaskManager - CodeChargeStudio
File Edit Project View Table Tools Help
Project Explorer
Footer
Header
Login
priorities_list
priorities_maint
projects_list
projects_maint
statuses_list
statuses_maint
tasks_list
tasks_maint
Header
tasks
project_id
priority_id
status_id
task_name
task_desc
task_resolution
user_id_assign_to
user_id_assign_by
assigned_by
date_assign
date_created
date_resolve
Insert
Update
Delete
tasks_maint
{Error}
Project
Select Value
Priority
Select Value
Status
Select Value
Name
{task_name}
Description
{task_desc}
Resolution
{task_resolution}
Assigned To
Select Value
Assigned By
{user_id_assign_by}
{assigned_by}
Date Created
{date_assign}
{date_created}
Date Resolved
{date_resolve}
Add
Submit
Delete
Design HTML Code Preview Live Page
Toolbox
Builders Forms HTML
Properties
Format Data Events
Hidden: assigned_by
Name assigned_by
Caption
Source Type Database Colun
Control Source user_id_assign_
Data Type Integer
Format
Default Session.UserID
Required
Unique
Input Validation
Error Control
Validation Rule
Validation Text

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.
Configure the new field as follows:

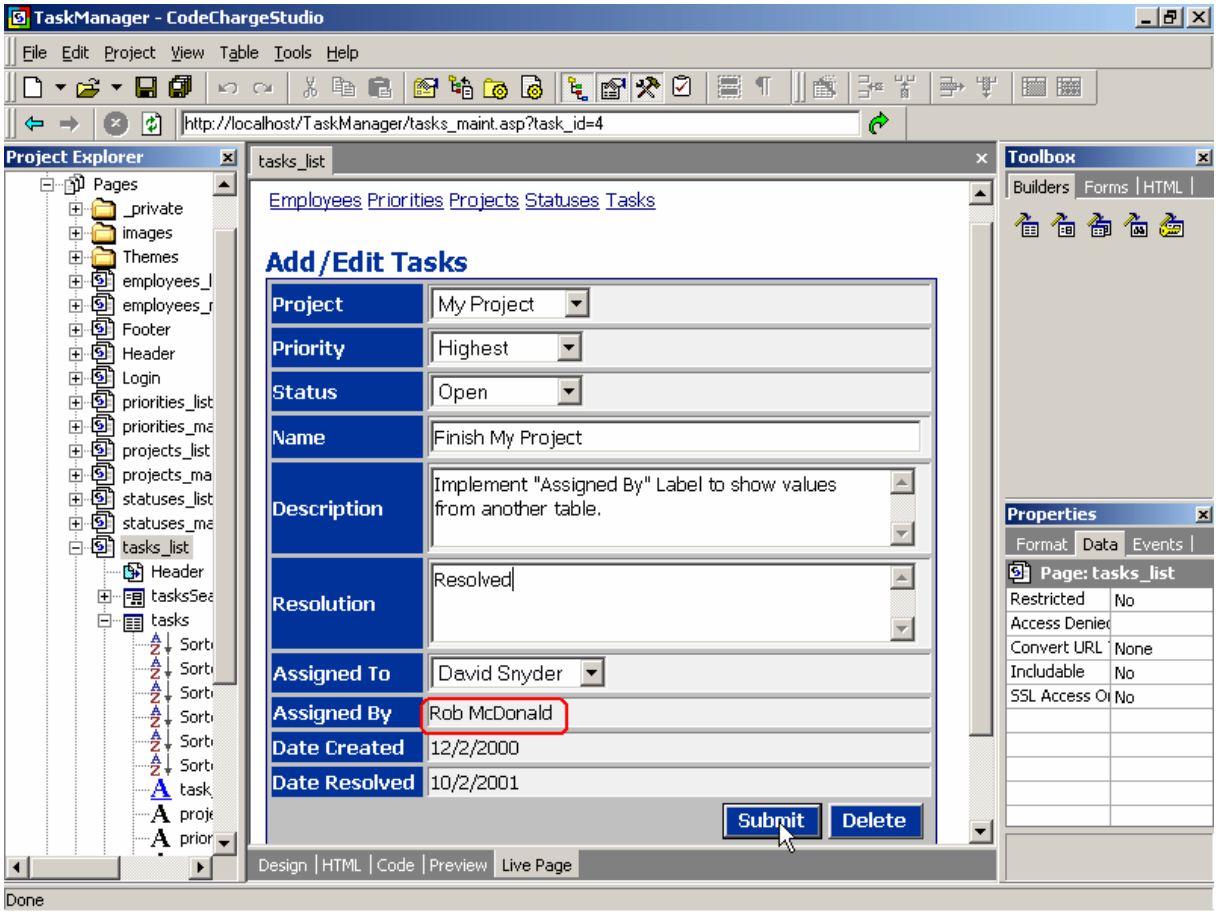
Name: *date_created*
Control Source: *date_assign*
Data Type: *Date*
Default: *CurrentDateTime* – The "CurrentDateTime" property allows you to automatically assign the current date and time to new tasks. The *Default* property doesn't affect existing records, thus the date/time of existing tasks won't be modified during updates.



Test the Label and Hidden Fields

Finally, you can switch to Live Page mode, select a Task, then login and see your Label display the name of the person who assigned the task.

The basic version of your Task Manager is now completed.
Don't forget to save it!



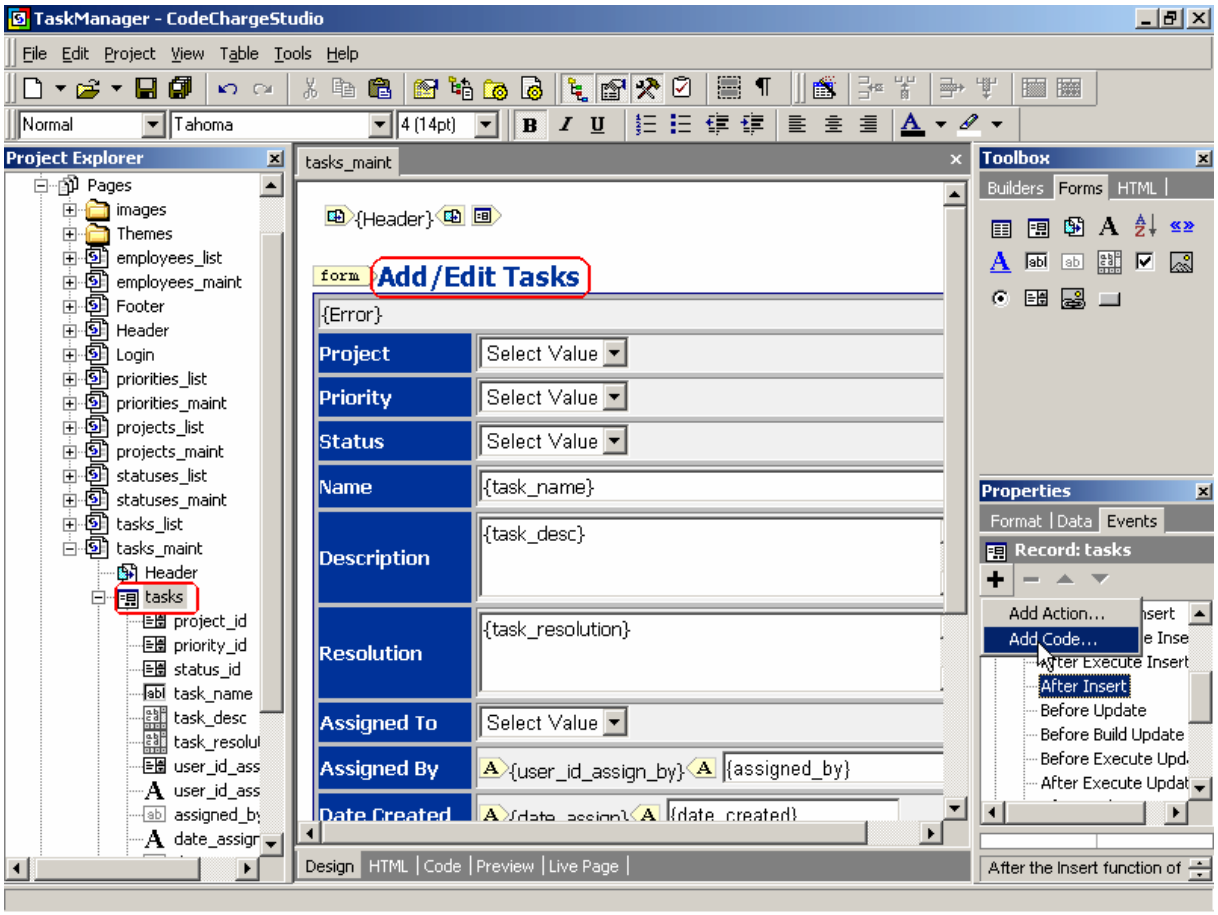
Programming the Record Form

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

Add Code in the "After Insert" Event to Send Emails

Select the "tasks" form by selecting it in the Project Explorer, or clicking anywhere within the form's caption. Then in the Properties window click on the "Events" tab and select the "After Insert" event.
Click on the [+] button, then select "Add Code…"



Once you're in the Code view, replace the generated comment:

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

with the code below:

```
<CF CCDLookUp   Field="emai1" Table="employees" Where="emp id=#flduser id assign to#" Connection="IntranetDB">
<CFSET mai1 To=CCDLookUp>
<CF CCDLookUp   Field="emai1" Table="employees" Where="emp id=#Session.UserID#" Connection="IntranetDB">
<CFSET mai1 From=CCDLookUp>
<CF CCDLookUp   Field="max(task_id)" Table="tasks" Where="user id assign by=#Session.UserID#"
Connection="IntranetDB">
<CFMAIL TO="#mai1 To#" FROM="#mai1 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 CCDLookUp   Field="emai1" Table="employees" Where="emp id=#flduser id assign to#" Connection="IntranetDB">
<CFSET mai1_To=CCDLookUp>
```

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.

```
<CF CCDLookUp   Field="emai1" Table="employees" Where="emp id=#Session.UserID#" Connection="IntranetDB">
<CFSET mai1_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 retrieves the id of the currently logged in user.

```
<CF CCDLookUp   Field="max(task id)" Table="tasks" Where="user id assign by=#Session.UserID#"
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="#mai1 To#" FROM="#mai1 From#" SUBJECT="New task for you" TYPE="HTML">
   The following task was submitted:<br><br>
   Task ID: #CCDLookUp#
   <br><br>#fldtask desc#
</CFMAIL>
```

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

```
   The following task was submitted:<br><br>
   Task ID: #CCDLookUp#
   <br><br>#fldtask desc#
```

The body of the email which consists of the task description and the task id.

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. Now implement similar functionality in “After Update” Event to notify assignee when an existing task is updated and reassigned to someone.
Click on the “tasks” form in the Project Explorer, then in the Properties window, select the “Events” tab, and then add the following “Custom Code” in “After Update” event:

```
<CF CCDLookUp   Field="emai1" Table="employees" Where="emp_id=#flduser_id_assign_to#"
Connection="IntranetDB">
<CFSET mai1 To=CCDLookUp>
<CF CCDLookUp   Field="emai1" Table="employees" Where="emp id=#Session.UserID#" Connection="IntranetDB">
<CFSET mai1 From=CCDLookUp>
<CF CCDLookUp   Field="max(task_id)" Table="tasks" Where="user id assign by=#Session.UserID#"
Connection="IntranetDB">
<CFMAIL TO="#mai1 To#" FROM="#mai1 From#" SUBJECT="New task for you" TYPE="HTML">
    The following task was assigned to you:<br><br>
    Task ID: #CCDLookUp#
   <br><br>#fldtask desc#
</CFMAIL>
```

Test Email Delivery

Before testing the system, you should add new users to your database with correct email addresses, or modify the existing test users by changing their email address. You can do this by opening the Intranet.mdb database that is located in your project directory. Alternatively, you may use the Task Manager itself and go to the Employees page to view and modify user emails there.
(Note: You will need Ms Access 2000 to manually edit the database file.)

Once you have users configured with test emails, save your project and switch to Live Page mode to test your system. In the Task Maintenance page, if you experience an error like "Invalid class string", it probably means that your ASPEmail component wasn't installed correctly. If your email component was properly installed, 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.

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 explain several of them.

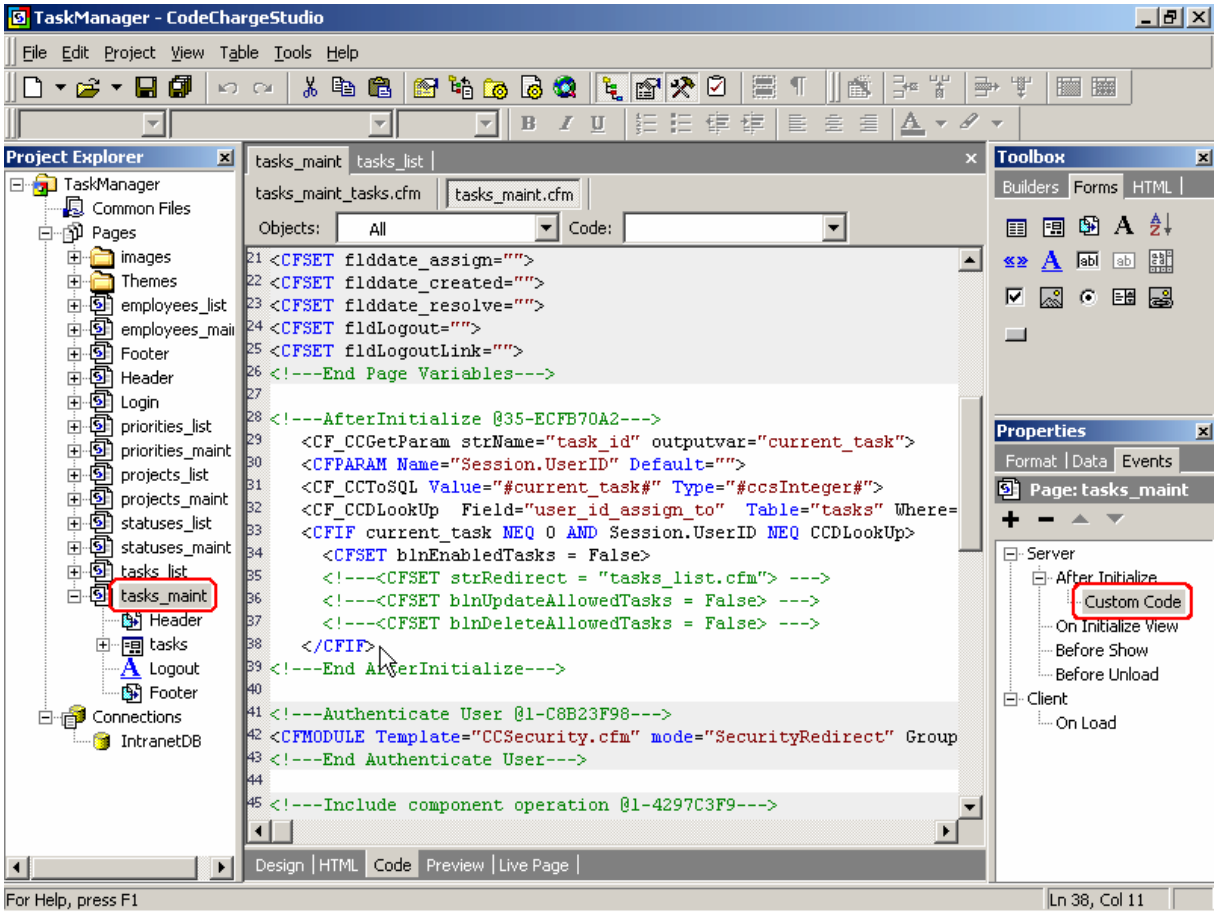
First, click on the “tasks_maint” page in the Project Explorer, then select “Events” tab in the Properties window. Add “Custom Code” to the “After Initialize” event of the page.

Once in the Code view, replace the generated comment:

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

with the code below:

```
<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 blnEnabledTasks = False>
        <!---<CFSET strRedirect = "tasks_list.cfm"> --->
        <!---<CFSET blnUpdateAllowedTasks = False> --->
        <!---<CFSET blnDeleteAllowedTasks = False> --->
        <!---<CFSET hideUpdate = True>--->
        <!---<CFSET hideDelete= True>--->
</CFIF>
```



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

1. **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 *blnEnabledTasks* 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 opposite, mutually exclusive criteria.

2. **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 blnEnabledTasks = False> --->
        <CFSET strRedirect = "tasks_list.cfm">
        <!---<CFSET blnUpdateAllowedTasks = False> --->
        <!---<CFSET blnDeleteAllowedTasks = False> --->
        <!---<CFSET hideUpdate = True>--->
        <!---<CFSET hideDelete= True>--->
        </
</CFIF>
```

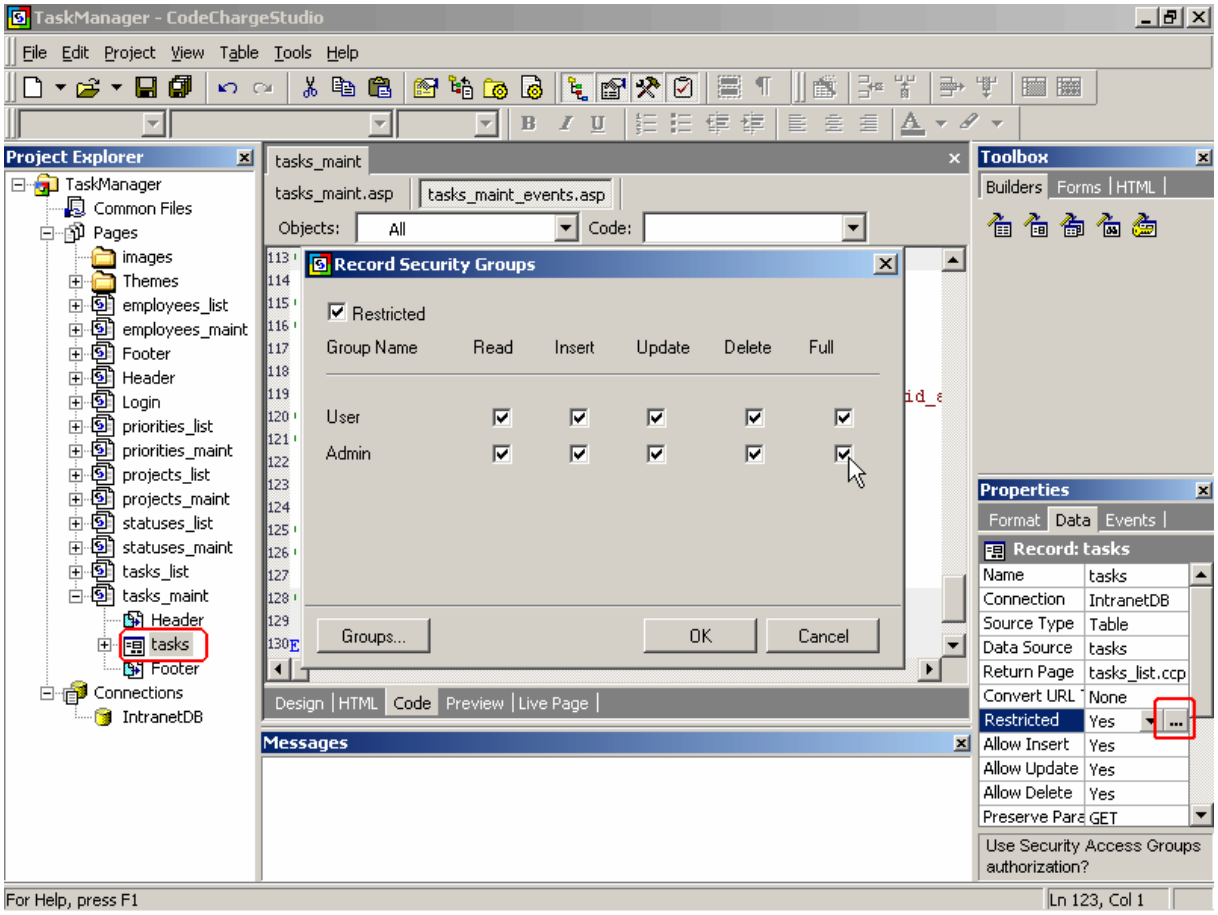
The above code shows that you should comment out the previously active line, and uncomment the line that starts with “Redirect”.
str*Redirect* is a variable used by CodeCharge Studio to determine if the current page should be redirected to another page, for example if a user is not logged in. This variable 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.

3. **Disable Update/Submit and Delete buttons for unauthorized users.**
Comment out the “Redirect” statement and uncomment the two lines of code below it, as shown here:

```
<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 blnEnabledTasks = False> --->
        <!---<CFSET strRedirect = "tasks_list.cfm"> --->
        <CFSET blnUpdateAllowedTasks = False>
        <CFSET blnDeleteAllowedTasks = False>
        <CFSET hideUpdate = True>
        <CFSET hideDelete= True>
        </
</CFIF>
```

This code shows how you can manipulate the *hideUpdate* and *hideDelete* variables within a record form. These variables control the appearance of the “Update” and “Delete” buttons on the page. If set to False, the buttons will not appear. Additional security is implemented via *blnUpdateAllowedTasks* and *blnDeleteAllowedTasks* variables to make impossible to update the record even if someone saves the page, adds the missing buttons and submits the page externally. However, the above code won’t work unless you also set the “Restricted” property

of your form to “Yes” and assign permissions to everyone. That’s because CodeCharge Studio generates the code appropriate for hiding and disabling buttons only when there is a need to do so, and restricting page access indicates that certain users are not allowed to add, update or delete records.



4. One more, although less secure method of disabling buttons on the record form is to hide the Update and Delete buttons on the page by adding the following custom code to the “Before Show” event of the form:

```
<CF_CCGetParam strName="task_id" outputvar="current_task">
<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 hideUpdate = True>
   <CFSET hideDelete= True>
</CFIF>
```

Appendix B – Additional Programming Examples

Listed here are additional programming example that may aid you in developing practical web applications.

Store User's Last Login Date and IP Address in the Database

To be able to save the additional information in the database, first add the appropriate columns to the *employees* table manually. For example:

last_login_ip (data-type text)
last_login_date (data-type date)

There are several places in the program(s) where you can place programming code that saves users' IP address and the date/time when they login

ASP/VBScript Adaptation

1. **Modify Common.asp**
Modify the standard login function used by CodeCharge Studio, which is stored in Common.asp. Double-click on the "Common Files" in Project Explorer, then find the function CCLoginUser within the Common.asp code and modify it as follows:

From:

```
If Result Then
   Session("UserID") = RecordSet("emp_id")
   Session("UserLogin") = Login
   Session("GroupID") = RecordSet("group_id")
End If
```

To:

```
If Result Then
   Session("UserID") = RecordSet("emp_id")
   Session("UserLogin") = Login
   Session("GroupID") = RecordSet("group_id")
   Connection.Execute("UPDATE employees SET last_login_date=#" & Now() & "#, last_login_ip=" &
      CCToSQL(Request.ServerVariables("REMOTE_ADDR"),"Text") & " WHERE emp_id=" &
      CCToSQL(RecordSet("emp_id"),"Integer"))
End If
```

The above code executes a database query that updates the *employees* table.
The following code uses a VBScript function to retrieve user's IP address:

```
CCToSQL(Request.ServerVariables("REMOTE_ADDR"),"Text")
```

CCToSQL function is used to add apostrophes around the string, which are needed in the SQL statement.

Here is an example of an SQL statement created and executed by the above code:

```
UPDATE employees SET last_login_date=#03/25/2002#, last_login_ip='127.0.0.0' WHERE emp_id=12
```

2. **Use “On Click” event on the Login page**
You can also update the database within the “On Click” event procedure of the Login button on the Login page. CodeCharge Studio automatically creates an On Click event for the Login page and you can also add your own additional code there.
Select the Login button (sometimes named “DoLogin”) on the Login page and add custom code to the “On Click” event, below the existing “Login” action.

Place the following code in your new event:

```
Dim Connection
  If Login_DoLogin_OnClick = True Then
  Set Connection = New clsDBIntranetDB
  Connection.Open
    Connection.Execute("UPDATE employees SET last_login_date=#" & Now() & "#, last_login_ip=" &
    CCToSQL(Request.ServerVariables("REMOTE_ADDR"),"Text") & " WHERE emp_id=" &
    CCToSQL(CCGetUserID,"Integer"))
  Connection.Close
  Set Connection = Nothing
End If
```

ASP.NET(C#) Adaptation

1. **Modify DataUtility.cs**
Modify the standard login function used by CodeCharge Studio, which is stored in *DataUtility.cs* file. Double-click on the "Common Files" in "Project Explorer", then find the method *CheckUser* within the *DBUtility* class and modify it as follows:

From:

```
if(ds.Tables[0].Rows.Count>0)
{
  HttpContext.Current.Session["UserID"]= ds.Tables[0].Rows[0]["emp_id"];
  HttpContext.Current.Session["GroupID"]= ds.Tables[0].Rows[0]["group_id"];
  HttpContext.Current.Session["UserLogin"]= UserName;
  return true;
}
```

To:

```
if(ds.Tables[0].Rows.Count>0)
{
  HttpContext.Current.Session["UserID"]= ds.Tables[0].Rows[0]["emp_id"];
  HttpContext.Current.Session["GroupID"]= ds.Tables[0].Rows[0]["group_id"];
  HttpContext.Current.Session["UserLogin"]= UserName;

  DataAccessObject ipCmd = Settings.IntranetDBDataAccessObject ;

  string sqlStr = "UPDATE employees SET last_login_date=#" + DateTime.Now +
 "#, last_login_ip='"+HttpContext.Current.Request.ServerVariables["REMOTE_ADDR"]
  + "' WHERE emp_id=" + ds.Tables[0].Rows[0]["emp_id"] ;

  ipCmd.ExecuteNonQuery( sqlStr ) ;
  return true;
}
```

The above code executes a database query that updates the *employees* table.
The above code uses a ASP.NET object *HttpContext.Current.Request.ServerVariables["REMOTE_ADDR"]* to retrieve user's IP address.
Here is an example of an SQL statement created and executed by the above code:

```
UPDATE employees SET last_login_date=#03/25/2002#, last_login_ip='127.0.0.0' WHERE emp_id=12
```

PHP Adaptation

1. **Modify Common.php**
Modify the standard login function used by CodeCharge Studio, which is stored in Common.php. Double-click on the "Common Files" in Project Explorer, then find the function CCLoginUser within the Common.php code and modify it as follows:

From:

```
  if ($Result)
  {
    CCSetSession("UserID", $db->f("emp_id"));
    CCSetSession("UserLogin", $Login);
    CCSetSession("GroupID", $db->f("group_id"));
  }
```

To:

```
if ($Result)
{
    CCSetSession("UserID", $db->f("emp_id"));
    CCSetSession("UserLogin", $Login);
    CCSetSession("GroupID", $db->f("group_id"));
    $db->query("UPDATE employees SET last_login_date=NOW(), last_login_ip="
      . $db->ToSQL(getenv("REMOTE_ADDR"), ccsText) . " WHERE emp_id="
      . $db->ToSQL($db->f("emp_id"), ccsInteger));
}
```

The above code executes a database query that updates the *employees* table.
The following code uses a PHP function to retrieve user's IP address:

```
$db->ToSQL(getenv ("REMOTE_ADDR"), ccsText)
```

The ToSQL() function is used to add quotation marks around the string, which are needed in the SQL statement.

Here is an example of an SQL statement created and executed by the above code:

```
UPDATE employees SET last_login_date=NOW(), last_login_ip='127.0.0.0' WHERE emp_id=12
```

2. **Use the "On Click" event on the Login page**
You can also update the database using the "On Click" event of the Login button on the Login page. CodeCharge Studio automatically creates an On Click event for the Login page and you can also add your own additional code there.

Select the Login button (sometimes named "DoLogin") and modify the Login action in "On Click" event.
From:

```
else
{
    global $Redirect;
    $Redirect = CCGetParam("ret_link", $Redirect);
    return true;
}
```

To:

```
else
{
    global $Redirect;
    $Redirect = CCGetParam("ret_link", $Redirect);
    $db->query("UPDATE employees SET last_login_date=NOW(), last_login_ip="
        . $db->ToSQL(getenv("REMOTE_ADDR"), ccsText) . " WHERE emp_id="
        . $db->ToSQL(CCGetUserID(), ccsInteger));
    return true;
}
```

Appendix C - Common Errors

Listed below are common issues and errors with running ASP applications on Windows servers.

Operation must use an updateable query. (Microsoft JET Database Engine)

This is one of the most common ASP errors that occurs when the data is being updated in a Microsoft Access database that doesn't have sufficient access privileges.

Solution:
In Windows 95, 98 or ME: Right click on the MS Access file (.mdb) and uncheck the "Read-only" property.
In Windows NT, 2000 or XP: Ask your system administrator to setup full access permissions for the anonymous user account called IUSR_<Machinename> to access the database.

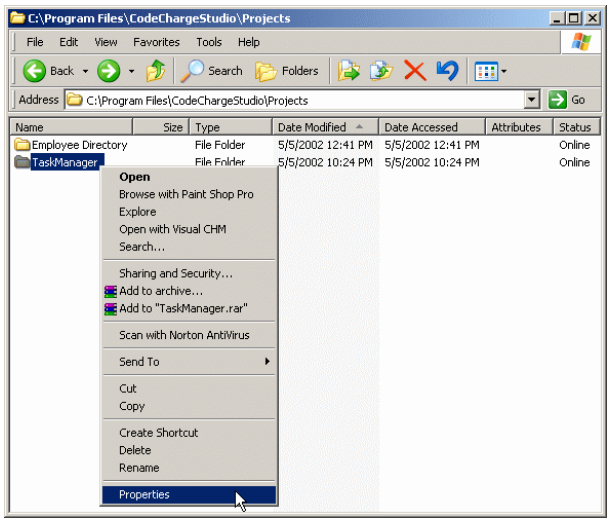
If you are hosting your website externally, your web hosting company most likely provides a special read-write folder for databases or will create one for you. Please check their FAQ and other information on your hosting company's website or contact them directly.

For more information, refer to: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q175168

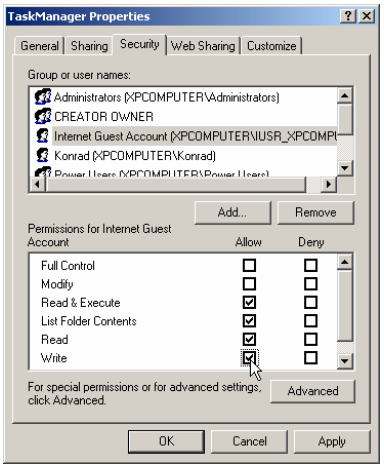
Microsoft JET Database Engine (0x80004005) Could not use ''; file already in use.

This error usually happens on Windows NT/2000/XP when your database file or database folder doesn't have write permissions. You can often solve it by following these steps:

1. Right-click on the project folder on your disk and select "Properties".



2. In the Security tab add Internet Guest Account and set Write permissions checkbox.



For more information, refer to: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q174943

HTTP 500 Internal Server Error

This message is a “friendly” representation of a non-standard error that occurred on the web server.
To see the full error message, turn off “friendly error messages in your web browser”.

For more information, refer to: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q294807

Page takes forever to load or the IIS web server appears to hang

This issue is usually caused by Norton AntiVirus being configured to block script execution.
If you have Norton AntiVirus installed on your machine, disable Script Blocking as shown below.