How To Create A New LabServer

Draft - 8 Dec 2009 - LJP

The following assumes that you are using Microsoft Visual Studio 2008 on a Microsoft Windows operating system with Internet Information Service (IIS) installed.

Two parts are required to create a new LabServer:

- **LabLibraries** – Code common to all LabServers which contains the base classes and other classes required for LabServer operation. Download a GNU tarball of the 'LabLibraries' folder from SourceForge.
- **Template** – A template or skeleton for the new LabServer implementation which contains the derived classes, setup and equipment drivers that may be required for LabServer operation. It can be compiled and run in its current state just to show something running before code gets changed for the new LabServer implementation. Download a GNU tarball of the 'Template' folder from SourceForge.

This part is optional but is good to have around as a reference:

- **TimeOfDay** - This example is one of several that are provided for comparison and gives an idea of the changes that need to be made to create a new experiment. It's a lot easier to 'Copy-Paste-Change' than it is to think it up from new. Download a GNU tarball of the 'TimeOfDay' folder from SourceForge.

Extract the 'LabLibraries' GNU tarball and the 'Template' GNU tarball so that the folders './LabLibraries/...' and './Template/...' exist in the same folder. Open the Visual Studio solution './Template/Template.sln'. There are ten projects in the solution:

- DummyServiceBroker
- LabClientHtml
- LabEquipment
- LabServer
- LibraryLab
- LibraryLabClient
- LibraryLabEquipment
- LibraryLabEquipmentEngine
- LibraryLabServer
- LibraryLabServerEngine

You only need to be concerned with these five:

- LabClientHtml
- LabEquipment
- LabServer
- LibraryLabEquipment
- LibraryLabServer

DO NOT make changes to the other projects! Well, you may want to later on but just get something going first.

DummyServiceBroker

This project is provided in the solution to allow development of the LabServer and LabClient in an enviroment without an iLab ServiceBroker getting in the way. The 'DummyServiceBroker' simply provides pass-through methods to allow LabClient-to-LabServer communication. It also generates experiment IDs when experiments are submitted.

Development Server Ports

The 'LabClientHtml' web application and the 'LabEquipment', 'LabServer' and 'DummyServiceBroker' web service applications use ports on 'localhost' and there is no need to use IIS Virtual Directories. This makes it much easier to move the solution to different development computers or even folders or drives on a single computer. To check these settings, open the project's 'Properties' page and select the 'Web' tab. The 'Using Visual Studio Development Server' will be selected and 'Specific Port' will be assigned a value at or above '8081'.

Compiling Template

Check that the 'Active solution configuration' is set to 'Debug' and not 'Release'. This can be checked by selecting 'Build->Configuration Manager...' from the menu. Compile the solution. It should compile without any errors or warnings. Don't run the solution yet because there are a few things that need to be done first.

Configuration

LabEquipment

In the folder './Template/LabEquipment', copy 'Web.config.template' to 'Web.config' and update the following keys:

- AllowedCaller - This is used to only allow a particular LabServer to communicate with the LabEquipment. For development, set to 'localhost, 8C0BA543419E4d4ab340d449acd3e057, fd3cf16cc855484fb06801379f475837, unused, u
- AuthenticateCaller - For development, you may choose to set this to 'false'.

In the 'system.web' section, you will need to change the value of the 'compilation' key from 'debug="false"' to 'debug="true"'.

LabServer

In the folder './Template/LabServer', copy 'Web.config.template' to 'Web.config' and update the following keys:

- LabServerGuid - For development, use '8C0BA543419E4d4ab340d449acd3e057'.
- AllowedCaller - This is used when the LabServer is configured on a ServiceBroker. For development, set to 'localhost, 13B164323CDE4aaaA7E2884F0B2F6110, fd3cf16cc855484fb06801379f475837, http://loc
- AuthenticateCaller - This is used when the LabServer is configured on a ServiceBroker. For development, you may choose to set this to 'false'.
- EquipmentService - For development, set to 'http://localhost:8087/EquipmentService.asmx, fd3cf16cc855484fb06801379f475837'.

In the 'system.web' section, you will need to change the value of the 'compilation' key from 'debug="false"' to 'debug="true"'.

The LabServer uses a database for the experiment queue, results and statistics. So we need to create a database and run some SQL scripts that will create the tables and stored procedures in the database.

The following procedure assumes that you have installed Microsoft SQL Server Express and Microsoft SQL Server Management Studio Express.

- Create a new database with the name 'Template_LabServer'. This name is specified in the 'SqlConnection' key in the Web.config file.
- In Object Explorer, select Security->Logins. For Windows XP, select MachineName\ASPNET where 'MachineName' is the name of the computer that this database is created on. (For Windows Server 2003, select NT AUTHORITY\NETWORK SERVICE). Right-click and select Properties. In the 'Login Properties' window, select 'User Mapping'. Find the Template_LabServer database and then check 'Map' in upper box and check 'db_owner' in the lower box. Click OK. Review the User Mapping. For each database, the Default Schema should now be set to 'dbo'.
- Open a Command Prompt window, change the directory to './Template/Published/DB_Scripts'. Run the batch file 'Template_LabServer.bat'. Check the log file 'Template_LabServer.log' to ensure that the scripts executed successfully.

- In Microsoft SQL Server Management Studio Express, refresh the database. You should now see three new tables and a number od stored procedures.

LabClientHtml

In the folder './Template/LabClientHtml', copy 'Web.config.template' to 'Web.config' and update the following keys:

- ServiceBrokerUrl - For development, set to 'http://localhost:8081/ServiceBrokerService.asmx'.
- LabServerGuid - For development, use '8C0BA543419E4d4ab340d449acd3e057'.
- FeedbackEmail - Set this to your email address in the form, 'you@your.email.address'.

In the 'system.web' section, you will need to change the value of the 'compilation' key from 'debug="false"' to 'debug="true"'.

DummyServiceBroker

The 'Web.config' file has already been supplied and does not need to be changed. 'DummyServiceBroker' is only used during development and does not get 'Published' in the 'Release' version.

Running the Template Solution

There is no need to compile the solution after making changes to the configuration files. But let's not jump in and press 'F5' just yet. We need to run up the solution in an orderly fashion.

LabEquipment

We need to check that this one runs ok. Open the 'Properties' page for the 'LabEquipment' project and select the 'Web' tab. Check that 'Start Action' is set to 'Specific Page' with the entry 'EquipmentService.asmx'. Now, right-click the 'LabEquipment' project in 'Solution Explorer' and select 'Debug->Start new instance' from the popup menu. If all goes well, you should be presented with the 'EquipmentService' webpage showing a number of operations with the first one being 'ExecuteRequest'.

Click on 'GetLabEquipmentStatus' to run the operation. A 'Test' webpage appears where the operation can be invoked. Click the 'Invoke' button. You should be presented with another webpage containing some information in XML format. If 'statusMessage' contains the string 'Access Denied!' then close the web browser and edit the 'Web.config' file for the 'LabEquipment' project. Set the 'AuthenticateCaller' key value to 'false' and then save and close the 'Web.config' file. Run the procedure above again. You should see 'online' should set to 'true' and 'statusMessage' set to 'LabEquipment is ready.'.

Let's check the log file to see what was going on. Open the file './LabEquipment/LogFiles/YYYYMMDD.log' where 'YYYY' is the year, 'MM' is the month and 'DD' is the day that the logile was created. The logfile contains a lot of information but gives the developer some idea of what's going on. If things didn't run as expected, check the logfile to find a possible cause.

Close the 'EquipmentService' browser window.

LabServer

We now need to check that this one runs ok. Open the 'Properties' page for the 'LabServer' project and select the 'Web' tab. Check that 'Start Action' is set to 'Specific Page' with the entry 'LabServerWebService.asmx'. Now, right-click the 'LabServer' project in 'Solution Explorer' and select 'Debug->Start new instance' from the popup menu. If all goes well, you should be presented with the 'LabServerWebService' webpage showing several operations including 'GetLabStatus'.

Click on 'GetLabStatus' to run the operation. A 'Test' webpage appears. No parameters can be entered for 'GetLabStatus' because it doesn't take any. Click the 'Invoke' button. You should be presented with another webpage containing some information in XML format. If 'labStatusMessage' contains the string 'Access Denied!' then close the web browser and edit the 'Web.config' file for the 'LabServer' project. Set the 'AuthenticateCaller' key value to 'false' and then save and close the 'Web.config' file. Run the procedure above again. This time, 'online' should be 'true' and 'labStatusMessage' should be contain the string 'Ready'.

Check the log file for the 'LabServer' project to see what was going on. Open the file './LabServer/LogFiles/YYYYMMDD.log' just like you did for the 'LabEquipment' project. If things didn't didn't run as expected, check the logfile to find a possible cause.

Close the 'LabServerWebService' browser window.

In 'Solution Explorer', expand the 'LabServer' project. Right-click on 'Administration.aspx' and select 'Set As Start Page' from the popup menu. Now, right-click the 'LabServer' project in 'Solution Explorer' and select 'Debug->Start new instance' from the popup menu. If all goes well, you should be presented with an 'Administration' webpage with 'Template LabServer' in the banner. Have a look at the experiment statistics, results and queue by clicking the 'Download' button for each. In each case, you will be presented with an XML file, although they will all be empty at this time.

Close the 'Template LabServer' browser window.

DummyServiceBroker

We now need to check that this one runs ok. Open the 'Properties' page for the 'DummyServiceBroker' project and select the 'Web' tab. Check that 'Start Action' is set to 'Specific Page' with the entry 'ServiceBrokerService.asmx'. Now, right-click the 'DummyServiceBroker' project in 'Solution Explorer' and select 'Debug->Start new instance' from the popup menu. If all goes well, you should be presented with the 'ServiceBrokerService' webpage showing a number of operations. We don't need to invoke any of these operations.

Close the 'ServiceBrokerService' browser window.

LabClientHtml

Finally, we now need to check that this one runs ok. Open the 'Properties' page for the 'LabClientHtml' project and select the 'Web' tab. Check that 'Start Action' is set to 'Specific Page' with the entry 'LabClient.aspx?couponID=12345&passkey=100453924900132'. This is the same URL that the ServiceBroker launches except that the numbers will be different. The 'DummyServiceBroker' only uses 'couponID' and 'passkey' is ignored.

Either press 'F5' or right-click the 'LabClientHtml' project in 'Solution Explorer' and select 'Debug->Start new instance' from the popup menu. If all goes well, you should be presented with a webpage displaying the title 'Template', an image showing some resistors on a circuit board and a navigation menu.

Check the log file for the 'LabClientHtml' project to see what was going on. Open the file './LabClientHtml/LogFiles/YYYYMMDD.log' just like you did for the 'LabServer' project. If things didn't run as expected, check the logfile to find a possible cause.

In the navigation menu, click 'Status'. The 'LabServer' should be 'Online' and 'Ready' with '0 experiments queued'.

In the navigation menu, click 'Setup'. There are two setups provided, one for driving the equipment and one for driving the simulation. These have almost no functionality but are provided as an example to work with. For each setup, click the 'Validate' button to validate the experiment specification. An estimated execution time will be shown in the message.

Select the setup 'Setup for Equipment' and click the 'Submit' button to submit an experiment specification to the LabServer for processing. Then go to the 'Status' webpage to check the status of the experiment just submitted. The experiment number should be displayed so click the 'Check' button. A message will be displayed indicating the status of the experiment.

When the experiment completes, navigate to the 'Results' webpage. The experiment number should be displayed so click the 'Retrieve' button. Information about the experiment should be displayed. The experiment results can be saved to a CSV file by clicking the 'Save' button.

Close the LabClient browser window.

Let's have another look at the LabServer's 'Administration' webpage. Examine the experiment statistics, results and queue by clicking the 'Download' button for each. This time, the XML file will contain information for the experiment statistics and results. Although, the experiment queue will again be empty.

Changing the Code - What to Change or Add

In the descriptions that follow, references will be made to comments inserted in the code. These comments are either:

```
//
// YOUR CODE HERE
//
```

or

```
//
// YOUR CODE BETWEEN HERE ...

// ... AND HERE.
//
```

This makes it a bit easier to know where to change or add code.

LibraryLabEquipment

There are five files here that need to be changed:

- Drivers\Hardware.cs
- Consts.cs
- EquipmentEngine.cs
- EquipmentManager.cs
- TypeCommandInfo.cs

Drivers\Hardware.cs

This is a skeleton of a simple hardware driver. Use this as a starting point for your own development.

Consts.cs

Put all of your XML string constants here.

EquipmentEngine.cs

This creates an instance of the hardware driver. There may be multiple hardware drivers depending on the equipment being used. The equipment may also need to be powered down when not in use. Processing of command requests are carried out here.

EquipmentManager.cs

This creates an instance of the EquipmentEngine. It also processes the XML command requests received from the LabServer via the EquipmentService. For non-execute commands, those that get the time to do something, it calls upon the hardware drivers to execute these commands. The execute commands, those that cause the hardware to do something, the requests are passed to the EquipmentEngine where the equipment may need to be powered up and initialised before the command can be executed.

TypeCommandInfo.cs

This contains the enumerated types for the non-execute and execute commands.

LabEquipment

There is just one file here that needs to be changed:

- App_Data\EquipmentConfig.xml

App_Data\EquipmentConfig.xml

This file contains configuration information required by the hardware drivers to configure and operate the equipment.

LibraryLabServer

This project contains the functionality specific to a LabServer. There are several files here that need to be changed:

- Drivers\Module\Simulation.cs
- Drivers\Setup\DriverEquipment.cs
- Drivers\Setup\DriverEquipmentExecTime.cs
- Drivers\Setup\DriverEquipmentExecute.cs
- Drivers\Setup\DriverSimulation.cs
- Drivers\Setup\DriverSimulationExecTime.cs
- Drivers\Setup\DriverSimulationExecute.cs
- Configuration.cs
- Consts.cs
- ExperimentEngine.cs
- ExperimentManager.cs
- ExperimentResult.cs
- Specification.cs
- TypeResultInfo.cs
- Validation.cs

Configuration.cs

Takes information from the 'Configuration' element of 'LabConfiguration.xml' and places it in private variables that can be accessed through 'Properties'. This is information is not the setups. The setups are already handled by the base class in 'LibraryLabServerEngine.LabConfiguration'.

Consts.cs

String constants and XML string constants are placed here in one place. It makes them easier to find and helps to eliminate typing errors by using 'constants' instead of 'strings'.

ExperimentEngine.cs

This creates an instance of the driver for the specified setup that will execute the experiment.

ExperimentResult.cs

Information is taken from the Specification and ResultInfo structure and placed in the XML ExperimentResult node which is then saved to the database. This information is available for retrieval by the ServiceBroker.

Specification.cs

Takes information from the XML Specification string and places it in private variables that can be accessed through 'Properties'. This information is then checked for validity. If the specification is valid, an instance of the driver for the specified setup is created to obtain the estimated execution time.

TypeResultInfo.cs

The 'ResultInfo' structure that contains the information that will be placed in the XML ExperimentResult node.

Validation.cs

May or may not be used, depending on the experiment. Boundary conditions such as 'Minimum' and 'Maximum' values are checked here.

Drivers\Module\Simulation.cs

This is a skeleton of a simple simulation driver. Use this as a starting point for your own development.

Drivers\Setup\DriverEquipment.cs
Drivers\Setup\DriverEquipmentExecTime.cs
Drivers\Setup\DriverEquipmentExecute.cs

These 'partial' classes drive the experiment for a particular 'setup' which makes calls to a LabEquipment web service.

Drivers\Setup\DriverSimulation.cs
Drivers\Setup\DriverSimulationExecTime.cs
Drivers\Setup\DriverSimulationExecute.cs

These 'partial' classes drive the experiment for a particular 'setup' which runs a simulation.

LabServer

There are three files here that need to be changed:

- App_Data\LabConfiguration.dtd
- App_Data\LabConfiguration.xml
- App_Data\SimulationConfig.xml
- Web.config

LabConfiguration.xml

This file contains the information required by the LabClient to display information on the 'Setup' webpage and populate the controls.

The LabClient may want to display various versions of an experiment ('Experiment Setup') that the user can choose from. The 'Configuration' element contains a list of the setups to display. Each setup has an 'id', 'name' and 'description'. The 'id' is used by both the LabServer and LabClient to determine what operations to perform. The 'name' and 'description' elements are displayed in fields in the 'Setup' webpage.

The 'Configuration' element may contain elements with information common to all listed setups.

An empty XML Validation element is provided so that the LabServer can determine if information in the experiment specification is valid.

An empty XML Specification element is provided so that the LabClient can fill in information and send it to the LabServer.

An empty ExperimentResults element is provided so that the LabServer can fill in information and send it to the LabClient.

SimulationConfig.xml

This file contains configuration information required by the simulation drivers to configure and operate the simulation.

Web.config

The SQL database connection string is specified here. Change the name of the database for your implementation.

The equipment web service URl and passkey are specified here. When the equipment web service is 'published', the URL will need to be changed to reflect to new location of the service.

The size of the LabServer farm is specified here. If there is only one set of equipment, the farm size will be just one.

LabClientHtml

There are several files here that need to be changed:

- LabControls\LabConsts.cs
- LabControls\LabResults.ascx.cs
- LabControls\LabSetup.ascx
- LabControls\LabSetup.ascx.cs
- LabControls\Result.cs
- LabControls\TypeResultInfo.cs
- Styles\LabControls.css
- Web.config

LabConsts.cs

XML string constants and string constants are placed here in one place. It makes them easier to find.

Used by 'LabResults.ascx.cs', 'LabSetup.ascx.cs' and 'Result.cs'.

LabResults.ascx.cs

Obtains the experiment specification and result information strings that are used by 'Results.ascx.cs' to display information on the 'Results' webpage.

LabSetup.ascx

This file contains the HTML markup that provides the entry of the experiment specification. Uses the styles in 'LabControls.css'.

LabSetup.ascx.cs

Add code to populate and update the webpage controls. The 'BuildSpecification' method takes information from the controls and places it in the XML specification node that is used by 'Setup.ascx.cs'.

Result.cs

Takes the information from the XML ExperimentResult node and places it into a 'ResultInfo' structure where it can be accessed by 'LabResults.ascx.cs'.

TypeResultInfo.cs

The 'ResultInfo' structure that contains the information taken from the XML ExperimentResult node.