Sample Scenario
This sample project imports contents of a text file into a database. The text file contains lines of text. Each line contains 4 strings representing date, time, and two numbers:

2010-10-29 11:45:01 0000118390 00001280
2010-10-29 12:00:01 0000118709 00001281
2010-10-29 12:15:00 0000119121 00002418
2010-10-29 12:28:10 0000119636 -0000004
2010-10-29 12:28:22 0000120000 -0000001
2010-10-29 12:28:29 0000130000 -0000001
2010-10-29 12:35:43 0000130000 00003600
2010-10-29 12:55:57 0000130086 00001287
2010-10-29 13:00:02 0000130404 00001272

We use a database table to store each line into 3 fields: DateTime, Number1, and Number2.
We will create a method to do the task.

In this method, we use a StreamReader class to read the text file line by line, until the end of file is reached. For each line, split the line into a string array of 4 items.

An EasyUpdater is used to insert each line into the database, using the string array items.

**Prepare Database Connection**

**Create Database Connection**

Right-click “Database Connections Used”, choose “Database Connections”:

![Database Connections](image1)

Click to create a new database connection:

![Create a new database connection](image2)

Give a name to the new connection:

![Database connections](image3)

We may use MySQL ADO.Net driver or MySQL ODBC driver to connect to a MySQL database. For details, see

[http://www.limnor.com/support/ConnectToMySqlWithADO.pdf](http://www.limnor.com/support/ConnectToMySqlWithADO.pdf)
Note that when we use 32-bit MySql driver, we need to set the project platform to x86:

Create Record Insert Component
EasyUpdator class can be used for inserting records into database:
Set ExecutionCommand property to use the EasyUpdater1 to insert record into database:

Use the “Database command builder” to build a SQL script for inserting record into database.

1. Select table “ImportData” which is the table new records will be created
2. Select “Insert” under “Command type” for inserting record
3. Check each field for specifying value for the new record. In this sample, we type in a parameter for each field: @datetime, @num1, @num2. A parameter is formed by a “@” symbol followed by a parameter name.

![Database command builder](image)

Click OK button. A dialogue box appears listing all the parameters. This is for verifying the data types of the parameters. You may correct the data types if needed. In this sample, the system gives correct data types for the parameters:
Create method for file importing

Create method with filename parameter

Right-click the “Event Map”, choose “Create method”:

Give a method name:
Right-click the parameter list to add a parameter for file name:

Select “String” as the data type of the parameter:

Select the newly created parameter:

Name the parameter as “filename”:

Use StreamReader to read file
Right-click the Variable Pane or Action Pane of the Method Editor, choose “Add local variable”: 
Find and select the StreamReader class:

The StreamReader class allows us to supply some values when creating an instance. In our sample, we may choose the constructor requiring a file path:
An action appears in the Action Pane. The action creates a local variable, StreamReader1. The action has a path parameter. We will assign the “filename” parameter of the method to the “path” parameter of the action. Select Property for the “path”:

Select the filename under the method; click Next:
Read text file line by line
The StreamReader class has a ReadLine method which can be used to read each line of the text file.

One issue we need to deal with is how to read each line again and again until the end of the file. This is done by creating a repeating action.

Right-click the Action Pane of the Method Editor; choose “Execute actions repeatedly”: 
An action named “Repeat1” appears in the Action Pane. Link its in-port with the out-port of existing action to form an execution sequence:

Set the Condition for the action:

Click the Property icon:
Double-click the Property item:

Select the EndOfStream property of the variable StreamReader1; click Next

Click “not” icon:
This is the condition for Repeat1:

```
NOT StreamReader1.EndOfStream
```

This condition indicates that all actions contained within Repeat1 will be executed again and again until end of the file is reached. Now we may add actions into Repeat1.

Right-click Repeat1, choose “Edit Repeated Actions”

To read a line from the file, right-click StreamReader1, choose “Create action”. Choose “ReadLine”:
“AssignTo” property of the action indicates where the line of the text read from the file will be stored. Choose “New local variable” to create a new variable to store the line read from the file:

This action reads a line of text from the file and stores the line of the text in a string variable named String1:
The action appears in the Action Pane:

When Repeat1 is executed since this ReadLine action is contained within Repeat1, it will be executed again and again until the end of file is reached. Starting from the first line, each time ReadLine reads the next line from the file. On reading the last line, the EndOfStream property of StreamReader1 becomes True, and thus stops the execution of Repeat1.

Parse a line of text and save to database
We are still working within Repeat1. We have created an action to read a line of text into a string variable named String1. Because the data in a line of text are separated by spaces, we may use Split method of String1 to parse the data.

Right-click String1; choose “Create action”; choose “Split”: 
Set its "separator":

Click Add to add a separator:
Select “Value” and type a space:

It is not easy to see a space, so, be careful to get it right. Click OK.

Choose “New local variable” for “AssignTo”: 
This action separates contents in String1 into an array and stores the array in a variable named Array1:

The new action appears in the Action Pane. Link it to the existing action:
For example, if a line of text is as shown below:

2010-10-29 11:45:01 0000118390 00001280

Then on executing the Split action, Array1 becomes an array of 4 items:

2010-10-29
11:45:01
0000118390
00001280

We may create an action to save these items into the database.

Right-click EasyUpdater1; choose “Create action”; choose “ExecuteWithParameterValues”

Choose “Math Expression” for “@datetime” of the action:
The value for `@datetime` is formed by the first two items of Array1, separated by a space. For example, “2010-10-29 11:45:01”

Click A+ icon to add two strings into the expression:

Select one string item and click A+ again to add one more string into the expression:
Select the first string. We want to get the first element of Array1. So, we may click the Method icon:

Double-click the method item:

Select Get method of Array1; click Next:
0 indicates the first array item:

Select the second string; type a space:
Select the third string; click the Method icon:

![Math Expression Editor](image)

Array1[0] " " A method call

Double-click the method item:

![Math Expression Editor](image)

Array1[0] " " ?

Select the Get method of Array1; click Next:

![Object picker](image)
By default the first array element is used:

```
Array1 [0] " " Array1 [0]
```

To get the second array element, select the array index:

```
Array1 [0] " " Array1 [0]
```

Type 1 to indicate the second array element:

```
Array1 [0] " " Array1 [1]
```

Now we have an expression formed by the first and second array elements, separated by a space:
For @num1, we want to use the 3rd element of Array1. So, select “Math Expression”:

Click Method icon:

Double-click the method item:
Select the Get method of Array1; click Next:

Select the array index:
Type 2 to indicate the 3rd array element:

For @num2, we want to use the 4th array element. So, select “Math Expression”:

This time we use 3 as the array index to indicate the 4th array element:
This action creates a new record in table ImportData, using elements from Array1:

This action appears in the Action Pane. Link it to the last action:

Those are all the actions we added to Repeat1. Note that those actions will be executed as many times as the number of lines in the text file.

The first action reads a line of text from the file.
The second action splits the line of text into an array of strings.

The third action creates a new record in the database using the array elements.

**Execute Method**

We have created a method to import text file into a database. To use the method, we may create an action using the method.

Right-click the method; choose “Create action”:

![Create action screenshot](image)

We use a TextBox for the text file name. So, we select “Property” for filename parameter of the action:

![Property selection screenshot](image)

Select the Text property of the TextBox:
This action executes ImportTextFile method, passing the Text property of the TextBox as the filename:

The new action appears under the method:
Assign this action to the button by right-clicking the button and choosing “Assign Action”. Select “Click” event:

Select the action. Click Next:
Now we may test the application:

The form appears:
Let's view the records of ImportData table before test. We can see that it is an empty table:

Click the button:

View the records again. This time the table contains 11 records:

Feedback
Please send your feedback to support@limnor.com