**Cover Page**

A Project of 2013 Clarkson University Sustainability Fund Grants

*Project Title:* CLICS-Lab: Building the Hardware/Software Foundation to Enable the Sustainable Evolution of   
Clarkson University's Cyber-Learning Infrastructure for Campus Sustainability (CLICS)   
through EE/CS Educational Projects

*Project lead:* Daqing Hou, Associate Professor, Electrical and Computer Engineering Department, Box 5720

*Project team:* Yingying Zhang (CS), Nathan Torkaman (ME), Daniel Deloff (SE), Sean Franklin (SE), Ras Kas Williams (SE), Cory W. Schutz (EE)

*Email:* [dhou@clarkson.edu](mailto:dhou@clarkson.edu)

April 22, 2013

**a. Problem Need**

**Background** Clarkson University has made a commitment to improve the sustainability of campus operations and education. For example, a pilot-scale greenhouse building is designed to balance the need for ample lighting while minimizing heat losses [Gu 2012]. Additional system components already on campus – a biomass-solar thermal heating system and an anaerobic digester for food waste management and heat and power production – have been integrated with the greenhouse system to further reduce fossil fuel consumption (Figure 1). The anaerobic digester uses a biological process to convert over 300 kg (650 lb) of our campus’ food waste each day into biogas, which is then used to produce electric power that will be used in the system [Grimberg 2012]. The digester also provides nutrient-rich fertilizer that will be considered for use in the greenhouse to support plant growth. The energy cabin, with its solar thermal/wood pellet 22 kW heating system, provides heat from renewable resources to maintain required temperatures in both the digester and the greenhouse.

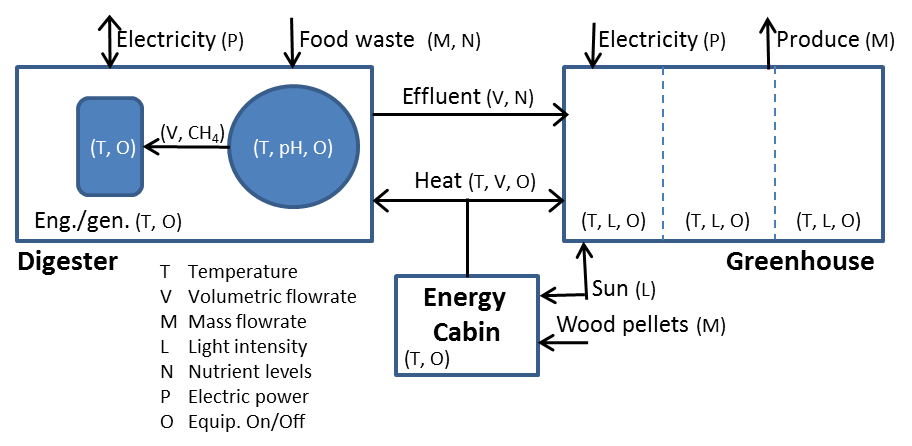
These facilities include an extensive array of sensors to monitor mass and energy flows and control heating and lighting systems to increase energy efficiency. Collectively, these systems represent an opportunity to teach our students green engineering and business practices to maximize mass and energy efficiencies as “waste” resources are shared among the facilities. Recently, NSF has awarded us a pilot project to integrate these systems with a broad range of classes to promote critical thinking skills and sustainability concepts [Powers et al. 2012]. In particular, we have proposed to develop CLICS (*Clarkson University's Cyber-Learning Infrastructure for Campus Sustainability*, Figure 2) to acquire data from, and control, these physical facilities. The PLCs (Programmable Logic Controller) periodically poll the sensors and send the acquired data to a dedicated computer server (CLICS server hereafter). The CLICS server also provides a web interface for accessing these data; it can also be used to remotely control the operations of the sensors and the PLC devices, such as turning on/off a pump or a light. The CLICS server achieves these by running the open-source Machine-to-Machine (a.k.a. industrial control, or SCADA) software called Mango M2M[[1]](#footnote-1). The CLICS design has an advantage of modularity; thus it can be relatively easy to accommodate different systems.

Figure 1: Simple schematic of Clarkson’s greenhouse, digester and energy cabin system with interconnected flows and an extensive sensor network.

Figure 2: Simple schematic of Clarkson’s greenhouse, digester and energy cabin system with interconnected flows and an extensive sensor network.

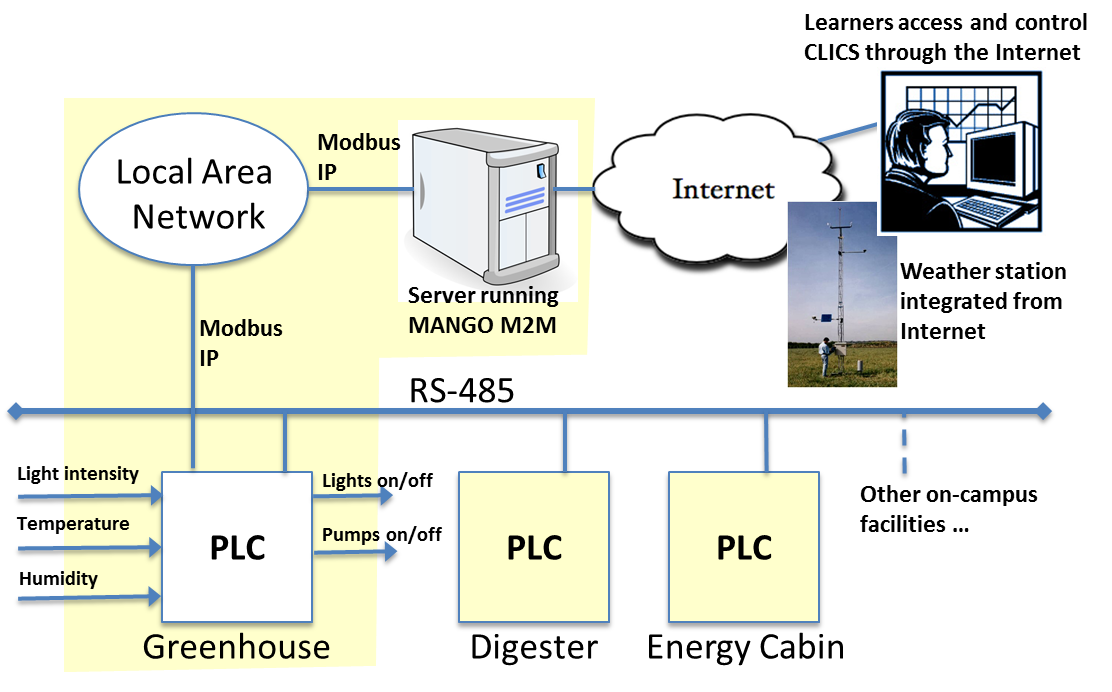


Figure 2: The CLICS network provides access to multiple sustainability projects with room for adding modules for other systems at a later date. The system that currently exists is highlighted in yellow. Integration of data from the digester and energy cabin are completed. The development of the cyber-learning environment is currently underway.

The several components depicted yellow in Figure 2 are already available and integrated, but further system integration and its extension as a cyber-learning tool are required. The greenhouse and digester are currently fully equipped with most of the sensors and PLCs that are needed for operation and analysis. Data for the greenhouse, the digester, and the energy cabin is currently archived by Mango. The integration of the food digester system and the energy cabin with Mango have been completed are part of this project. The web interface design is currently underway in my group.

**Problem Need** Field trips, inquiry through data acquisition and analysis, and communication of findings to external audiences – all of these pedagogical practices help to engage our students with the ill-posed problems they will face as STEM professionals. In this regard, Clarkson’s campus sustainability infrastructure presents a great opportunity to use our own campus communities as relevant, experiential learning sites to integrate these practices to motivate students and help them excel. The CLICS automation system, in particular, is designed to help overcome the barriers with access to information and real data from our campus operations. We have been working closely with the stakeholders of the campus sustainability projects to understand their needs in order to drive the design and implementation of CLICS.

Nonetheless, for any useful software project, there will always be demands for new features long after the initial system is delivered [Pressman 2010]. CLICS is of no exceptions. Furthermore, in future, other campus sustainability projects may need to be connected like CLICS (these may include a wind power test facility, rainwater collection and treatment for use in toilets, and a micro-turbine that generates heat and power). *Therefore, we must explore ways to sustain and evolve CLICS to satisfy these demands, even beyond the completion of a funded project, making CLICS a truly sustainable educational resource.* The goal of this proposal was to establish the necessary hardware/software foundation for training students about CLICS so that we can evolve CLICS by means of free, open-source student projects. Due to the constant demands for engaging educational projects, and the inherent “coolness” factor offered by industrial control systems such as CLICS, which extend beyond just computer programming, we have high confidence that CLICS will become a popular and successful project for the EE and CS students alike! Anecdotally, interacting with student volunteers for campus sustainability projects, we have encountered several undergraduate/graduate students expressing keen interests in sensor / control related projects but lacking the needed technical guidance and suitable project opportunities. This project was designed to meet these needs.

The objectives of this proposal are to establish the necessary *hardware/software foundation* for training students about CLICS and conduct a *pilot study* to explore how we can add new features to CLICS by means of student projects. There will be four project deliverables:

1. A working CLICS-Lab that can be used to teach PLC systems and the web-based Mango
2. Teaching modules (Topics: PLC and Ladder Logic Programming, lab exercises, Mango design)
3. A pilot student project in EE564-Enterprise Computing, Spring 2013 and in another EE/CS course
4. Final project report

**b. Project Approach**

Since April 2012, the PI has been studying both the Greenhouse control system and the food digester control system. The goal was to understand enough of the internal design of these systems in order to be able to customize these systems. In particular, to work with such systems, one needs to learn the fundamentals of PLC systems and its programming language—Ladder Logic. With fund provided by this project, we have set up a basic PLC system that can be used for demonstration and experimentation.

In addition to working with the PLC systems, new CLICS projects may involve adding or modifying existing features to extend the Mango software system. To facilitate this process, the PI and his EE564 class have developed lessons that can be used to teach Mango’s design (see attachments). The name CLICS-Lab denotes a local area network made of the PLC system and a PC server that runs the Mango system.

The PI conducted a pilot study in the context of EE564-Enterprise Computing in Spring 2013, where the PI and EE564 students defined three course projects that required customization of the Mango software (the first three in the list below). The Greenhouse, CLICS-Lab, and the developed teaching modules were used to introduce students to campus sustainability projects and CLICS. Examples of features that students can work on include, but not limited to:

* Mango reports, converting CSV data from sequential to parallel. The current Mango reporting mechanism stores multiple time series sequentially, making it difficult to find where the one time series ends and the next starts, among other challenges. We propose to format the multiple time series data parallel to each other. We leave a cell blank if a value is missing from the corresponding time series.
* Schematic/image based interfaces. A schematic/image-based interface overlays sensor information on top of a background image, which typically depicts the physical context where the sensors are deployed. We believe that such interfaces will be “natural” for both instructors and students. The coordinators of each of the campus sustainability projects will design and create their own schematics by using the Mango’s “Graphic Views” functionality.
* Weather. Integrating local weather information from the ERC weather station. Weather data can also be obtained from weatherunderground.com. This will allow for additional learning opportunities such as trend analysis, correlation studies, and predictions of system performance based on historical and predicted weather patterns. Interestingly, Mango is able to treat data on the Internet the same as sensor data; as a result, Internet data can be integrated with some programming [Weather APIs 2009].
* User management. Students and instructors will be given accounts for accessing CLICS by logging-in from a web browser, but with different levels of privilege; instructors may have the special permissions to turn on/off pumps, lights etc. We will create functionality to streamline the process of user management for a new class.
* Access log. We will modify Mango to add logging capability such that user interactions with the system can be logged, which can be used for course assessment purpose.

The PI has contacted the respective instructors for CS462/562 Web Development, EE418 Senior Design, and EE408 GUI Design. The current plan is to use CLICS as a course project for EE408 (GUI design) and EE418 (SE Senior Design) in Fall 2013. Before the end of the current semester, the PI will work with Professor Susan Conry (Instructor of EE418) to define a course project for use in the Fall semester.

**c. Impacts and Assessment**

*Table I. Student projects initiated / completed as part of this research. One MS and five undergraduate students have participated in related projects.*

|  |  |  |
| --- | --- | --- |
| **Student** | **Project** | **Status** |
| Yingying Zhang, CS MS | Digester / Mango data integration, Methane sensor data integration, Energy Cabin data integration | Implemented two solutions (1) modbus IP, and (2) Http Retrieval from the HMI. Both worked in daily operation. |
| Nathan Torkaman, ME senior | Digester PLC code modification in response to change in physical devices and layout | Completed and worked in daily operation. |
| Daniel Deloff, SE major | Greenhouse PLC control documentation | Work in progress. GH documentation is expected to finish in the first two weeks in Summer 2013. |
| Cory W. Schutz, EE MS (EE564) | Mango report (Excel data format fix—from sequential to parallel) | Design completed and reviewed with instructor. Working prototype expected by the end of the semester. |
| Sean Franklin, SE major (EE564) | ERC Weather Station Data Integration | Design completed and reviewed with instructor. Working prototype expected by the end of the semester. |
| Ras Kas Williams, SE major (EE564) | Schematic based Mango interface | Working prototype expected by the end of the semester. |

Table I summarizes the student projects initiated as part of this research and their current status. A total of six students were involved. To bring the EE564 students up to speed, a set of design notes have been developed in the process (see attachments). Lastly, we are planning to use CLICS in two more EE courses in Fall 2013. In particular, Professor Susan Conry has expressed an interest in defining a project based on CLICS for her class EE418 Senior Design. I plan to use Mango as a case study subject in EE408 GUI Design.

My assessment of this research project follows:

1. As far as the goal of providing the basis to sustain the cyber-learning infrastructure for campus sustainability CLICS is concerned, I believe that this project has been successful in showing the great potential. All participants have made contributions that are valuable to the stakeholders. The number of graduate and undergraduate students participating in related projects is appropriate to allow me to exercise the proper quality control over their individual contributions.
2. Students can make useful, technical contributions, provided that they are given enough time and guidance to work on. Students are probably not good for operational support that requires emergency response.
3. The CLICS project has provided many challenges that one would only be able to encounter in real world projects, for example, the Http retriever fix to obtain data from the slow HMI processor (March), which requires good understanding of practical networking, and the Derby database fix (April 12-15) that requires in-depth knowledge of the database. It remains to be seen how such challenges can be used to benefit education and research in the future.
4. The Mango software is a full-scale example of a modern, web-and-database-based, distributed system. As such, it incurs a relatively steep learning curve to get oneself familiar with it in order to engage in any actual customization. This implies that by working on such a project, students get industrial experience, but it also means that the scope of each course project must be limited so as to be feasible within a single semester. In the case of EE564, we have to cover the following (long) list of topics before working with the course projects, HTML / CSS / Javascript / JSP / Java / SQL / XML, Servlets, HTTP, Spring Framework, Ajax / DWR, Derby / MySQL, JDK 6, Tomcat 6, ant (latest release), Eclipse. TCP / IP, modbus IP, Ladder Logic, PLC architecture (Direct Logic and GE Fanuc), and HMI.

**d. Recommendation**

I strongly recommend Clarkson to formalize the CLICS-Lab as a multi-disciplinary effort, perhaps by providing the necessary encouragement and financial support from management, and by inviting more participation from the environmental engineering faculty and students. The involvement from environment engineering is critical to provide a use case for everything that is being worked on as motivation, and to explore new possibilities and opportunities collaboratively. The project can benefit greatly from one or two research assistants as gatekeepers for quality control.

In the long run, I expect CLICS-Lab to become a clearing house to help interested students establish meaningful projects in the areas of industrial controls and web development; efficiently guide them through the initial learning curve to become productive sooner; and serve as a reference model for other similar projects on campus. Finally, participating courses would qualify to add to the list of sustainability-related courses (http://www.clarkson.edu/green/docs/sustainability\_courses.pdf).

**e. References**

[Grimberg 2012] Stefan Grimberg. *Feasibility Study for the Anaerobic Digestion of Clarkson Food Waste.* Project report. 2012. <http://www.clarkson.edu/green/docs/NYSP2I%20Digester-Final.pdf>. Last accessed: October 20, 2012.

[Gu 2012] Zhengteng Gu. *Lettuce growing in aeroponics-based CEHRF system*. MS Thesis. Clarkson University. 2012. <http://moodle.clarkson.edu/mod/resource/view.php?id=78859>. Last accessed: October 20, 2012.

[Powers et al. 2012] Susan Powers, Jan DeWaters, Stefan Grimberg, Mary Margaret Small, Daqing Hou*. Engage and Excel: A framework for promoting 21st century skills by integrating campus sustainability initiatives and data into instructional practices.* NSF TUES Proposal. 2012. Funded.

[Pressman 2010] Roger S. Pressman. *Software Engineering: A Practitioner's Approach.* 7th Edition, McGraw Hill Publishing Company, 2010.

[Weather APIs 2009] <http://blog.programmableweb.com/2009/04/15/5-weather-apis-from-weatherbug-to-weather-channel/>. Last accessed: August 2012.

**The network**

*IP Addresses*

* GH Master PLC (ip address: 10.0.0.2) plus three slave PLCs, one in each of the three rooms (all four are of the Direct Logic 06 model)
* Mango computer is the gateway for the digester subnet, with IP: 10.0.0.1?
* WattNode power meter with IP 192.168.127.250
* Digester HMI (GE Fanuc QuickPanel Control IC574CSL, 12”, model: ES1222; ip: 192.168.1.102)
* Digester Node 1 (GE Fanuc IC200EBI001; ip: 192.168.1.105), a “rack” made of One ENIU[[2]](#footnote-2) (Ethernet Network Interface Unit, IC200EBI001) plus eight IO modules (“Slot” 1 to “Slot” 8; from left to right, model: IC200ALG264E, IC200ALG264E, IC200ALG320E, IC200MDL640F, IC200MDL640F, IC200MDL640F, IC200MDL940E, IC200MDL940E)
* Digester Node 2 (GE Fanuc IC200EBI001; 192.168.1.106), a “rack” made of One ENIU plus one IO module (Slot 1: IC200ALG264E)
* Mango computer is the gateway for the digester subnet, with IP: 192.168.1.?
* Energy Cabin ioLogik E1200[[3]](#footnote-3) 1 (192.168.127.254), flow meters
* Energy Cabin ioLogik E1200 2 (192.168.127.253), thermo couples
* Mango computer is the gateway for the Energy Cabin subnet, with IP: 192.168.127.1

*Modbus serial communication protocol*

<http://www.consultants-online.co.za/pub/itap_101/html/ch03s05.html>

<http://www.modbus.org/faq.php>

**A table that maps each variable to slot id and address offset is needed for both the digester and the greenhouse PLC.**

*Troubleshooting*

On-campus, wireless connection to Mango system

http://greenhouse.wlan.clarkson.edu:8080

admin/admin

**I have installed and configured the new wireless unit in Cheel. Here are the network details:**

In case no network connection, recover using Ralink wireless router configuration software, instead of the usual windows user interface:

**SSID: ClarksonGreenhouse**

**WPA2 Passphrase: GreenhouseDigester2013**

Network congestion (indicated by the green cycles in Mango’s watch list)

Ubiquiti Nanostation LOCO M2 Outdoor MIMO 2x2 802.11g/n

by [Ubiquiti](http://www.amazon.com/s/ref=bl_sr_pc?_encoding=UTF8&field-brandtextbin=Ubiquiti&node=172282)

<http://www.amazon.com/Ubiquiti-Nanostation-LOCO-Outdoor-802-11g/dp/B004EGI3CI/>

**Mango Installation**

Mango has been shown working successfully on Windows and Mac OS X. We have experienced problems with running it on Ubuntu, but suspect that these are not fundamental but a matter of following proper procedures.

The last open-source release is Mango 1.12.5. Unfortunately, it contains a bug in the non-open-source portion owned by serotoninsoftware.com. So we used the more reliable recent release 1.12.3.

Mango also requires the following software to run properly:

Apache Tomcat 6

Java Development Kit 1.6

*Installation Write up*

Cory Schutz 4/9/2013

The following is a description of the steps that a user needs to go through in order to install the mango system. This write up may not be based on the most efficient way or latest versions but rather shows how I personally got Mango up and running without any errors. This write up assumes that the user has apache-tomcat 6 installed. In my case, the location of my apache tomcat bin is: C:\apache-tomcat-6\bin. Make sure that the local of the apache tomcat bin is pointed to by the CATALINA\_HOME environment variable.

1. Download Mango Source Version 1.12.3 from:

[http://sourceforge.net/projects/scadabr/files/Software/mango-src/mango-source-1.12.3.zip/dowbunload](http://sourceforge.net/projects/scadabr/files/Software/mango-src/mango-source-1.12.3.zip/download). The download should start automatically. Mango versions 1.12.4 and 1.12.5 compile and deploy but create a server error on apache when the site is actually running. For this reason we will use Mango version 1.12.3 which does not cause this error.

The top-level directories of Mango source code appear as follows:

Daqings-MacBook:mango-source-1.12.3 dhou$ ls

build db src war

build.properties derby.log src\_gen

build.xml lib templates

1. Compiling with Java 7 causes Mango to through errors because of the private access methods. To easily solve this issue, we use Java 6 instead, which compiles the 1.12.3 Mango source code provided without any errors. I therefore went to oracle’s download page at: <http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html> for Windows/Linux, and <https://support.apple.com/kb/DL1572?viewlocale=en_US&locale=en_US> for Mac. From here I clicked “Previous Releases” and then chose Java SE 6. I scrolled down to the Java SE Development Kit 6u41 and downloaded: [jdk-6u41-windows-i586.exe](javascript:%20void(0)) because I am using an x86 machine. Run the installer and install to Program Files (x86\_ directory). This will create a java folder with subdirectories: jdk1.6.0\_41 and a jre6. Add the following environment variables:

PATH: C:\Program Files (x86)\Java\jdk1.6.0\_41\bin

JAVA\_HOME: C:\Program Files (x86)\Java\jdk1.6.0\_41

JRE\_HOME: C:\Program Files (x86)\Java\jre6

1. Download Ant 1.8.4 from the following site:

<http://archive.apache.org/dist/ant/binaries/>[apache-ant-1.8.4-bin.zip](http://archive.apache.org/dist/ant/binaries/apache-ant-1.8.4-bin.zip)

I installed my version directly under the C: drive at C:\apache-ant-1.8.4\bin

Once installed, add this bin path to the PATH environment variable. Next, set the ANT\_HOME environment variable to the directory where you installed Ant.

1. In the folder that the Mango 1.12.3 source code downloaded in there will be a build.properties file. Open this file and change the first line “tomcat.home=” so that it points to your apache-tomcat folder: C:/apache-tomcat-6. Also, I changed the following lines: “tomcat.appdir” and “tomcat.apppath” so that when I deployed my Mango install it wouldn’t be called test. You may name them whatever you’d like (I used myMango and /myMango).
2. Also in the folder that the Mango 1.12.3 source code downloaded in there will be a build.xml file which contains build information for the Mango system. In the first lines, there are XML property tags that point to the following directories: src, src\_gen, war, templates, build, and /WEB-INF/classes. All of these folders exist except for: bin, build, lib, src\_gen, which must be created manually. Nothing needs to be placed in the bin, build, src\_gen folder, just create it and leave it empty. Create lib folder manually under mango-souce-1.12.3 copy all the \*.jar files from lib folder under war/WEB-INF into it. Copy the jsp-api.jar servlet-api.jar from the lib under tomcat to the newly create lib folder.
3. Using the command prompt, navigate to the mango source directory where the file build.xml is located. Type “ant compile” which will compile the mango source code. This step is to make sure that the source code compiles correctly but is not entirely necessary because the next steps also compile the code. From here there are two different options to deploy the compiled mango source code to the apache-tomcat directory. One is “ant deploy” and the other is “ant fullDeploy”. The deploy option compiles mango and copies all the static assets to the build directory. The fullDeploy option does the same thing but it also execute the option createConfigFiles that creates a mango log for viewing errors.
4. Once you have successfully deployed, you can open a web browser and use <http://localhost:8080/mango-name> to open up mango (mango-name represents the value you put for tomcat.appdir in step 4 which is test by default).
5. Lastly, login to Mango. On the download page for mango it notes that, “Upon installation, Mango creates a single login account with username "admin" and password "admin". Once you log in, you are strongly encouraged to change at least the password for this account on the "Users" page (). Also, you can set various system properties on the "System settings" page ().”

**Derby database and recovery process**

run ij tool to access derby db:

*cd the\_directory where your derby db is located.*

*java -cp /Users/dhou/work/apache-tomcat-6.0.10/webapps/test/WEB-INF/lib/derbytools.jar:/Users/dhou/work/apache-tomcat-6.0.10/webapps/test/WEB-INF/lib/derby.jar org.apache.derby.tools.ij*

*-- load the Derby driver so that a connection -- can be made:*

*ij> driver 'org.apache.derby.jdbc.EmbeddedDriver';*

*ij> connect 'jdbc:derby:mangoDB-04122013;create=false';*

*ij>*

The driver class org.apache.derby.jdbc.EmbeddedDriver is in derby.jar, and the ij tool is in derbytools.jar; that is why we need to put these two jars on our class path.

Once you logged into ij, you can issue various SQL commands; for example, using "show tables;" to see all database tables.

See more documentation about derby tool ij and derby here:

[*http://db.apache.org/derby/docs/10.0/manuals/tools/tools09.html#HDRSII-TOOLS-37363*](http://db.apache.org/derby/docs/10.0/manuals/tools/tools09.html#HDRSII-TOOLS-37363)

The detailed process for recovering derby db can be found under */Users/dhou/work/apache-tomcat-6.0.10/bin*, particularly the SQL script file *derby-recovery.sql*.

**Example exporting all data from a table to a single export file**

The following example shows how to export data from the STAFF table in a sample database to the file myfile.del.

CALL SYSCS\_UTIL.SYSCS\_EXPORT\_TABLE(null,'STAFF','myfile.del',null,null,null);

import:

CALL SYSCS\_UTIL.SYSCS\_IMPORT\_TABLE(null,'POINTVALUES','pointvalues.dat',null,null,null,1);

I have saved a copy of the derby db from the greenhouse as of April 12, 2013.

The *database schema for Mango* can be found /Users/dhou/work/apache-tomcat-6.0.10/mango-source-1.12.3/db/createTables-derby.sql:

--

-- System settings

create table systemSettings (

settingName varchar(32) not null,

settingValue clob

);

alter table systemSettings add constraint systemSettingsPk primary key (settingName);

--

-- Users

create table users (

id int not null generated by default as identity (start with 1, increment by 1),

username varchar(40) not null,

password varchar(30) not null,

email varchar(255) not null,

phone varchar(40),

admin char(1) not null,

disabled char(1) not null,

lastLogin bigint,

selectedWatchList int,

homeUrl varchar(255),

receiveAlarmEmails int not null,

receiveOwnAuditEvents char(1) not null

);

alter table users add constraint usersPk primary key (id);

create table userComments (

userId int,

commentType int not null,

typeKey int not null,

ts bigint not null,

commentText varchar(1024) not null

);

alter table userComments add constraint userCommentsFk1 foreign key (userId) references users(id);

--

-- Mailing lists

create table mailingLists (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

name varchar(40) not null

);

alter table mailingLists add constraint mailingListsPk primary key (id);

alter table mailingLists add constraint mailingListsUn1 unique (xid);

create table mailingListInactive (

mailingListId int not null,

inactiveInterval int not null

);

alter table mailingListInactive add constraint mailingListInactiveFk1 foreign key (mailingListId)

references mailingLists(id) on delete cascade;

create table mailingListMembers (

mailingListId int not null,

typeId int not null,

userId int,

address varchar(255)

);

alter table mailingListMembers add constraint mailingListMembersFk1 foreign key (mailingListId)

references mailingLists(id) on delete cascade;

--

--

-- Data Sources

--

create table dataSources (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

name varchar(40) not null,

dataSourceType int not null,

data blob not null

);

alter table dataSources add constraint dataSourcesPk primary key (id);

alter table dataSources add constraint dataSourcesUn1 unique (xid);

-- Data source permissions

create table dataSourceUsers (

dataSourceId int not null,

userId int not null

);

alter table dataSourceUsers add constraint dataSourceUsersFk1 foreign key (dataSourceId) references dataSources(id);

alter table dataSourceUsers add constraint dataSourceUsersFk2 foreign key (userId) references users(id) on delete cascade;

--

--

-- Data Points

--

create table dataPoints (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

dataSourceId int not null,

data blob not null

);

alter table dataPoints add constraint dataPointsPk primary key (id);

alter table dataPoints add constraint dataPointsUn1 unique (xid);

alter table dataPoints add constraint dataPointsFk1 foreign key (dataSourceId) references dataSources(id);

-- Data point permissions

create table dataPointUsers (

dataPointId int not null,

userId int not null,

permission int not null

);

alter table dataPointUsers add constraint dataPointUsersFk1 foreign key (dataPointId) references dataPoints(id);

alter table dataPointUsers add constraint dataPointUsersFk2 foreign key (userId) references users(id) on delete cascade;

--

--

-- Views

--

create table mangoViews (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

name varchar(100) not null,

background varchar(255),

userId int not null,

anonymousAccess int not null,

data blob not null

);

alter table mangoViews add constraint mangoViewsPk primary key (id);

alter table mangoViews add constraint mangoViewsUn1 unique (xid);

alter table mangoViews add constraint mangoViewsFk1 foreign key (userId) references users(id) on delete cascade;

create table mangoViewUsers (

mangoViewId int not null,

userId int not null,

accessType int not null

);

alter table mangoViewUsers add constraint mangoViewUsersPk primary key (mangoViewId, userId);

alter table mangoViewUsers add constraint mangoViewUsersFk1 foreign key (mangoViewId) references mangoViews(id);

alter table mangoViewUsers add constraint mangoViewUsersFk2 foreign key (userId) references users(id) on delete cascade;

--

--

-- Point Values (historical data)

--

create table pointValues (

id bigint not null generated by default as identity (start with 1, increment by 1),

dataPointId int not null,

dataType int not null,

pointValue double,

ts bigint not null

);

alter table pointValues add constraint pointValuesPk primary key (id);

alter table pointValues add constraint pointValuesFk1 foreign key (dataPointId) references dataPoints(id) on delete cascade;

create index pointValuesIdx1 on pointValues (ts, dataPointId);

create index pointValuesIdx2 on pointValues (dataPointId, ts);

create table pointValueAnnotations (

pointValueId bigint not null,

textPointValueShort varchar(128),

textPointValueLong clob,

sourceType smallint,

sourceId int

);

alter table pointValueAnnotations add constraint pointValueAnnotationsFk1 foreign key (pointValueId)

references pointValues(id) on delete cascade;

--

--

-- Watch list

--

create table watchLists (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

userId int not null,

name varchar(50)

);

alter table watchLists add constraint watchListsPk primary key (id);

alter table watchLists add constraint watchListsUn1 unique (xid);

alter table watchLists add constraint watchListsFk1 foreign key (userId) references users(id) on delete cascade;

create table watchListPoints (

watchListId int not null,

dataPointId int not null,

sortOrder int not null

);

alter table watchListPoints add constraint watchListPointsFk1 foreign key (watchListId) references watchLists(id) on delete cascade;

alter table watchListPoints add constraint watchListPointsFk2 foreign key (dataPointId) references dataPoints(id);

create table watchListUsers (

watchListId int not null,

userId int not null,

accessType int not null

);

alter table watchListUsers add constraint watchListUsersPk primary key (watchListId, userId);

alter table watchListUsers add constraint watchListUsersFk1 foreign key (watchListId) references watchLists(id);

alter table watchListUsers add constraint watchListUsersFk2 foreign key (userId) references users(id) on delete cascade;

--

--

-- Point event detectors

--

create table pointEventDetectors (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

alias varchar(255),

dataPointId int not null,

detectorType int not null,

alarmLevel int not null,

stateLimit double,

duration int,

durationType int,

binaryState char(1),

multistateState int,

changeCount int,

alphanumericState varchar(128),

weight double

);

alter table pointEventDetectors add constraint pointEventDetectorsPk primary key (id);

alter table pointEventDetectors add constraint pointEventDetectorsUn1 unique (xid, dataPointId);

alter table pointEventDetectors add constraint pointEventDetectorsFk1 foreign key (dataPointId)

references dataPoints(id);

--

--

-- Events

--

create table events (

id int not null generated by default as identity (start with 1, increment by 1),

typeId int not null,

typeRef1 int not null,

typeRef2 int not null,

activeTs bigint not null,

rtnApplicable char(1) not null,

rtnTs bigint,

rtnCause int,

alarmLevel int not null,

message clob,

ackTs bigint,

ackUserId int,

alternateAckSource int

);

alter table events add constraint eventsPk primary key (id);

alter table events add constraint eventsFk1 foreign key (ackUserId) references users(id);

create table userEvents (

eventId int not null,

userId int not null,

silenced char(1) not null

);

alter table userEvents add constraint userEventsPk primary key (eventId, userId);

alter table userEvents add constraint userEventsFk1 foreign key (eventId) references events(id) on delete cascade;

alter table userEvents add constraint userEventsFk2 foreign key (userId) references users(id);

-- Event handlers

--

create table eventHandlers (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

alias varchar(255),

-- Event type, see events

eventTypeId int not null,

eventTypeRef1 int not null,

eventTypeRef2 int not null,

data blob not null

);

alter table eventHandlers add constraint eventHandlersPk primary key (id);

alter table eventHandlers add constraint eventHandlersUn1 unique (xid);

--

-- Scheduled events

--

create table scheduledEvents (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

alias varchar(255),

alarmLevel int not null,

scheduleType int not null,

returnToNormal char(1) not null,

disabled char(1) not null,

activeYear int,

activeMonth int,

activeDay int,

activeHour int,

activeMinute int,

activeSecond int,

activeCron varchar(25),

inactiveYear int,

inactiveMonth int,

inactiveDay int,

inactiveHour int,

inactiveMinute int,

inactiveSecond int,

inactiveCron varchar(25)

);

alter table scheduledEvents add constraint scheduledEventsPk primary key (id);

alter table scheduledEvents add constraint scheduledEventsUn1 unique (xid);

--

-- Point Hierarchy

--

create table pointHierarchy (

id int not null generated by default as identity (start with 1, increment by 1),

parentId int,

name varchar(100)

);

alter table pointHierarchy add constraint pointHierarchyPk primary key (id);

--

--

-- Compound events detectors

--

create table compoundEventDetectors (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

name varchar(100),

alarmLevel int not null,

returnToNormal char(1) not null,

disabled char(1) not null,

conditionText varchar(256) not null

);

alter table compoundEventDetectors add constraint compoundEventDetectorsPk primary key (id);

alter table compoundEventDetectors add constraint compoundEventDetectorsUn1 unique (xid);

--

--

-- Reports

--

create table reports (

id int not null generated by default as identity (start with 1, increment by 1),

userId int not null,

name varchar(100) not null,

data blob not null

);

alter table reports add constraint reportsPk primary key (id);

alter table reports add constraint reportsFk1 foreign key (userId) references users(id) on delete cascade;

create table reportInstances (

id int not null generated by default as identity (start with 1, increment by 1),

userId int not null,

name varchar(100) not null,

includeEvents int not null,

includeUserComments char(1) not null,

reportStartTime bigint not null,

reportEndTime bigint not null,

runStartTime bigint,

runEndTime bigint,

recordCount int,

preventPurge char(1)

);

alter table reportInstances add constraint reportInstancesPk primary key (id);

alter table reportInstances add constraint reportInstancesFk1 foreign key (userId) references users(id) on delete cascade;

create table reportInstancePoints (

id int not null generated by default as identity (start with 1, increment by 1),

reportInstanceId int not null,

dataSourceName varchar(40) not null,

pointName varchar(100) not null,

dataType int not null,

startValue varchar(4096),

textRenderer blob,

colour varchar(6),

consolidatedChart char(1)

);

alter table reportInstancePoints add constraint reportInstancePointsPk primary key (id);

alter table reportInstancePoints add constraint reportInstancePointsFk1 foreign key (reportInstanceId)

references reportInstances(id) on delete cascade;

create table reportInstanceData (

pointValueId bigint not null,

reportInstancePointId int not null,

pointValue double,

ts bigint not null

);

alter table reportInstanceData add constraint reportInstanceDataPk primary key (pointValueId, reportInstancePointId);

alter table reportInstanceData add constraint reportInstanceDataFk1 foreign key (reportInstancePointId)

references reportInstancePoints(id) on delete cascade;

create table reportInstanceDataAnnotations (

pointValueId bigint not null,

reportInstancePointId int not null,

textPointValueShort varchar(128),

textPointValueLong clob,

sourceValue varchar(128)

);

alter table reportInstanceDataAnnotations add constraint reportInstanceDataAnnotationsPk

primary key (pointValueId, reportInstancePointId);

alter table reportInstanceDataAnnotations add constraint reportInstanceDataAnnotationsFk1

foreign key (pointValueId, reportInstancePointId) references reportInstanceData(pointValueId, reportInstancePointId)

on delete cascade;

create table reportInstanceEvents (

eventId int not null,

reportInstanceId int not null,

typeId int not null,

typeRef1 int not null,

typeRef2 int not null,

activeTs bigint not null,

rtnApplicable char(1) not null,

rtnTs bigint,

rtnCause int,

alarmLevel int not null,

message clob,

ackTs bigint,

ackUsername varchar(40),

alternateAckSource int

);

alter table reportInstanceEvents add constraint reportInstanceEventsPk primary key (eventId, reportInstanceId);

alter table reportInstanceEvents add constraint reportInstanceEventsFk1 foreign key (reportInstanceId)

references reportInstances(id) on delete cascade; create table reportInstanceUserComments (

reportInstanceId int not null,

username varchar(40),

commentType int not null,

typeKey int not null,

ts bigint not null,

commentText varchar(1024) not null

);

alter table reportInstanceUserComments add constraint reportInstanceUserCommentsFk1 foreign key (reportInstanceId)

references reportInstances(id) on delete cascade;

--

-- Publishers

--

create table publishers (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

data blob not null

);

alter table publishers add constraint publishersPk primary key (id);

alter table publishers add constraint publishersUn1 unique (xid);

--

-- Point links

--

create table pointLinks (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

sourcePointId int not null,

targetPointId int not null,

script clob,

eventType int not null,

disabled char(1) not null

);

alter table pointLinks add constraint pointLinksPk primary key (id);

alter table pointLinks add constraint pointLinksUn1 unique (xid);

--

--

-- Maintenance events

--

create table maintenanceEvents (

id int not null generated by default as identity (start with 1, increment by 1),

xid varchar(50) not null,

dataSourceId int not null,

alias varchar(255),

alarmLevel int not null,

scheduleType int not null,

disabled char(1) not null,

activeYear int,

activeMonth int,

activeDay int,

activeHour int,

activeMinute int,

activeSecond int,

activeCron varchar(25),

inactiveYear int,

inactiveMonth int,

inactiveDay int,

inactiveHour int,

inactiveMinute int,

inactiveSecond int,

inactiveCron varchar(25)

);

alter table maintenanceEvents add constraint maintenanceEventsPk primary key (id);

alter table maintenanceEvents add constraint maintenanceEventsUn1 unique (xid);

alter table maintenanceEvents add constraint maintenanceEventsFk1 foreign key (dataSourceId) references dataSources(id);

**The Mango software**

**Notes on how Mango starts up**

Daqing Hou 4/10

*tomcat\_home* represents the directory where Tomcat is installed.

*webapps* is a default directory under *tomcat\_home* where web applications are stored.

*mango* is the root directory for the mango application.

*tomcat\_home/webapps/mango/WEB-INF/web.xml*

**308   <welcome-file-list>**

**309     <welcome-file>index.jsp</welcome-file>**

**310   </welcome-file-list>**

311

312   <error-page>

313     <exception-type>com.serotonin.mango.vo.permission.PermissionException</exception-type>

314     <location>/exception/accessDenied.jsp</location>

315   </error-page>

...

322   <error-page>

323     <error-code>404</error-code>

324     <location>/exception/**404.jsp**</location>

325   </error-page>

*tomcat\_home/webapps/mango/index.jsp*

19 <html>

 20 <head>

 21   <title>Mango M2M, by Serotonin Software</title>

 22   <meta http-equiv="content-type" content="application/xhtml+xml;charset=utf-8"/>

 23   <meta http-equiv="Content-Style-Type" content="text/css" />

 24   <meta name="Copyright" content="&copy;2006-2011 Serotonin Software Technologies Inc."/>

 25   <meta name="DESCRIPTION" content="Mango M2M Serotonin Software"/>

 26   <meta name="KEYWORDS" content="Mango M2M Serotonin Software"/>

 27   <link href="resources/common.css" type="text/css" rel="stylesheet"/>

 28   <link rel="icon" href="images/favicon.ico"/>

 29   <link rel="shortcut icon" href="images/favicon.ico"/>

 30 </head>

 31 <body>

 32 **<script language="JavaScript">window.location="login.htm";</script>**

 33 <a style="font-size: 13px; color: #804000; font-family: Verdana, Arial, Helvetica, sans-serif;" href="login.htm">Mango M2M, by Serotonin Software</a>  **<--------- this line has no effect!**

 34 </body>

 35 </html>

*tomcat\_home/webapps/mango/WEB-INF/web.xml*

231   <!--

232     Servlet mappings.

233    -->

234   <servlet-mapping>

235     <servlet-name>**springDispatcher**</servlet-name>

236     <url-pattern>**\*.htm**</url-pattern>

237   </servlet-mapping>

158   <!--

159     Servlet definitions.

160    -->

161   <servlet>

162     <servlet-name>**springDispatcher**</servlet-name>

163     <servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>

164     <load-on-startup>1</load-on-startup>

165   </servlet>

my.htm

my.html

<http://localhost:8080/mango/my.htm> will show, but <http://localhost:8080/mango/my.html> will not.

This is because the servlet mapping from \*.htm to springDispatcher.

Mango maps \*.htm to \*.jsp page. But  my.jsp does not exist.

*tomcat\_home/webapps/mango/exception/404.jsp*

<%@ include file="/WEB-INF/jsp/include/tech.jsp" %>

<br/>

<span class="bigTitle">Page Not Found!</span><br/>

<br/>

The page that you requested was not found. Please check your URL

and try your request again, or start again from the <a href="${pageContext.request.contextPath}/login.htm">login page</a>.<br/>

<br/>

<br/>

*tomcat\_home/webapps/mango/WEB-INF/web.xml*

118   <filter-mapping>

119     <filter-name>IsLoggedIn</filter-name>

120     <url-pattern>\*.shtm</url-pattern>

121   </filter-mapping>

*tomcat\_home/webapps/mango/WEB-INF/web.xml*

 81   <filter>

 82     <filter-name>IsLoggedIn</filter-name>

 83     <filter-class>**com.serotonin.mango.web.filter.NormalLoggedInFilter**</filter-class>

 84     <init-param>

 85       <param-name>forwardUrl</param-name>

 86       <param-value>/login.htm</param-value>

 87     </init-param>

 88   </filter>

...

118   <filter-mapping>

119     <filter-name>IsLoggedIn</filter-name>

120     <url-pattern>\*.shtm</url-pattern>

121   </filter-mapping>

122

LoggedInFilter is the superclass of NormalLoggedInFilter.

*src/com/serotonin/mango/web/filter/LoggedInFilter.java*

48     public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain)

 49             throws IOException, ServletException {

 50         // Assume an http request.

 51         HttpServletRequest request = (HttpServletRequest) servletRequest;

 52         HttpServletResponse response = (HttpServletResponse) servletResponse;

 53

 54         boolean loggedIn = true;

 56         User user = Common.getUser(request);

 57         if (!checkAccess(user))

 58             loggedIn = false;

...

 65

 66         if (!loggedIn) {

**69             response.sendRedirect(request.getContextPath() + forwardUrl); <----- redirected to login.htm**

 70             //request.getRequestDispatcher(forwardUrl).forward(request, response);

 71             return;

 72         }

*WEB-INF/springDispatcher-servlet.xml*

 38   <!--

 39       URL mappings to controllers

 40   -->

 41   <bean id="publicUrlMappings" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">

 42     <property name="mappings">

 43       <props>

 44         <!-- All user URLs -->

 45         <prop key="/compound\_events.shtm">compoundEventsController</prop>

 46         <prop key="/data\_point\_details.shtm">dataPointDetailsController</prop>

 47         <prop key="/data\_point\_edit.shtm">dataPointEditController</prop>

 48         <prop key="/data\_source\_edit.shtm">dataSourceEditController</prop>

 49         <prop key="/data\_sources.shtm">dataSourceListController</prop>

 50         <prop key="/emport.shtm">emportController</prop>

 51         <prop key="/event\_handlers.shtm">eventHandlersController</prop>

 52         <prop key="/events.shtm">eventsController</prop>

 53         <prop key="/help.shtm">helpController</prop>

 54         **<prop key="/login.htm">loginController</prop>**

 55         **<prop key="/logout.htm">logoutController</prop>**

 56         <prop key="/mailing\_lists.shtm">mailingListsController</prop>

 57         <prop key="/maintenance\_events.shtm">maintenanceEventsController</prop>

 58         <prop key="/point\_hierarchy.shtm">pointHierarchyController</prop>

 59         <prop key="/point\_links.shtm">pointLinksController</prop>

 60         **<prop key="/public\_view.htm">publicViewController</prop>**

 61         <prop key="/publisher\_edit.shtm">publisherEditController</prop>

 62         <prop key="/publishers.shtm">publisherListController</prop>

 63         <prop key="/reports.shtm">reportsController</prop>

 64         <prop key="/reportChart.shtm">reportChartController</prop>

 65         <prop key="/scheduled\_events.shtm">scheduledEventsController</prop>

 66         <prop key="/sql.shtm">sqlController</prop>

 67         <prop key="/system\_settings.shtm">systemSettingsController</prop>

 68         <prop key="/users.shtm">usersController</prop>

 69         <prop key="/views.shtm">viewsController</prop>

 70         <prop key="/view\_edit.shtm">viewEditController</prop>

 71         <prop key="/watch\_list.shtm">watchListController</prop>

 72         **<prop key="/webcam\_live\_feed.htm">webcamLiveFeedController</prop>**

 73

 74         <!-- Mobile user URLs -->

 75         <prop key="/mobile\_login.htm">mobileLoginController</prop>

 76         <prop key="/mobile\_logout.htm">mobileLogoutController</prop>

 77         <prop key="/mobile\_watch\_list.shtm">mobileWatchListController</prop>

 78       </props>

*WEB-INF/springDispatcher-servlet.xml*

 28   <!-- View resolver -->

 29   <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">

 30     <property name="viewClass"><value>org.springframework.web.servlet.view.JstlView</value></property>

 31     <property name="prefix"><value>/WEB-INF/jsp/</value></property>

 32     <property name="suffix"><value>.jsp</value></property>

 33   </bean>

two kinds of view

write: the view produces new content

*WEB-INF/springDispatcher-servlet.xml*

134   <bean id="loginController" class="com.serotonin.mango.web.mvc.controller.LoginController">

135     <property name="commandName"><value>login</value></property>

136     <property name="commandClass"><value>com.serotonin.mango.web.mvc.form.LoginForm</value></property>

137     <property name="formView"><value>login</value></property>  **<------- login.jsp**

138     <property name="successUrl"><value>watch\_list.shtm</value></property>

139     <property name="newUserUrl"><value>help.shtm</value></property>

140     <property name="bindOnNewForm"><value>true</value></property>

141   </bean>

read-only: the view does not produce new content;  only displays existing content.

228   <bean id="watchListController" class="com.serotonin.mango.web.mvc.controller.WatchListController">

229     <property name="viewName"><value>watchList</value></property>   **<------watchList.jsp**

230   </bean>

Cory Schutz

4/10/2013

**Export Data**

The following is an explanation of how the export data functionality works for the reporting page. The purpose of this write up is to provide an understanding of how reports are exported to Excel files so that the functionality can be changed. Specifically, I would like to change how the data is exported so that if there are multiple points in the report that they are exported side by side rather than one after another.

The following assumes that the user starts on the reports page (after logging in, simply click on the Reports button in the menu bar at the top of the page). This page that is being displayed is the reports.jsp view. On this page there is a Report Queue that shows information about each generated report. The first button in the Report Queue box which is a down arrow is the export data button. On line 255 of reports.jsp, the “onclick” of the button triggers the exportData function with the report instance’s name and id as parameters. The exportData, located on line 300 of reports.jsp changes the window location of the DOM to “export/reportname.csv?instanceId=reportID” where reportname and reportID are the name and ID of the report instance.

The TomCat container uses the servlet-name and servlet-mapping parameters to map the url-pattern “/export/\*” to the servlet ReportExportServlet. This occurs on lines 195 and 265 of web.xml. The servlet container(TomCat), uses the URL and understands that this represents a GET request. This is because only GET requests have parameters in the URL (in this case the ?instanceId=reportID) while POST puts these parameters inside the body of the request.

ReportExportServlet extends the ReportExportBase class which extends Httpservlet. All the ReportExportServlet class does is override the doGet method of Httpservlet and changes the serialization ID of the servlet. The overridden doGet method calls execute(), the only method in the ReportExportBase class.

The execute method first retrieves the reportID parameter from the GET request and then creates a new report data access object (ReportDao). The new ReportDao’s method getReportInstance(id) is used to retrieve the report instance that should be exported from the database table reportInstances (line 126 of ReportDao). This instance is used to ensure that the user that sent to request has permissions to access the report.

The next step in the ReportExportBase is that the response context is test to csv and the ResourceBundle (related to the language translations) to retrieved from Common.ensureI18n. In interesting side note that I learn is that I18n stands for internationalization where the ‘I’ is the first letter, ‘N’ is the last letter, and there are 18 letters between the two.

In ReportExportBase, a new ReportCsvStreamer is created. This uses the responses Printwriter to output the five column titles: Point Name, Time, Value, Rendered, Annotation. Then reportInstanceData() is called from the ReportDao with the reportID and the previously mentioned ReportCsvStreamer as parameters.

This is where I am currently stuck at. Within the ReportDao class the reportInstanceData() method is currently confusing me. Specifically the call to query(). I thought that query was a method from the ExtendedJdbcTemplate class (reference: <http://static.springsource.org/spring/docs/2.5.x/api/org/springframework/jdbc/core/JdbcTemplate.html>). What confuses me, however, is that it looks like there is a method being defined as a parameter and that there is no comma after the third parameter: RowMapper. This is where I am at currently.

The following is an original attempt to find out how the derby database is originally created, but in fact how to convert the existing db to another kind. The sql scripts to create the all the tables in the db are at WEB-INF/db/createTables-derby.sql:

The ReportDao extends the BaseDao class so when a new ReportDao is created the constructor for BaseDao is called. This constructor calls getDatabaseAccess() method from the static context wrapper in ctx. The getDatabaseAccess() method within ContextWrapper.java tries to access the attribute DATABASE\_ACCESS on line 52 from the ServletContext. So, somewhere before the attribute DATABASE\_ACCESS must be set. The servletContext is maintained by TomCat, and after searching around the DATABASE\_ACCESS is set within the MangoContextListener class. A context listener receives notification that the web application initialization process is starting so that it can execute initialization code. The contextListeners are defined on lines 298 and 299 of web.xml (with the important one in this case being MangoContextListener). The MangoContextListener calls databaseInitialize on line 252 which calls the following methods in the DatabaseAccess class: createDatabaseAccess and databaseInitialize. This is where the “db.type=derby” is extracted from the env.properties file and the database is originally created.

Mango M2M WatchList Workflow

Ras Kas Williams, 4/9/2013

When a user successfully login to the Mango system, they are shown the  
default view into the system. This is a WatchList. In springDispatcher­servlet.xml,a login Controller bean is defined. As you may have inferred, this controller has a class associated with it (com.serotonin.mango.web.mvc.controller.LoginController) that controls the user login workflow. It has a property named successUrlthat defines what URL the user will be directed upon a successful login. This property’s value is watch\_list.shtm. So.....

1. Upon successful login to the Mango system, the user is redirected to /watch\_list.shtm. In springDispatcher­servlet.xml, watchListControlleris mapped to this URL.
2. The com.serotonin.mango.web.mvc.controller.WatchListControlle rclass is instantiated and it’s handleRequestInternalmethod is called by springDispatcher­servlet.xml. The bean for watchListController has the WatchListControllerclass associated with it in the springDispatcher­servlet.xmlconfig file.
3. All handleRequestInternal does is return a new instance of ModelAndView that has the viewName(which is passed in as a bean property) and the return value of createModel as parameters.
4. createModel is a method of WatchListController that has an HTTP request object as a parameter. It is the workhorse of this class. It does the following:
   1. It obtains the user associated with the request.
   2. Obtains each watchlist associated with that user. If the user

doesn’t have a watchlist defined, a default one is created.

* 1. Obtains the data points for each watchlist.
  2. All of this data is saved in a new instance of Model which is just a

HashMap<String,Object>and returned.

1. The watchList.jsp file associated with the watchListController

bean (Spring MVC convention) is then instantiated into HTML at runtime and shown to the user. The Spring MVC framework knows that watchList.jsp should be used as a view because it is passed as the

viewName property to the WatchListController. This is defined in the springDispatcher­servlet.xml.

Mango Login Workflow

Ras Kas Williams, 3/28/2013

When users access the Mango system, the their main point of access is usually to the root URL, http://greenhouse.wlan.clarkson.edu:8080. Assuming this is the normal user behavior, the following happens:

1. Going to the root URL will load the index.jsp file at the root directory of all J2EE projects.
2. Upon loading, the JavaScript code within the index.jsp file redirects the user to /login.htm[[4]](#footnote-4). /login.htm is a publicly accessible URL for Mango. All publicly accessible URLs for Mango are defined in springDispatcher­servlet.xml (Mango is built on top of the Spring Framework).
3. There are 3 filters, defined in web.xml, that are triggered upon any HTTP request to any of the publicly accessible URLS for Mango. The filter we are interested in is IsLoggedIn. Asitsnamesuggests,itcheckstoseeiftheuserhasloggedinornot. At this point the user isn’t logged in, so necessary flags are set internally to reflect that. The HTTP request continues to its destination of /login.htm.
4. Since all publicly accessible URLs for Mango are mapped to a controller in springDispatcher­servlet.xml, going to the /login.htm executes LoginController.javawhich generates the necessary login form.
5. Upon successful login, the user is redirected to /watch\_list.shtm, which calls WatchListController.java(of course, after the filters execute).
6. WatchListController.javagenerates the main user interface for the Mango system, which is as its name suggests, a watch list of all registered sensors in the system.

Spring Workflow

The Spring framework is an inversion of control framework to help decouple your project components’ dependencies. The framework is currently at version 3.x. The Spring framework uses dependency injection (DI) design pattern to manage object dependencies. TheSpring MVC framework is a Java MVC framework that builds on top of SpringCore. Three of the core components are HandlerMapping, Controllers, and View Resolvers. Handler Mappings define how web request’s (URLs) map to Controllers. Controllers handle web requests (very similar to servlets). View Resolvers resolve the “view name” returned by from a controller class to a physical view page or JSP.

DWR is a Java library that enables Java on the server and JavaScript in a browser to interact and call each other as simply as possible.

page 137, ServletDispatcher

<http://www.tutorialspoint.com/spring/spring_tutorial.pdf>

vi WEB-INF/web.xml

vi WEB-INF/applicationContext.xml

vi war/WEB-INF/springDispatcher-servlet.xml

vi WEB-INF/jsp/login.jsp

org.springframework.web.servlet.view.InternalResourceViewResolver

overview of dwr

<http://directwebremoting.org/dwr/introduction/index.html>

common javascript code that enables common features such as sound, can be found in common.js file.

Cory Schutz

4/20/13

Project update/plans

I have an initial implementation idea planned out, but I have yet to start implementing it. I am not sure if in it the most computationally effective way of exporting in the new format, but I am going to try to implement my initial plan and then possibly try other implementations to compare run times. My goal is to have code to show by Tuesday.

As for my implementation plan, I will modify the use of the query method of the jbdcTemplate class that is used within the reportInstanceData method of ReportDao.java. The query method takes a RowCallbackHandler as a parameter which has a processRow method that is overridden. This processRow method is overridden which executes on each record in the query result. The query is currently executed on each data point in the report and extracts the query record’s information into a reportDataValue. The problem is that the reportDataValue is immediately sent to a ReportCsvStreamer which writes the record to Excel. My implementation will keep much of this same code but instead of sending the reportDataValue to the ReportCsvStreamer right away, the reportDataValue will be sent to a priority queue. This means that after the queries, all the records that correspond to the report’s data points will be stored in one priority queue.

Since the priority queue will contain ReportDataValues so a comparator must be written to order the ReportDataValues by time. This should be pretty easy since the date/time is stored as a long type until it is written to the csv file where it is converted to a date-time format. The comparator, given two ReportDataValues, will simply return the smaller of the two long values that represent the time.

Lastly, once the priority queue is set up I can remove them one at a time. I will have to check the time of the currently removed ReportDataValue and then peek at the next ReportDataValue in the priority queue to see if they are the same. If the times are the same they need to go on the same row in Excel and I will continue to peek at the next ReportDataValue until the time is different. The majority of the work is changing the ReportCsvStreamer class to ensure that it handles printing to the excel file correctly.I am hoping to have a basic implementation complete for Tuesday.

**Notes on Relational Database and Object mapping**

Daqing Hou 4/11

Two important classes: ReportExportBase, ReportCsvStreamer

On Apr 11, 2013, at 11:47 AM, Daqing Hou wrote:

502  vi ./src/com/serotonin/mango/db/dao/ReportDao.java

 503  find . -name BaseDao.java

 504  vi ./src/com/serotonin/mango/db/dao/BaseDao.java

 505  vi ./src/com/serotonin/mango/db/dao/ReportDao.java

 506  ls

 507  cd lib

 508  ls

 509  jar tvf seroUtils.jar | grep Utils

 510  pwd

 511  cd ..

 512  ld

 513  ls

 514  cd src

 515  find . -name MangoContextListener.java

 516  vi ./com/serotonin/mango/MangoContextListener.java

 517\* find . -name D

 518\* vi ./com/serotonin/mango

 519  vi ./com/serotonin/mango/db/DatabaseAccess.java

 520  find . -name DerbyAccess.java

 521  vi ./com/serotonin/mango/db/DerbyAccess.java

 522  vi ./com/serotonin/mango/db/DerbyAccess.javaPLC

Online tutorials for PLC programming

Part 1: <http://iamechatronics.com/notes/general-engineering/261-plc-logic-programming-part-1>

Part 2: <http://iamechatronics.com/notes/general-engineering/262-plc-logic-programming-part-2>

Greenhouse PLC Programming

Clarkson University

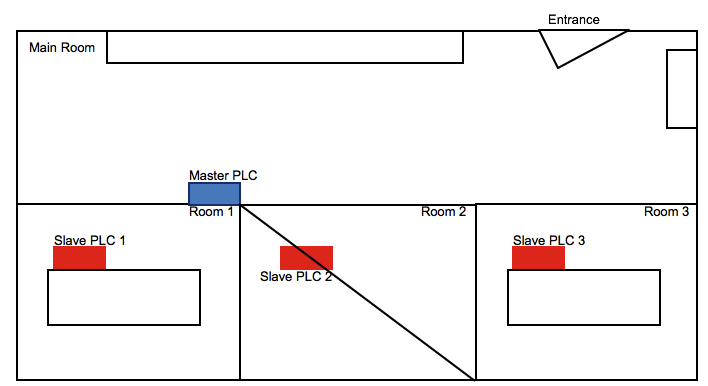
Spring 2013

Daniel Deloff

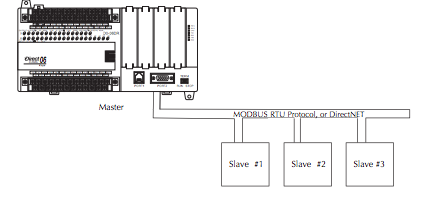
The purpose of this document is to inform progress made thus far, and to give objectives and plans for the future of the project for the upcoming weeks. This document will provide ladder logic code from the PLCs accompanied with diagrams and explanations. I have already identified the purpose of some sections of the code, but more progress has to be made.

Examining The Configuration:

Diagrammed below is a rough architecture of the greenhouse with the placement of the PLCs.



The PLCs are connected together in a Master-Slave configuration. The three PLCS in the small rooms are called the Slaves, with the fourth in the main room called the Master. A simple Master-Slave configuration can be diagramed as such:

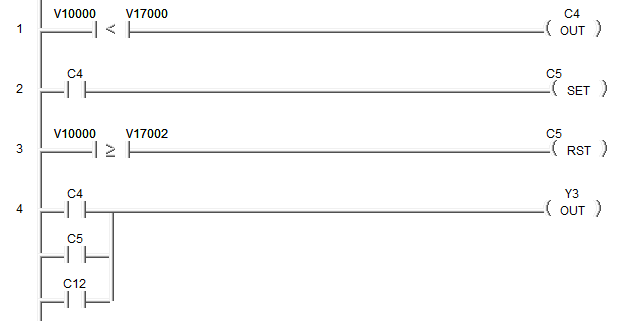


The Master station will read and write bytes to a slave station’s registers, one station at a time using the MODBUS protocol. Only the Master station has control over initiating requests to read and write data. The slave stations do not have control over when the data will be read, and are almost completely autonomous to each other. Inspecting the code on each slave station, I can conclude that they have the same processes in each. If the slave stations run the same processes on the same memory addresses, the Master station will only have to read bytes from a specific address from each slave.

Examining The Code:

To fully understand ladder logic on the DL06 PLC, we must examine the variable data types and understand how they written. There are five main data types: Input points, output points, control relays, timers, and counters. Input points are denoted with an X, output points are denoted with a Y, control relays are denoted with a C, timers are denoted with a T, and counters are denoted with a CT. X and Y data types are both physical input and output points on the PLC. These points are where wires can be connected to the device, so that when a variable is energized or turned high in the code, a signal is transmitted through the wire connected at the point. C data types are non-physical, internal points in the CPU itself, and can control program flow. Timers can be used to have certain processes run for a determined amount of time. And counters are used for limiting a certain process to run a certain number of times. Each of these data types, excluding the X and Y points, are stored in the system memory, or V memory. An address in the V memory is referenced by a capital V. For example, the address 10000 would be written as V10000. Knowing where the PLC reads and writes data in the memory, and knowing what the data is, is the key to understanding any Ladder Logic process.

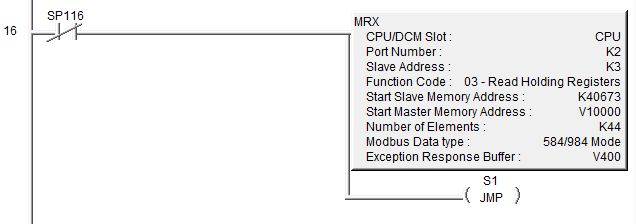
Shown below is a snippet of code from the Master station which controls the logic for toggling a water pump:



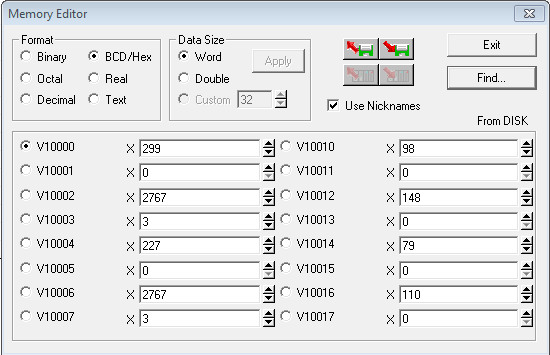
During execution of the program, the PLC will run each rung or line in the code in order. Each rung is labeled with a number to the far left. This snippet starts by comparing the value at V10000 with a value at V17000. If V10000 is lower than the threshold, the control relay C4 is energized or set equal to one. In the next rung, it checks to see if C4 is high, and if so, it sets C5 to high. Next, V10000 is compared to an upper threshold at V17002, and if it is greater than it, C5 will be turned low. Evaluating the values at V17000 and V17002 show that the memory addresses are equal to 30 and 60 respectively. On the last rung, if C4 or C5 is high, the output Y3 will be energized. This snippet of code is repeated two more times but with different outputs, including Y2 and Y1, and a different testing address in the V memory including V10200 and V10400. To my knowledge, control relay C12 has no function in the code, as this is the only time it is referenced in the program. Examining the PLC, wires are connected to Y3, Y2, and Y1, and run to water valves against the wall.

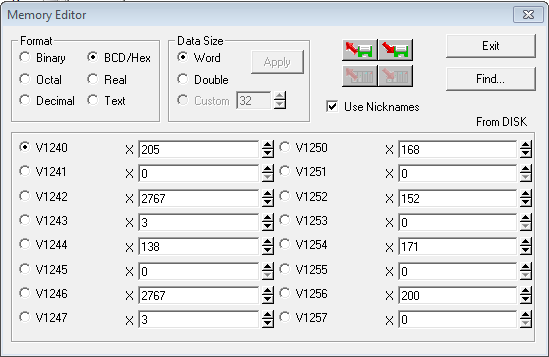


Below these three logic constructs the Master station, there are methods for writing and reading data from the Slave stations. There are three read methods and two write methods for each slave station. One of the read methods is shown below.



MODBUS read functions are defined by MRX as shown in the code. A MRX function has a number of attributes, including the Slave address, the Start Memory Address, and the Start Master Memory Address. The value for the Slave Address determines which slave to read from, the Start Slave Memory Address is the MODBUS memory address in the slave that is being read, and the Start Master Memory Address is the address where the read memory will be placed. The address K40673 is not an actual address found on the slave station, but a V memory address that has been converted into a MODBUS address. Using a .xls file, I was able to convert the MODBUS address back to a V memory address of V1240. Examining the contents of V1240 on the slave station and V10000 on the master station confirms this calculation.

 Memory on Master Station



Memory on Slave Station

The values may not be exact but however this is because the values were taken at different times.

Overview of The Water Pump Process:

-Values are read from the slave stations at address V1240.

-These values are then placed at locations V10000, V10020, and V10040 in the Master Station.

-The Master Station values are then compared with the lower threshold at V17000

- If less than, C4 is turned high.

- If C4 is high, C5 is set to high.

- The Master Station value is then compare with the upper threshold at V17002

- If greater than or equal, C5 is reset or turned off.

- Then if C4 or C5 is high, then the output is turned on, which turns on the pump.

- As long as V10000 is less than the upper threshold, the output Y3 will stay high.

Overview of The Master Station Code:

* Logic for controlling water pumps, repeated three times for each room in greenhouse.
* After, three MRX functions for Slave Station 1 are written, the first being the water pump data with the other two still unknown.
* After the three MRX functions, there are two MWX functions to write to the Slave Station data for a timing function inside the Slave Stations for possibly controlling the water sprayers.
* These three MRX functions and two MWX are repeated three times, for the three Slave Stations.
* After the writing and reading, the program ends and continues on to the beginning to start the process again.

Future Objectives:

To further document the PLC code, I must:

-Understand what the other read and write functions of the Master Station.

-Understand what sort of processes that the slave station performs.

-Research what the data actually equates to, not just numbers in memory.

It is my belief that I have a strong understanding as to how the Master Station functions, the next step is understand the Slave Station processes. My goal is to have a rough draft of the documentation in the next upcoming weeks.

**Sensors in GH**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sensor Type** | **Raw Signal** | **Engineering units** | **Part Number** | **Manufacturer** | **Link to data file** |
| pH | 4-20 mA | 2-12 | PHE-7151-15 | Omega | http://www.omega.com/pptst/PHE7151\_5312\_ORE7151.html |
| EC | 4-20 mA | 0-2 mS | CDE1202 | Omega | http://www.omega.com/Green/pdf/CDTX1200.pdf |
| Air Temp | RTD | 0-50 C | PRTF-10-2-100-1/4-6-E | Omega |  |
| Water Temp | RTD | 0-50 C | PRTF-10-2-100-1/4-6-E-SB | Omega |  |
| Radiator Temp | RTD | 0-50 C | SA1-RTD-B | Omega |  |
| Light Intensity | 0-5VDC | 0-1250 w/m^2 | SP-215 | apogee instruments | www.apogeeinstruments.com/specsheets/SP-212\_215specs.pdf |
| Light PPF | 0-5VDC | 0-2500 | SQ-215 | apogee instruments | www.apogeeinstruments.com/specsheets/SQ-200specs.pdf |
| Water tank Pressure | 4-20 mA | 0-100 in of water | PTD25-20-0100WCH | ProSense | www.automationdirect.com/static/specs/prosensetransmitters.pdf |
| Pump Pressure | 4-20 mA | 0-200 PSI | 628CR-12-GH-P1-E1-S1 | Dwyer Instruments | http://www.dwyer-inst.com/Product/Pressure/SinglePressure/Transmitters/Series628CR/Specs |
| Power meter | Modbus | NA | WNC-3Y-208-MB | CCS | http://www.ccontrolsys.com/w/WattNode\_Modbus\_-\_Specifications |

|  |
| --- |
| **Link to data file** |
| http://www.omega.com/pptst/PHE7151\_5312\_ORE7151.html |
| http://www.omega.com/Green/pdf/CDTX1200.pdf |
|  |
|  |
|  |
| www.apogeeinstruments.com/specsheets/SP-212\_215specs.pdf |
| www.apogeeinstruments.com/specsheets/SQ-200specs.pdf |
| www.automationdirect.com/static/specs/prosensetransmitters.pdf |
| http://www.dwyer-inst.com/Product/Pressure/SinglePressure/Transmitters/Series628CR/Specs |
| http://www.ccontrolsys.com/w/WattNode\_Modbus\_-\_Specifications |

EC is electroconductivity, its used to determine how high the salt concentration in a liquid is.

RTD is resistance temperature detector, in our case its a platinum wire that changes resistance proportional to temperature.

VDC is Voltage Direct Current.

PPFD: http://chameleongrowsystems.com/Chameleon\_Grow\_Systems/Photosynthetic\_Photon\_Flux\_(PPF).html

<http://en.wikipedia.org/wiki/Resistance_thermometer>

<http://en.wikipedia.org/wiki/Analog-to-digital_converter>

1. <http://mango.serotoninsoftware.com/home.jsp> [↑](#footnote-ref-1)
2. http://www.pdfsupply.com/pdfs/gfk1860a.pdf [↑](#footnote-ref-2)
3. http://www.moxa.com/doc/man/ioLogik\_E1200\_Series\_Users\_Manual\_v9.pdf [↑](#footnote-ref-3)
4. http://www.tizag.com/javascriptT/javascriptredirect.php [↑](#footnote-ref-4)