

User Interface - Feature #1526

Feature # 1522 (Closed): Jetty Upgrade

integrate the new version into P2J

09/10/2012 12:35 PM - Greg Shah

Status:	Closed	Start date:	10/25/2013
Priority:	Normal	Due date:	11/01/2013
Assignee:	Marius Gligor	% Done:	100%
Category:		Estimated time:	40.00 hours
Target version:	GUI Support for a Complex ADM2 App	version:	
billable:	No		
vendor_id:	GCD		
Description			

History

#1 - 09/24/2012 01:03 PM - Greg Shah

- Assignee set to Stanislav Lomany

#2 - 10/31/2012 02:58 PM - Greg Shah

- Target version set to Milestone 12

#3 - 10/20/2013 01:08 PM - Greg Shah

- Assignee changed from Stanislav Lomany to Marius Gligor

- Estimated time set to 40.00

- Due date set to 11/01/2013

- Start date changed from 09/10/2012 to 10/25/2013

See src/com/goldencode/p2j/main/WebServer.java which is the core of where we handle our Jetty integration. Take the patched version of Jetty 9.1 and replace the Jetty 6.1.14 that is currently in use.

This change will require automated runtime regression testing as well as manual testing of the "admin console" using the TIMCO Majic application. The good news is that once the admin console is loaded (it is an applet) and the admin user is shown to be able to login and do any normal function, that should be sufficient to prove that Jetty is doing its job. We don't have to test each function of the admin console, which will save a great deal of time.

#4 - 10/29/2013 01:57 PM - Marius Gligor

- Status changed from New to WIP

- % Done changed from 0 to 10

#5 - 10/30/2013 04:03 AM - Marius Gligor

I started to integrate the jetty 9.1 into the P2J project by replacing the old jetty 6.1.14 jars with the new jetty-all-9.1.0-SNAPSHOT.jar

Also the servlet-api-2.5-6.1.14.jar has been replaced with javax.servlet-api-3.1.0.jar as dependencies.

Four classes are affected by this upgrade: AppletHandler.java, ChuiAppletHandler.java, StandardServer.java and WebServer.java

I did a lot of changes (refactoring) on those classes looking at examples from <http://www.eclipse.org/jetty/documentation/current/>

Some of the changes are trivial, just changing the imports for the new packages, other are profound like on the WebServer.java

The changes aren't committed yet to my local repository.

After refactoring the code I have no compile errors and I did a build of the P2J project.

Starting the server and the client with the converted 4GL ask.p and primes.p everything works fine on the new build. However this is not enough to find out if the jetty 9.1 integration is completed.

Please helps me with more details related to automated runtime regression testing and manual testing of the "admin console" using the TIMCO Majic application?

#6 - 10/30/2013 10:37 AM - Greg Shah

The changes aren't committed yet to my local repository.

Good. If you commit to a bsr check-out of P2J, it will commit to the trunk which is not good until fully tested and approved.

Starting the server and the client with the converted 4GL ask.p and primes.p everything works fine on the new build. However this is not enough to find out if the jetty 9.1 integration is completed.

Correct. One simple thing to do here is to enable the admin console in the directory.xml and then connect to your local server's admin console. Please see the P2J Runtime Installation, Configuration and Administration Guide for details.

Please helps me with more details related to automated runtime regression testing and manual testing of the "admin console" using the TIMCO Majic application?

I will get you details on the regression testing shortly. Before then, it is important to do the following:

0. Test the simple server admin console as noted above.
1. Test your changes with the admin console in a P2J server setup for the converted TIMCO Majic application. This will be done with your devsrv01 account and I will provide details shortly.
2. Create an "update" zip file and upload it here for review.

#7 - 10/30/2013 11:46 AM - Marius Gligor

I have the following settings in directory.xml file of simple server

```
<node class="container" name="server">
  <node class="container" name="default">
    <node class="boolean" name="adminEnabled">
      <node-attribute name="value" value="TRUE"/>
    </node>
    <node class="container" name="chuiApplet">
      <node class="boolean" name="enabled">
        <node-attribute name="value" value="TRUE"/>
      </node>
      ....
    <node class="integer" name="adminPort">
      <node-attribute name="value" value="7443"/>
    </node>
  </node>
</node>
```

However when I started the server the admin console is not available.

I found no INFO messages on the server.log file regarding the SslSocketConnector starting.

I tested a P2J build without my changes, that is, I'm using the actual implementation using jetty 6.1.14

#8 - 10/30/2013 12:04 PM - Greg Shah

This is a good opportunity for you to debug through the server startup to see why Jetty is not initialized.

#9 - 10/31/2013 04:25 AM - Marius Gligor

I started the server using the following command line parameters:

net:connection:secure=true net:server:secure=3334

Trying to connect to <https://localhost:3334/admin> I've got an error:

javax.net.ssl.SSLException: Connection has been shutdown: javax.net.ssl.SSLHandshakeException: Received fatal alert: unknown_ca

Here I have 2 questions:

1. The secure port is used to communicate between nodes in the network?
2. Why we need also to define an insecured port when we already defined a secured one?

After a deep debug on the server code I found that the web server listen on port 7443 obtained from directory.

Also I changed the log level in directory from WARINING to INFO and the jetty INFO messages are now in the log

[10/31/2013 10:00:24 EET] (sun.reflect.NativeMethodAccessorImpl:INFO) jetty-6.1.14

[10/31/2013 10:00:24 EET] (sun.reflect.NativeMethodAccessorImpl:INFO) Started [SslSocketConnector@0.0.0.0:7443](#)

[10/31/2013 10:00:24 EET] (SessionManager.listen():WARNING) {00000000:00000001:standard} INSECURE sockets in use!

[10/31/2013 10:00:24 EET] (SessionManager.listen():INFO) {00000000:00000001:standard} Server ready

Using:

<https://localhost:7443/admin>

the AdminClient applet is being loading. First time I sow an GCD icon but since that I didn't succeed to load the applet.

The Firefox browser seems to be freeze loading the applet!
I'm going to debug the AdminClient applet.

#10 - 10/31/2013 07:40 AM - Marius Gligor

- File *adminconsole.png* added

The AppletHandler works properly and the applet code embedded on the admin_login.html after the placeholders are set looks like:

```
<applet code="com.goldencode.p2j.admin.client.AdminClient"
        archive="/admin/p2jadmin.jar"
        width="100%" height="100%">
    <param name="p2jservername" value="127.0.0.1">
    <param name="p2jserverport" value="3334">
    <param name="java_arguments" value="">
    <param name="separate_jvm" value="true">
</applet>
```

The p2jserverport is equal to secured port and I think that is used to communicate with p2server using the Remote Object Protocol.

The p2jadmin.jar could be downloaded from <https://localhost:7443/admin/p2jadmin.jar>

Finally I managed to start the AdminClient applet on my web browser (see the attached picture). Strange, but I have to use reload option from my web browser in order to load the applet because on the first load no controls are rendered on the applet canvas.

#11 - 10/31/2013 11:20 AM - Marius Gligor

I need to logon on the AdminConsole and I have to use an user id and a password.

1. It's OK to use the admin user and password registered in the directory.xml?
2. If yes, is the password encrypted?
3. If yes, how I could decrypt the password?

#12 - 10/31/2013 02:05 PM - Marius Gligor

- % Done changed from 10 to 50

Today I managed to run the AdminClient applet using the current P2J project. Also I studied the server and client code from AdminClient AdminLogon, and other related classes in order to learn more about the process flow. Next I've created, edited and deleted some users and I observed the changes on the directory.xml file. Finally I've created a new P2J build this time using my changes, jetty-all 9.1 jar and servlet 3.1 API jar.

I started the server again and the AdminClient seems to work properly on jetty 9.1 after my changes.
Nevertheless I have to do more tests on my local environment and after that I have to prepare the P2J for regression tests.

#13 - 11/01/2013 08:25 AM - Marius Gligor

- File *mag_upd20131101a.zip* added

Today I tested the P2J build having my changes.
First I tested from Eclipse IDE and I found to work properly.
Next I tested the server from the command line on a terminal window (server.sh) and found to work properly.
I changed also the manifest file in order to add the new jars and delete the old jars from classpath
On the attached archive are my changes and the new jar files.

#14 - 11/01/2013 08:32 AM - Marius Gligor

- File *deleted (mag_upd20131101a.zip)*

#15 - 11/01/2013 08:33 AM - Marius Gligor

- File *mag_upd20131101a.zip* added

This the archive containing my changes.

#16 - 11/01/2013 09:07 AM - Greg Shah

Code Review 1101a

The update is pretty good.

1. 9.1 is still not listed as the latest "stable" release for Jetty. Are there still changes being made to jetty 9.1 such that this snapshot is not going to be the "official" 9.1 stable release?
2. In WebServer, what is this code: `config.setSecurePort(8443)`? Where does the 8443 come from and what is its purpose? We can't have hard coded ports in P2J since we support multiple simultaneous P2J servers on the same system.
3. What is a SecureRequestCustomizer, why do we need it and what are the implications of using it?
4. Is there no equivalent to `connector.setHandshakeTimeout(2000)`?
5. The copyright date range for `AppletHandler.java` should be changed from 2009-2011 to 2009-2013.
6. The copyright date range for `ChuiAppletHandler.java` should be 2011-2013 instead of just 2011.

#17 - 11/01/2013 11:39 AM - Marius Gligor

- File *netbeans-7.4.desktop* added

1. Jetty 9.1 indeed is not a stable release and could be changed until the final release.

Jetty 9.0.6 is the latest stable release according to <http://download.eclipse.org/jetty/>

2. At least one `HttpConfiguration` should be used.

The port number is set on the `ServerConnector` and is not the port from the `HttpConfiguration`.

```
https == SSL + http
```

```
Or in connector terms, you connect first with SSL, and then just talk HTTP within that SSL connection.  
So the http configuration can be reduced just to:
```

```
HttpConfiguration config = new HttpConfiguration();
```

The snippet of code was taken from an example since the jetty embedded documentation is poor.

Only javadoc and a few examples.

Here is the javadoc for `HttpConfiguration`.

```
/* ----- /
```

```
/* HTTP Configuration. * <p>This class is a holder of HTTP configuration for use by the * {@link HttpChannel} class. Typically a HTTPConfiguration instance * is instantiated and passed to a {@link ConnectionFactory} that can * create HTTP channels (eg HTTP, AJP or SPDY).</p> * <p>The configuration held by this class is not for the wire protocol, * but for the interpretation and handling of HTTP requests that could * be transported by a variety of protocols. * </p>
```

```
*/
```

3. The `HttpConfiguration` accept a lot of customizer objects. Here is a description from the jetty examples

```
// SecureRequestCustomizer which is how a new connector is able to resolve the https connection before  
// handing control over to the Jetty Server.
```

The javadoc for `SecureRequestCustomizer` is missing in the jetty project!

According to my tests this customizer is not required and could be eliminated.

Here are some jetty 9.1 examples

<http://www.eclipse.org/jetty/documentation/current/embedded-examples.html>

4. So far I didn't found an equivalent. Nevertheless I will try to search if an equivalent exists by looking on the code, but this take a time since the jetty project became a big one since version 6, having many new features an changes.

5. Done.

6. Done.

#18 - 11/01/2013 11:44 AM - Marius Gligor

- File *mag_upd20131101b.zip* added

#19 - 11/01/2013 01:03 PM - Marius Gligor

- File *log_files.zip* added

- % Done changed from 50 to 70

Today I've finished to test the P2J integration of jetty 9.1 on my workstation.
Later I started to prepare the regression tests environment.
Unfortunately the build stops after 12 minutes due to a fatal error.
I attached the log files.

#20 - 11/01/2013 01:08 PM - Marius Gligor

- File *deleted (netbeans-7.4.desktop)*

#21 - 11/03/2013 07:04 AM - Greg Shah

1. Jetty 9.1 indeed is not a stable release and could be changed until the final release.

Do you know the target date for when they expect to make a final release?

2. At least one `HttpConfiguration` should be used.

The port number is set on the `ServerConnector` and is not the port from the `HttpConfiguration`.

I'm sorry, but I still don't understand what this port number is being used for. I understand that the docs are poor. Please look inside the source code to find the answer.

We can't hard code port numbers. If the port isn't really being used for communications, then why is it configured?

LE: I see that you have been able to remove the hard coded port from the configuration. That is good and we can drop this inquiry since it doesn't seem to be required.

Nevertheless I will try to search if an equivalent exists by looking on the code, but this take a time since the jetty project became a big one since version 6, having many new features an changes.

I understand. It is OK to look deeper into their code to find the answer.

#22 - 11/03/2013 07:10 AM - Greg Shah

Code Review 1101b

The code looks good. I see that you removed the concerning code with the fixed port 8443. That's great. It is still important to know if the removal of any of the other `HttpConfiguration` settings will cause a problem (and also if the handshake timeout can be set).

#23 - 11/03/2013 08:05 AM - Greg Shah

In regard to the failure in the regression testing environment, the problem location is here:

```
[java] -----
[java] Generate Hibernate Mappings (database P2Os)
[java] -----
[java] ./data/namespace/majic.p2o
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AccrualImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/ActionCodesImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AdditionalWorkRequestImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AdditionalWorkRequestSkillImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AirTypeImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AircraftImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AircraftVendorLinkImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AogImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AuditImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/Audit2Impl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AuditDueParametersImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/AuditMethodImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/BatchImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/BlanketBatchImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/BlanketJobTimeImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/BlanketTaClockImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/BuyerImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CcJobImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/ChargeMasterImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/ChartImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CheckNotesImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CheckTypeImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CloseOperationImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CloseRouteImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CmuFunctionImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CommissionDueImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CommissionTabImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CommissionTransactionImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/ContactImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/ContractImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CostRecognitionImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CountryImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CourseImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/Course2Impl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/Training2Impl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CrewImpl.hbm.xml
[java] #
[java] # A fatal error has been detected by the Java Runtime Environment:
[java] #
[java] # Internal Error (assembler_x86.inline.hpp:46), pid=25617, tid=139947532257024
[java] # guarantee(this->is8bit(imm8)) failed: Short forward jump exceeds 8-bit offset
[java] #
[java] # JRE version: 7.0_25-b30
[java] # Java VM: OpenJDK 64-Bit Server VM (23.7-b01 mixed mode linux-amd64 compressed oops)
[java] # Failed to write core dump. Core dumps have been disabled. To enable core dumping, try "ulimit -c
unlimited" before starting Java again
[java] #
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CrewHistoryImpl.hbm.xml
[java] # An error report file with more information is saved as:
[java] # /home/mag/testing/majic/hs_err_pid25617.log
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CrewOffImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CrewTempImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CrossReferenceEditImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CrossReferenceItemImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CtsImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CustomerApprovalImpl.hbm.xml
```



```
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CustomerItemCostImpl.hbm.xml
[java] /home/mag/testing/majic/src/aero/timco/majic/dmo/majic/impl/CustomerOrderAliasImpl.hbm.xml
[java] #
[java] # If you would like to submit a bug report, please include
[java] # instructions on how to reproduce the bug and visit:
[java] # https://bugs.launchpad.net/ubuntu/+source/openjdk-7/
[java] #
```

BUILD FAILED

/opt/secure/clients/timco/testing/build_rt.xml:141: Java returned: 134

The important parts of the hotspot log:

```
#
# A fatal error has been detected by the Java Runtime Environment:
#
# Internal Error (assembler_x86.inline.hpp:46), pid=25617, tid=139947532257024
# guarantee(this->is8bit(imm8)) failed: Short forward jump exceeds 8-bit offset
#
# JRE version: 7.0_25-b30
# Java VM: OpenJDK 64-Bit Server VM (23.7-b01 mixed mode linux-amd64 compressed oops)
# Failed to write core dump. Core dumps have been disabled. To enable core dumping, try "ulimit -c unlimited"
before starting Java again
#
# If you would like to submit a bug report, please include
# instructions on how to reproduce the bug and visit:
# https://bugs.launchpad.net/ubuntu/+source/openjdk-7/
#

----- T H R E A D -----

Current thread (0x00007f483012f000):  JavaThread "C2 CompilerThread1" daemon [_thread_in_native, id=25645, sta
ck(0x00007f4812e28000,0x00007f4812f29000)]

Stack: [0x00007f4812e28000,0x00007f4812f29000], sp=0x00007f4812f24b90, free space=1010k
Native frames: (J=compiled Java code, j=interpreted, Vv=VM code, C=native code)
V [libjvm.so+0x84e26b] VMError::report_and_die()+0x16b
V [libjvm.so+0x3ec04b] report_vm_error(char const*, int, char const*, char const*)+0x5b
V [libjvm.so+0x2380d4] Label::patch_instructions(MacroAssembler*)+0x104
V [libjvm.so+0x703bee] Compile::fill_buffer(CodeBuffer*, unsigned int*)+0x2be
V [libjvm.so+0x39be70] Compile::Code_Gen()+0x5c0
V [libjvm.so+0x3a0925] Compile::Compile(ciEnv*, C2Compiler*, ciMethod*, int, bool, bool)+0xc65
V [libjvm.so+0x3188e0] C2Compiler::compile_method(ciEnv*, ciMethod*, int)+0xe0
V [libjvm.so+0x3a631a] CompileBroker::invoke_compiler_on_method(CompileTask*)+0x32a
V [libjvm.so+0x3a7014] CompileBroker::compiler_thread_loop()+0x514
V [libjvm.so+0x80e512] JavaThread::thread_main_inner()+0xf2
V [libjvm.so+0x6f2e32] java_start(Thread*)+0xf2

Current CompileTask:
C2: 523139 2473 com.goldencode.p2j.pattern.TemplateWorker$Template::graftAt (242 bytes)
...
```

This tells us the following:

1. The failure is deep inside the JVM C++ code that is part of the C2 compiler. The OpenJDK JVM is based on the Sun "Hotspot" native compilation approach. Traditional JIT (just in time) compilers would compile all loaded bytecode into native code before execution. But with Hotspot, Sun only compiles code that is executed with enough frequency to ensure that the cost of native compilation is "paid for" by the resulting gains in execution speed. They thus interpret all bytecode to start with and then monitor which code is executed the most. Those hotspots are then compiled to native code. As the JVM executes this is an ongoing process as more and more code passes the thresholds for compilation and gets compiled (and sometimes re-compiled). OpenJDK, as with the other Hotspot JVMs before them, includes 2 completely different native compilers: C1 (the "client compiler") and C2 (the "server compiler"). Both compilers use different strategies for compilation. The C2 compiler is more aggressive about compiling code (lower thresholds) with the idea that the process will be long running (it is expected to be run on servers) and the optimizations applied during compilation are more aggressive as well. Thus the C2 compiler attempts to trade off more compilation time for better performance over the lifetime of the "server" process.
2. It is almost always the case that when you get hotspot log files (e.g. hs_err_pid25617.log), that the bug/failure is in the JVM. This is NOT a failure in any Java code written by us. The one exception to this rule is when you write native code accessed via JNI (or when you write native code that accesses the JVM through JNI). In this case, you are writing code that can be potentially unsafe and it is possible to contribute to the JVM failure from improperly written native code. Although we do have JNI code in our project, none of it is used at conversion time and I am confident that the problem is purely a bug in the JVM itself.

3. The failure occurred on one of the 2 dedicated C2 compiler threads that run as daemon threads in the JVM. The stack trace shows the code is deep in its own compilation logic.
4. The code being compiled is `com.goldencode.p2j.pattern.TemplateWorker$Template::graftAt()` which is definitely our code. But that code has not been changed in years. Sometimes it is possible to have a specific pattern of Java code that can trigger C1/C2/JIT compiler failures. These are rare, but when they happen, usually they fail consistently (the compilation of that Java code always fails). We know in this case that this failure does not always occur, since most of the time this never happens. It is possible that something in this particular code is more likely to trigger the problem.
5. The mention of the Internal Error and `guarantee(this->is8bit(imm8))` failed: Short forward jump exceeds 8-bit offset tells us that the internal state of the C2 compiler is corrupt. If I recall correctly, `guarantee` is a C++ macro. `guarantee` is a kind of runtime assertion that checks that a particular condition holds and if it does not, it will abort the JVM.
6. Eric recently (last weekend) saw some hotspot failures, but he did not save off his log files. Shame on you Eric. :) It is likely that he was seeing this same problem (since it is rare to see any hotspot problem, it is unlikely we are encountering more than one). But we can't know.

Action plan:

Marius: just re-execute the testing. The problem has nothing to do with your code and it is not even in the P2J code. It won't fail every time (or even most times).

Constantin: You and I will need to make it a priority to upgrade the software on devsrv01. I think we are pretty back level anyway (7u25-2.3.10-1ubuntu0.12.10.2 is our version and I think the stable version is 7u40 and 7u60 is being worked on). But it is the entire OS that is now 2 version behind with all the associated packages being backlevel too. The problem is that upgrading is tricky (because we have mismatching versions installed) and may cause breakage. I think we will try to get past this current M7 push and then do the upgrades, so that we don't cause any delay in our dev schedules.

#24 - 11/04/2013 02:26 AM - Marius Gligor

- File `mag_upd20131104c.zip` added

Regarding SSL handshake timeout.

`SslSocket` extends `Socket` class and provides secure socket using protocols such as the "SecureSockets Layer" (SSL) or IETF "Transport Layer Security" (TLS) protocols. The method `SslSocket#startHandshake()` starts a SSL handshake on this connection. This method is synchronous for the initial handshake on a connection and returns when the negotiated handshake is complete.

Basically the SSL handshake timeout is roughly the same as `Socket SO_TIMEOUT` timeout. On the `ServerConnector#setIdleTimeout(long idleTimeout)` javadoc I found the following description:

```
/**
 * <p>Sets the maximum Idle time for a connection, which roughly translates to the {@link Socket#setSoTimeout(int)} * call, although with
 * NIO implementations other mechanisms may be used to implement the timeout.</p>
 * <p>The max idle time is applied:</p>
 * <ul>
 * <li>When waiting for a new message to be received on a connection</li>
 * <li>When waiting for a new message to be sent on a connection</li>
 * </ul>
 * <p>This value is interpreted as the maximum time between some progress being made on the connection.
 * So if a single byte is read or written, then the timeout is reset.</p>
 * @param idleTimeout the idle timeout
 */
```

In conclusion this method is equivalent with the old `setHandshakeTimeout`.
I changed the value for `ServerConnector#setIdleTimeout(long idleTimeout)` to 2000 ms as in the old version.

1. jetty-9.1.0-SNAPSHOT is official documented on the Eclipse web site (<http://www.eclipse.org/jetty/documentation/current/jetty-javaee.html>)

The release date for this version is 2013-10-23 01:00:13

On the maven repository I found 9.1.0.M0 (milestone) 9.1.0.RC0 (release candidate) and 9.0.6.v20130930 (stable)

I don't know exactly when but it's possible to have a 9.1.0 stable version soon.

2. The `HttpConfiguration` seems to be used on mixing protocols where is possible to redirect http requests to https.

In P2J WebServer we have defined a single SSL connector for https, no mixing protocols.

The jetty `ServerConnector` constructor require a parameter `HttpConnectionFactory` instance otherwise a `NullPointerException` is throw at runtime.

The `HttpConnectionFactory` has two constructors.

If we do not pass a `HttpConfiguration` instance as parameter on the constructor a default will be created.

```
• ----- /
/* A Connection Factory for HTTP Connections. * <p>Accepts connections either directly or via SSL and/or NPN chained connection factories.
The accepted * {@link HttpConnection}s are configured by a {@link HttpConfiguration} instance that is either created by * default or passed in
to the constructor.
*/
public class HttpConnectionFactory extends AbstractConnectionFactory implements HttpConfiguration.ConnectionFactory {
    private final HttpConfiguration _config;

    public HttpConnectionFactory() {
        this(new HttpConfiguration());
        setInputBufferSize(16384);
    }

    public HttpConnectionFactory(@Name("config") HttpConfiguration config) {
        super(HttpVersion.HTTP_1_1.toString());
        _config=config;
        addBean(_config);
    }

    /** HTTP Configuration. * <p>This class is a holder of HTTP configuration for use by the * {@link HttpChannel} class. Typically a
    HTTPConfiguration instance * is instantiated and passed to a {@link ConnectionFactory} that can * create HTTP channels (eg HTTP, AJP or
    SPDY).</p> * <p>The configuration held by this class is not for the wire protocol, * but for the interpretation and handling of HTTP requests that
    could * be transported by a variety of protocols. * </p>
    */
}
```

The most interesting settings on this calss are:

```
- Set the URI scheme used for CONFIDENTIAL and INTEGRAL redirections. (default "https")
- Set the TCP/IP port used for CONFIDENTIAL and INTEGRAL redirections. (default 0)
```

A description of what is mixing protocols =====

Many web applications, especially those deployed for e-commerce, necessitate the transmission of sensitive data between the web server and the client browser. This data may include passwords, credit card numbers, bank account numbers or any other information that users would not want to divulge to the general public. To protect sensitive data during transmission, application developers typically use the Secure Sockets Layer (SSL) and its companion protocol, HTTP over Secure Sockets Layer (HTTPS). HTTPS employs SSL to protect data by encrypting it at the source, be it the server or the client, and decrypting it at the destination. This prevents anyone monitoring Internet data transmissions from easily capturing this data. The client and server exchange public keys to enable encryption and decryption to occur.

The encryption/decryption process comes at a performance price, however. The throughput of data for a web server transmitting via HTTPS is often as little as one-tenth that of data transmission via HTTP. For this reason, it is undesirable to deploy an entire web application under SSL. For fastest performance, it is best to deploy a web application under HTTP and employ HTTPS only for those pages and processes that transmit sensitive data.

Mixing Protocols in Web Applications

Switching back and forth between the two protocols can require hard-coding the protocol and full URL in every link to each resource in the web application. This creates an ongoing maintenance headache for developers each time a server name changes or secure protocol requirements change for resources in the web app.

Another significant hazard is that there is nothing to prevent a user from specifying the wrong protocol by manually entering a URL into the browser. The penalty for manually specifying HTTPS for a page or servlet that does not require HTTPS is reduced performance. Far worse is the penalty for manually specifying HTTP for non-secure access of a page that does require HTTPS: public exposure of sensitive data.

Help from Deployment Descriptor

To help overcome the problem of non-secure access of sensitive data, the Java Servlet Specification (versions 2.2 and 2.3) defines the transport-guarantee element of the web.xml deployment descriptor file. The transport-guarantee element must specify one of three types of protection for communication between client and server: NONE, INTEGRAL, or CONFIDENTIAL. For most containers a specification of INTEGRAL or CONFIDENTIAL is treated as a requirement for SSL usage. Web application containers will prevent users from accessing web resources over HTTP if they have been so specified.

The implementation for blocking HTTP access to web resources specified as INTEGRAL or CONFIDENTIAL varies from container to container. If a user attempts to access such a resource over HTTP, some containers will present that user with an error message instructing them to use the HTTPS protocol for accessing the requested resource. Other containers will actually redirect the request using the HTTPS protocol, but then continue using the HTTPS protocol for all subsequent requests, even those for resources with a transport-guarantee specification of NONE.

#26 - 11/04/2013 08:36 AM - Greg Shah

Code Review 1104c

1. The timeout can't be set to 2000ms.

In conclusion this method is equivalent with the old `setHandshakeTimeout`.

It may replace `setHandshakeTimeout()`, but it also seems to replace `setMaxIdleTime()`. So where we had a more granular configuration, we now have a single value. While 2 seconds may be OK for just the handshake timeout, it is much too short for general idle timeout processing on the connection. It looks like we have to stick with the 30 second value and accept that it will take 15 times as long to detect a dead handshake.

2. Please don't break out individual imports, our standard is to import * unless there is a specific conflicting class name that must be imported on its own.

3. The update filename suffix (a, b, c...) is an index of the update which you have personally done on that date. So your first update zip created on 1104 should be 1104a. There is no linkage to the prior update zips for the same task.

4. The stable jetty question.

I don't know exactly when but it's possible to have a 9.1.0 stable version soon.

No problem. It won't be hard to update versions in the next few months if we need to do so.

5. In regard to the mixing protocol issue, we do not plan to support http, only https. Please consider if we need to make any changes to disable http support.

#27 - 11/04/2013 09:06 AM - Marius Gligor

- File deleted (*mag_upd20131104c.zip*)

#28 - 11/04/2013 09:07 AM - Marius Gligor

- File `mag_upd20131104a.zip` added

The daily index should be a instead c.

#29 - 11/04/2013 09:09 AM - Marius Gligor

- File `mag_upd20131104b.zip` added

1. restore idle timeout to 30000 ms.
2. organize imports.

#30 - 11/04/2013 09:26 AM - Greg Shah

Code Review 1104b

It looks good. Put it through runtime regression testing. There is no need for conversion testing on this change.

Please consider if we need to make any changes to disable http support.

Do you have any thoughts in regard to this?

#31 - 11/04/2013 09:39 AM - Marius Gligor

1. We already disabled the http by defining a single SSL connector which use only the https protocol.
We have no connectors for http.

2. Regarding regression tests. I did the conversion and now I'm working to run the regression tests.
My first regression test failed on CTRL-C test. What's this?
I restarted the test and this time I'm using `\]` instead `]` on my password.
By the way the document which describe regression testing has a little miss, the ant task runtime-regression.

```
run_regression.sh <<< yourpassword is not correct
```

should be

```
run_regression.sh runtime-regression <<< yourpassword  
or  
run_regression.sh runtime-regression -Dmajic-built=true <<< yourpassword
```

when we want to skip the build of the MAJIC project.

#32 - 11/04/2013 10:00 AM - Greg Shah

My first regression test failed on CTRL-C test. What's this?

As you may have seen in the 4GL, the user can issue CTRL-C at any time (even when the UI is not enabled for editing/interaction) and it will interrupt the currently executing code and raise a STOP condition. Although Progress 4GL is single threaded, the CTRL-C abort processing is asynchronous. This means that we must include more complex logic to interrupt the processing. In addition, because of our architecture (thin client with no converted code + an application server where multiple users run on threads with isolated context), we inherently split the Progress 4GL fat client design across the network. That makes the CTRL-C processing significantly harder AND integrally tied into the network protocol and security implementation on both sides. Users can also invoke code that accesses remote P2J servers through their local P2J server. Although they have a direct connection to the local server, they have a virtual connection to the remote P2J server. This virtual connection can be multiplexed over a single server-to-server connection. This multi-server environment further complicates CTRL-C processing. Finally, the STOP condition must interact in very specific ways with the converted block structure of the resulting program. Some 4GL code may "catch" the STOP condition and other code may not. And there is default behavior that is hidden in the runtime too, to duplicate the 4GL behavior.

Because this is so tricky, we heavily test such scenarios using the so called CTRL-C regression tests. In addition to CTRL-C, we are also simulating client abends/the user killing their client process. This client disappears situation must be cleanly and properly handled by the server.

As noted in the regression testing text file (not the new docs yet), there are some known failures in CTRL-C testing which occur only on devsrv01 right now. Please see there for more details.

#33 - 11/04/2013 10:02 AM - Greg Shah

By the way the document which describe regression testing has a little miss, the ant task runtime-regression.

Good catch. I'll fix it. Thanks.

#34 - 11/04/2013 01:30 PM - Marius Gligor

- *File testing.zip added*

- *% Done changed from 70 to 90*

Today I started the the regression tests.
The first conversion was done without any error.

Currently I have a regression test running on devsvr01.
I started this tests by using the final changes on my source codes.
So far on the screen I have the following lines but the process is stopped.
I saw a ^C on the screen window but I didn't press the CTRL-C key.

```
[exec] * Running CTRL-C part...  
[exec]  
[exec] Running test plan... Completed in 1 hours, 6 minutes and 48.908 seconds. Status = FAILED.  
[exec] Generating HTML reports... Completed in 0.668 seconds.  
[exec] * CTRL-C part has failed!  
[exec] ** Running main part...  
[exec]  
[exec] Running test plan... ESC M : R is 8, tm is 8, bm is 19  
[exec] ESC M : R is 8, tm is 8, bm is 19  
[exec] ESC M : R is 0, tm is 0, bm is 3
```

I have to restart the regression test.
My password for devsvr01 contains "]" character.
Should this character escaped "]"?

#35 - 11/04/2013 01:49 PM - Greg Shah

I have to restart the regression test.

Make sure you wait long enough for the main part to finish. It can take 4-5 hours.

My password for devsvr01 contains "]" character.
Should this character escaped "]"?

Probably. I don't know for sure, it depends on how bash treats it.

#36 - 11/05/2013 10:47 AM - Marius Gligor

- File runtime_regression_logs.zip added

1. Today I started a new regression test from the beginning, including conversion. Unfortunately I got the same result FAILED! I have no idea why. Please have a look over the attached log files and give me a hint.

2. What I also don't understand is why the file src.diff is empty?
My expectation is to find here the differences generated by my changes.
I saw also that changes to MAJIC should be uploaded to ~/testing/updates/majic/
and the changes to P2J to ~/testing/updates/p2j/
Is src.diff file generated by the MAJIC build?
You said to upload my changes to ~/testing/updates/p2j/ is this correct?
Sorry but I'm a little bit confused regarding the changes upload location.

#37 - 11/05/2013 11:05 AM - Greg Shah

Please have a look over the attached log files and give me a hint.

Unfortunately, I don't have the time right now to look this over. The important thing for you to do is to look at each failing test which is not failing due to a failed dependency. Look at the details report for that test and read through the results for the test steps. Find the first failing test step and get some idea of what is going on at that point that might be due to your changes.

As noted in the regression testing notes (text file):

- some tests do fail intermittently, but not always
- some tests fail when run at a certain time of day but otherwise succeed
- one test always fails

If you have more than one test run and the "combined good results" from these show that all tests except the expected failure do pass, then you are OK.

On the other hand, if the same unexpected test failures occur over and over, then you have a real regression.

2. What I also don't understand is why the file src.diff is empty?

As far as I know, you have not changed anything associated with the conversion. The diffs in question are output from comparing the converted Java version of MAJIC from your current build against a prior converted version of MAJIC. If you make no conversion changes, then there should be no diffs.

I saw also that changes to MAJIC should be uploaded to ~/testing/updates/majic/ and the changes to P2J to ~/testing/updates/p2j/

Yes.

You said to upload my changes to ~/testing/updates/p2j/ is this correct?

Yes.

#38 - 11/05/2013 11:54 AM - Marius Gligor

- File *regression_tests_reports.zip* added

According to the regression tests reports attached the final result is PASSED.

#39 - 11/05/2013 12:02 PM - Constantin Asofiei

Marius: if the failures are related to the false negatives described in the automated testing regression notes, then yes, it is passed. Usually, note that the index page does not provide enough info to determine that the tests have passed or have failed; if you want us to review a test log which you are not sure of, sent an archive with the html pages for the failed tests via email. We don't want to post in the "public" Redmine area MAJIC info such as screens (from the regression testing logs) or MAJIC source code.

#40 - 11/06/2013 11:18 AM - Marius Gligor

- % Done changed from 90 to 100

- File *adminconsole_on_gso.png* added

- Status changed from WIP to Review

Testing Admin Console on MAJIC server:

1. Create on your home directory the following structure:

```
~/testing
/majic
/build
/lib
/run
/server
/p2j
/build
/lib
```

2. Download the MAJIC jar files to ~/testing/majic/build/lib

```
$ scp -P 2224 $USER@localhost:~/testing/majic/build/lib/*.jar ~/testing/majic/build/lib/
```

3. Download MAJIC server configuration files from ~/testing/majic/run/server

You may download the entire folder which take some time:

```
$ scp -P 2224 $USER@localhost:~/testing/majic/run/server/* ~/testing/majic/run/server/
```

Or you can download only some specific files

```
$ scp -P 2224 $USER@localhost:~/testing/majic/run/server/*-directory.xml ~/testing/majic/run/server/
```

```
$ scp -P 2224 $USER@localhost:~/testing/majic/run/server/*_server.xml ~/testing/majic/run/server/
```

```
$ scp -P 2224 $USER@localhost:~/testing/majic/run/server/*.store ~/testing/majic/run/server/
```

```
$ scp -P 2224 $USER@localhost:~/testing/majic/run/server/*.sh ~/testing/majic/run/server/
```

```
$ scp -P 2224 $USER@localhost:~/testing/majic/run/server/*.properties ~/testing/majic/run/server/
```

4. Build the p2j project and copy all the files from \$P2J_HOME/build/lib to ~/testing/majic/p2j/build/lib

```
$ cp -r -v $P2J_HOME/build/lib/ ~/testing/majic/p2j/build/lib
```

5. Go to ~/testing/majic/run/server

```
$ cd ~/testing/majic/run/server
```

6. Open the file gso-directory.xml on your preferred text editor and go to line:

```
<node-attribute name="value" value="jdbc:postgresql://localhost:<port_number>/gso_20090909_staging"/>
```

Replace the <port_number> with 25432. You already have a ssh tunnel when you run the gcd_remote.sh script which allow you to access the remote postgres server using the local port 25432.

You can check by trying to connect to the postgres server using the pgAdmin GUI tool using:

Host: localhost

Port: 25432.

Username and Password are listed on gso-directory.xml.

```
<node-attribute name="value" value="jdbc:postgresql://localhost:25432/gso_20090909_staging"/>
```

Save your changes and exit from editor.

7. From the same location start the server.

```
$ ./server.sh
```

8. Wait until the sever is ready. Check the server.log file from time to time:

```
$ tail server.log
```

until you'll see the following message:

```
[11/06/2013 16:36:25 EET] (SessionManager.listen():INFO) {00000000:00000001:gso} Server ready
```

or an error message if the server cannot be started for some reasons.

9. When the server is ready open the following URL on your web browser (Firefox):

https://localhost:<gso_admin_port>/admin

Where the <gso_admin_port> is the GSO admin port usually 7673.

The <gso_admin_port> is also listed on the gso-directory.xml file.

Note: If the applet is not rendered properly on the first load try to reload the page.

One easy way to do that is to press CTRL-R if you are using Mozilla Firefox.

10. Enter your credentials and enjoy your work.

For testing purposes use the following credentials:

username: admin

password: test123

#41 - 11/06/2013 05:10 PM - Greg Shah

All that remain is to commit the changes on the bazaar repository and send an email to the team members to inform them about my changes.

Yes, go ahead and check it in and distribute it.

I saw that on test regression tests both the old and the new jar files are kepted. This is not a problem since the classpath is explicitly on the manifest file. However I think that we can remove now the old jar files and the patch file from repository and add the new jar files.

Yes, we certainly don't want to continue including the old jetty jar or the patch file. Please remove them as part of the check in.

#42 - 11/07/2013 09:01 AM - Greg Shah

- Status changed from Review to Closed

Committed to bzt revision 10410.

#43 - 11/16/2016 12:13 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

#44 - 11/30/2016 08:38 AM - Greg Shah

- File deleted (testing.zip)

#45 - 11/30/2016 08:38 AM - Greg Shah

- File deleted (runtime_regression_logs.zip)

#46 - 11/30/2016 08:38 AM - Greg Shah

- File deleted (regression_tests_reports.zip)

Files

adminconsole.png	42.9 KB	10/31/2013	Marius Gligor
mag_upd20131101a.zip	2.45 MB	11/01/2013	Marius Gligor
mag_upd20131101b.zip	2.45 MB	11/01/2013	Marius Gligor
log_files.zip	45.8 KB	11/01/2013	Marius Gligor
mag_upd20131104a.zip	2.45 MB	11/04/2013	Marius Gligor
mag_upd20131104b.zip	2.45 MB	11/04/2013	Marius Gligor
adminconsole_on_gso.png	41.8 KB	11/06/2013	Marius Gligor