

Build and Source Control - Feature #2699

convert the ant build environment to gradle

09/09/2015 05:35 PM - Greg Shah

Status:	Closed	Start date:	
Priority:	Normal	Due date:	
Assignee:	Hynek Cihlar	% Done:	100%
Category:		Estimated time:	0.00 hour
Target version:	Deployment and Management Improvements		
billable:	No	vendor_id:	GCD
Description			
Related issues:			
Related to Build and Source Control - Feature #4645: Migrate Ant build logic ...			New

History

#1 - 09/09/2015 05:43 PM - Greg Shah

Based on my research, I can see the following paths forward:

- 1. Continue using ant and add ivy for dependency management.
- 2. Use Maven.
- 3. Use Gradle.

Adding ivy to ant is possibly less change, but the overall result is not nearly as good as using Gradle.

Maven forces you to live within their conventions (e.g. project structure) and has serious limitations in multi-module projects. It lacks flexibility and overall it seems like much more work than is necessary.

I don't see any better option than Gradle, though if someone has other evidence as to a better alternative I am happy to consider it.

Assuming that Gradle is the tool, as part of this conversion, if it is non-trivial to re-implement some features in Gradle, it is OK to use their ant support to reuse portions of the existing build environment.

Likewise, we should consider how exactly to integrate the native build portions.

This task will also move all of our project dependencies (e.g. p2j/lib/*) out of our source control system. We should consider whether it makes sense to support a binary artifact repo (e.g. Artifactory). I think this may make sense so that we don't have to rely upon 3rd party repos. Gradle should give us nice tools for this.

I want to add a distribution "target" so that we can easily package up the results of a build for installation.

#2 - 08/12/2016 01:23 PM - Greg Shah

- Start date deleted (09/09/2015)

The following article has a nice summary of the comparison between ant, maven and gradle.

<http://www.drdobbs.com/jvm/why-build-your-java-projects-with-gradle/240168608>

#3 - 08/15/2016 10:35 AM - Greg Shah

A useful comment from Hynek:

I used Gradle on one small project. From the short experience my only objection was its very unintuitive weakly-typed language/DSL. From the examples I've seen it can get pretty messy and hard to understand (even worse than Ant). At the same time I think Gradle is probably the best option to go with at the moment. There is a new generation of build systems emerging, like Kobalt (<http://beust.com/kobalt/home/index.html>), which feel to address the language complexity, but these are still too unstable.

#4 - 08/15/2016 11:00 AM - Hynek Cihlar

Greg Shah wrote:

A useful comment from Hynek:

I used Gradle on one small project. From the short experience my only objection was its very unintuitive weakly-typed language/DSL. From the examples I've seen it can get pretty messy and hard to understand (even worse than Ant). At the same time I think Gradle is probably the best option to go with at the moment. There is a new generation of build systems emerging, like Kobalt (<http://beust.com/kobalt/home/index.html>), which feel to address the language complexity, but these are still too unstable.

Btw., Gradle 3.0 stable has been released just this month with (among others) Kotlin support.

From my own experience Kotlin is a more straightforward language for a Java developer compared to Groovy. It doesn't try to come up with its own type system, instead it cleverly extends the existing Java system and reuses the standard Java APIs. I think it is halfway between Java and Scala, it only adopts the most useful features of a modern language (that are also seen in Scala) that bring the most benefits.

Kotlin on top of Gradle could be a way to simplify the build scripts, I believe.

#5 - 08/15/2016 11:32 AM - Hynek Cihlar

Hynek Cihlar wrote:

Kotlin on top of Gradle could be a way to simplify the build scripts, I believe.

Some links for the subject.

<http://melix.github.io/blog/2016/05/gradle-kotlin.html> (written by one of the Groovy's active committers)
<https://gradle.org/blog/kotlin-meets-gradle/>

#6 - 08/15/2016 01:45 PM - Greg Shah

Kotlin on top of Gradle could be a way to simplify the build scripts, I believe.

I'm OK with this approach, so long as what we need to do is possible. I can see the IDE tooling to be very useful (to those that use IDEs).

#7 - 08/15/2016 02:10 PM - Greg Shah

As noted above, moving to a modern build environment seems an opportune time to move our dependencies out of version control. This begs the question: how should we manage these dependencies?

Some items in our dependencies will be customized versions of an otherwise well known open source project. Our patched version of Hibernate falls into this category. Since we must build these artifacts, managing them carefully is quite important. Over time there may be multiple versions in use, matching the different P2J versions available.

Other dependencies may include versions that are back level from the latest found in public repositories. So long as the back level versions remain available publicly, we might just want to cache them so that there is a single place to find such artifacts.

Publishing these for our customers will be needed. And we will need a release process for managing the specific revisions that are published. So far, we have not needed such a process but we are soon going to need to take a more formal approach.

We already provide a custom build line of P2J as a service. This means that specific customers may need to be able to access their own build (as well as the trunk builds). The solution must allow granular access control to make this possible.

These requirements may make the use of a binary repository. If we go forward with this, we would plan to host this ourselves. We would only use a binary repo that is open source. I'm also convinced that our requirements are quite limited in this regard, so I would prefer to limit the complexity of any solution.

The two most obvious choices are Apache Archiva and jFrog Artifactory. See <https://binary-repositories-comparison.github.io/> for a comparison of features.

Archiva:

<https://archiva.apache.org/index.cgi>
<https://archiva.apache.org/docs/2.2.1/quick-start.html>
<https://www.mkyong.com/maven/how-to-install-apache-archiva-in-ubuntu/>
<https://archiva.apache.org/docs/2.2.1/userguide/deploy.html>

Artifactory:

<https://www.jfrog.com/open-source/#os-arti>
<https://www.jfrog.com/confluence/display/RTF/Installing+Artifactory>
<http://www.naturalborncoder.com/methodology/2015/05/26/artifactory-on-ubuntu-14-04/>

Artifactory is more popular, has more features and is generally much more integrated with other projects. But the setup and configuration of it is more complicated, which also means that the maintenance of it over time is going to be more costly. For example, it runs using a bundled version of Apache Tomcat 8. This means that we have to properly configure and secure a Tomcat installation, which is time consuming and tricky enough that it will need to be revisited over time. On a positive note, we are already securing some other Tomcat installs at GCD, but it is still extra work and there is

more security exposure than with Archiva. Another example is that by default the proxying support is not configured (if I understand properly), getting it done right is more work.

Archiva allows a standalone mode with a bundled Jetty environment. And its default setup is already quite close to our requirements. It may not have as much headroom to grow as our requirements expand, but it is likely to be much easier to start with. Although we would still need to configure SSL support (and disable the non-secure HTTP), I think the default configuration is already pretty close to what we need.

I don't know the right answer yet and I am open to thoughts. I do know that adding a production server VM for this will cost more time than I would like to get it 100% correct.

The other consideration here is public accessibility and how we properly secure it. I think both solutions have some capabilities in this regard, but it will need further exploration and configuration.

#8 - 08/15/2016 02:34 PM - Greg Shah

- Assignee set to Hynek Cihlar

#9 - 11/16/2016 01:06 PM - Greg Shah

- Target version changed from *Deployment and Management Improvements* to *Deployment and Management Improvements*

#10 - 01/08/2017 02:13 PM - Hynek Cihlar

Created task branch 2699a and committed an initial version of gradle build script.

This is a hybrid solution where the new gradle script coexists with the original ant script. The gradle's main responsibility is to resolve dependencies and pass them to the original ant build.

The checked in version is not complete. Although it already passes all build targets, the set of resolved dependencies requires cleanup. I hope to be able to exactly match the libs currently in the p2j/lib project folder.

#11 - 01/08/2017 02:20 PM - Hynek Cihlar

In p2j/lib project folder we have a custom patched hibernate-core library. Greg, in order to also remove it from p2j/lib, we will have to establish a public repository and move the patched lib there. I think the repository could be a simple static HTTP endpoint, installing a Nexus server (or similar full-featured repo) would be an overkill for this case.

#12 - 01/08/2017 04:18 PM - Greg Shah

I think the repository could be a simple static HTTP endpoint

Can we specify any URL? How about uploading the jar (or jars) to Redmine and pointing at that?

#13 - 01/09/2017 02:22 AM - Hynek Cihlar

Greg Shah wrote:

I think the repository could be a simple static HTTP endpoint

Can we specify any URL? How about uploading the jar (or jars) to Redmine and pointing at that?

Yes, we can do that. We will have to upload the repo metadata files (Maven POM) together with the jar files and put them to the formal directory structure (in case of Maven the jar must be located in the artifacts namespace).

#14 - 01/09/2017 08:42 AM - Greg Shah

Why do we need a Maven POM? Aren't we using Gradle's built-in dependency management?

#15 - 01/09/2017 08:43 AM - Hynek Cihlar

2699a revision 11136 resolves dependencies so that they match the lib dir. There are two exceptions:

- antlr.jar, the file in lib seems to be quite archaic. the closest (newer) version I could get from the official repositories was 2.7.7
- jta.jar, again, the closest newer version I could find was 1.1. But since this is only API spec this shouldn't be a problem.

I did a quick smoke-test, successfully reconverted and ran Hotel GUI.

The next steps will be to resolve the patched libs, finalize the build script and remove the lib files from lib dir.

#16 - 01/09/2017 08:45 AM - Hynek Cihlar

Greg Shah wrote:

Why do we need a Maven POM? Aren't we using Gradle's built-in dependency management?

It could be Ivy, too. Gradle works with multiple repository formats.

#17 - 01/09/2017 09:08 AM - Greg Shah

I thought that we would be defining the dependencies in the gradle build script itself and that there would be no need for a Maven POM (or for use of

Ivy). Am I misunderstanding the capabilities of gradle?

#18 - 01/09/2017 09:26 AM - Hynek Cihlar

Greg Shah wrote:

I thought that we would be defining the dependencies in the gradle build script itself and that there would be no need for a Maven POM (or for use of Ivy). Am I misunderstanding the capabilities of gradle?

Yes, this is how it works. Gradle build script contains dependency definitions and also the target repositories where the dependencies (also called artifacts) are fetched from. Every artifact stored in a repository needs metadata information, this is the POM file in case of Maven repository or an Ivy file in case of Ivy repository.

If we want to move the patched libs out of lib dir, the only way is to put them in a repository recognized by Gradle (Maven, Ivy, etc). In order for the patched libs to be recognized as valid repository artifacts they must be accompanied by the metadata files (Maven POM, Ivy file, etc.).

#19 - 01/09/2017 09:29 AM - Hynek Cihlar

Hynek Cihlar wrote:

Greg Shah wrote:

I thought that we would be defining the dependencies in the gradle build script itself and that there would be no need for a Maven POM (or for use of Ivy). Am I misunderstanding the capabilities of gradle?

Yes, this is how it works. Gradle build script contains dependency definitions and also the target repositories where the dependencies (also called artifacts) are fetched from. Every artifact stored in a repository needs metadata information, this is the POM file in case of Maven repository or an Ivy file in case of Ivy repository.

If we want to move the patched libs out of lib dir, the only way is to put them in a repository recognized by Gradle (Maven, Ivy, etc). In order for the patched libs to be recognized as valid repository artifacts they must be accompanied by the metadata files (Maven POM, Ivy file, etc.).

In short, we will need metadata files only for those artifacts we ourselves put into the repository.

#20 - 01/09/2017 09:30 AM - Greg Shah

If we need to match a specific directory structure, then Redmine will not be a good fit. I don't think we can control the structure in Redmine.

On the other hand, we use Apache2 2.4.18 as the web server on proj.goldencode.com. We can expose the files there. Please prepare the configuration changes needed for me to add those to the web server.

I also want to enable offline use from a local file system. There are plenty of times when that makes sense and I prefer to avoid people being dependent upon a slow (or missing) internet connection in order to setup a new branch or installation.

#21 - 01/09/2017 09:31 AM - Greg Shah

In short, we will need metadata files only for those artifacts we ourselves put into the repository.

This makes sense, thanks.

#22 - 01/09/2017 10:21 AM - Hynek Cihlar

Greg Shah wrote:

I also want to enable offline use from a local file system. There are plenty of times when that makes sense and I prefer to avoid people being dependent upon a slow (or missing) internet connection

Gradle caches the resolved artifacts on a local file system and it allows to work completely offline from the cached files. Please see https://docs.gradle.org/current/userguide/dependency_management.html#sec:cache_command_line_options.

in order to setup a new branch

Can you please provide more info about this use case?

or installation.

As part of the build process the resolved dependency jar files are copied to the build/lib directory. The new gradle build process only differs in where the dependencies are resolved from (repo instead of lib dir). So the obvious workflow is to build on a machine with access to the project artifacts repository (private or public) and install the bundle with all dependencies to the target environment.

#23 - 01/09/2017 10:51 AM - Greg Shah

I'd like to enable a workflow like this:

1. User is on redmine, uses the Download page to pull down some snapshot of FWD.
2. As part of this, they also download a zip of a local artifact repo that has all the dependencies in it.
3. They disconnect from the internet and are on some desert island when they decide to build FWD. They point to the local artifact repo and the gradle build works properly.

#24 - 01/09/2017 11:13 AM - Hynek Cihlar

Greg Shah wrote:

I'd like to enable a workflow like this:

1. User is on redmine, uses the Download page to pull down some snapshot of FWD.
2. As part of this, they also download a zip of a local artifact repo that has all the dependencies in it.
3. They disconnect from the internet and are on some desert island when they decide to build FWD. They point to the local artifact repo and the gradle build works properly.

This can be achieved with Gradle as follows.

1. User is on redmine, uses the Download page to pull down some snapshot (sources) of FWD.
2. The user builds with gradle. As part of the build process all dependencies are fetched and cached on the user's local file system.
3. The user disconnects from the internet and is on some desert island when they decide to build FWD. The user build as usual, the dependencies are taken from the user's local file system (--offline parameter may be needed when invoking gradle).
4. The user can alternatively download the binary bundle which will contain all the required dependencies.

One of the advantages of using gradle (and the public repositories) is that the dependencies are not fetched from GCD's servers (except of the patched libs) but from the public repositories. The public repositories are usually served from CDNs (Content Delivery Networks) and thus are much much faster.

#25 - 01/09/2017 12:25 PM - Hynek Cihlar

Does anybody know if the file p2j/lib/H2Dialect.java.txt is used anywhere? I could not find any reference myself.

#26 - 01/09/2017 01:46 PM - Hynek Cihlar

- File p2j-hibernate-core.zip added

The attached zip file contains the patched hibernate-core with the directory structure expected by Gradle. Note that no metadata are needed as I previously thought so, Gradle is smart enough to just read the jar file if it doesn't find the metadata file.

Greg, please upload the content of the attached zip file to a public http endpoint. So that the resulting URL is as follows `http(s)://<reposerver>/<repopath>/com` where 'com' is the root content of the attached zip.

#27 - 01/09/2017 02:31 PM - Greg Shah

I think I have Apache2 configured properly on proj.goldencode.com.

Try: `https://proj.goldencode.com/artifacts/com/...`

#28 - 01/09/2017 02:50 PM - Hynek Cihlar

Greg Shah wrote:

I think I have Apache2 configured properly on proj.goldencode.com.

Try: `https://proj.goldencode.com/artifacts/com/...`

Thanks, works OK.

#29 - 01/10/2017 09:55 AM - Hynek Cihlar

2699a revision 11141 contains a release-candidate of p2j build powered by gradle dependency resolution. Please review.

The build process now uses both ant and gradle. The gradle's role is to resolve project dependencies, while ant contains the old good build logic. The new build process goes as follows. gradle is executed with the build task(s) (aka target in ant), dependencies are resolved by gradle and the build task(s) executed in ant. The resolved dependencies are passed to ant as ant path structures. Other than that the build logic didn't change.

The project root directory contains the script gradlew/gradlew.bat (also known as gradle wrapper script) that is used as the build entry point. The advantage of the wrapper is that gradle doesn't have to be installed in order to build p2j.

To start the build gradlew is invoked with the desired tasks (previously ant targets). Ex.: `gradlew clean jar`.

For compatibility reasons build can be initiated by ant, too. So the above example will work equally well when issuing the command `ant clean jar`.

I made the following changes to the project structure. The jar and zip dependencies in lib were removed. The license text file in lib were moved to license dir. The patch sources in lib were moved to patch dir.

The remaining tasks are to move the ant build logic to gradle and remove the ant build script.

#30 - 01/10/2017 11:50 AM - Hynek Cihlar

Greg, FYI, I am getting a "connection refused" error when trying to regression test the changes in 2699a on devsrv01. Obviously the internet access there is limited.

#31 - 01/12/2017 04:55 AM - Greg Shah

Hynek Cihlar wrote:

Greg, FYI, I am getting a "connection refused" error when trying to regression test the changes in 2699a on devsrv01. Obviously the internet access there is limited.

Yes, both inbound and outbound traffic are limited. I have opened outbound 443. Try it now.

#32 - 01/12/2017 05:51 AM - Hynek Cihlar

Greg Shah wrote:

Yes, both inbound and outbound traffic are limited. I have opened outbound 443. Try it now.

Works OK, thanks.

#33 - 01/17/2017 04:11 AM - Hynek Cihlar

2699a passed ChUI regression tests.

#34 - 01/17/2017 04:00 PM - Greg Shah

Code Review Task branch 2699a Revision 11147

This is pretty close. Some thoughts and questions:

1. I want to remove the patches from bsr. We will provide them separately through Redmine. I think the patches/ directory can be removed.
2. I want to remove the license files from bsr. Since we are not distributing those jars with our code, I don't think they need to be in our project repo. I think the license/ directory can be removed, unless you know of some reason we need it.
3. Please remove the support for building p2j_rt.jar and the related manifest. It doesn't work and leaving it in is more confusing than helpful.
4. Is it common to have the gradle bootstrapping code included inside a project? Although I'm a big fan of making things easy and automated, I prefer not to include the jars and gradle wrapper scripts in our code repo. I installed gradle and ran it directly and it runs very nicely. We already assume you have to install ant. It doesn't seem too onerous to have to install gradle too.

Greg Shah wrote:

Code Review Task branch 2699a Revision 11147

This is pretty close. Some thoughts and questions:

1. I want to remove the patches from bsr. We will provide them separately through Redmine. I think the patches/ directory can be removed.

OK.

2. I want to remove the license files from bsr. Since we are not distributing those jars with our code, I don't think they need to be in our project repo. I think the license/ directory can be removed, unless you know of some reason we need it.

I am not aware of any reason we should keep the license files, I will remove them.

3. Please remove the support for building p2j_rt.jar and the related manifest. It doesn't work and leaving it in is more confusing than helpful.

OK.

4. Is it common to have the gradle bootstrapping code included inside a project? Although I'm a big fan of making things easy and automated, I prefer not to include the jars and gradle wrapper scripts in our code repo. I installed gradle and ran it directly and it runs very nicely. We already assume you have to install ant. It doesn't seem too onerous to have to install gradle too.

Yes, it is a recommended practice to include the 50KB gradle bootstrap jar in the code repository. The motivations are,

- no need to document the gradle version to download, no need for the developers to search through the docs to figure out which version to download, everyone (including build servers) is guaranteed to reproduce the same outcome from the build,
- you save the time installing gradle on every dev and build station,
- the build tool version is versioned together with the project sources, if for any reason we decide to use another version of gradle, we will still be able to reproduce the build correctly for past versions.

#36 - 01/17/2017 04:22 PM - Greg Shah

no need to document the gradle version to download, no need for the developers to search through the docs to figure out which version to download, everyone (including build servers) is guaranteed to reproduce the same outcome from the build,

...

the build tool version is versioned together with the project sources, if for any reason we decide to use another version of gradle, we will still be able to reproduce the build correctly for past versions.

These are unexpected to me. Why would different versions of gradle yield different builds? I would have expected the build results to be defined by the build.gradle and build.xml and that the results should be repeatable with multiple versions of gradle.

#37 - 01/17/2017 04:52 PM - Hynek Cihlar

Greg Shah wrote:

no need to document the gradle version to download, no need for the developers to search through the docs to figure out which version to download, everyone (including build servers) is guaranteed to reproduce the same outcome from the build,

...

the build tool version is versioned together with the project sources, if for any reason we decide to use another version of gradle, we will still be able to reproduce the build correctly for past versions.

These are unexpected to me. Why would different versions of gradle yield different builds? I would have expected the build results to be defined by the build.gradle and build.xml and that the results should be repeatable with multiple versions of gradle.

Gradle is a complicated beast and I can imagine breaking changes between versions in different areas: DSL, dependency resolution, default configurations of gradle plugins.

I've seen this with maven in the past. With one of the new major versions of maven, some of the projects at that time started to fail. We had to update the build scripts for all the failing projects, make sure everyone was on the same maven version, update build machines. Having the gradle wrapper script at that time we would have saved ourselves some trouble.

While at GCD we are a small team where the build environment can be easily kept under control, we have no control over the open-source community or other customers if they decide to build FWD themselves. Having the build tools coupled (and versioned) with the sources is IMHO an elegant way how to transparently ensure a stable build process.

#38 - 01/17/2017 05:03 PM - Greg Shah

OK, we'll go with the checked in wrapper and bootstrap jar.

#39 - 01/18/2017 06:46 AM - Hynek Cihlar

Greg Shah wrote:

1. I want to remove the patches from bsr. We will provide them separately through Redmine. I think the patches/ directory can be removed.
2. I want to remove the license files from bsr. Since we are not distributing those jars with our code, I don't think they need to be in our project repo. I think the license/ directory can be removed, unless you know of some reason we need it.
3. Please remove the support for building p2j_rt.jar and the related manifest. It doesn't work and leaving it in is more confusing than helpful.

1-3 resolved in 2699a revision 11151.

#40 - 01/18/2017 07:08 AM - Hynek Cihlar

2699a rebased against trunk revision 11136.

#41 - 01/18/2017 08:04 AM - Greg Shah

Code Review Task Branch 2699a Revision 11153

I'm fine with the changes. Once the final changes from [#28](#) are included, this can go into a final round of regression testing.

#42 - 01/23/2017 03:46 PM - Greg Shah

Eric reviewed the 11164 revision as is OK. As noted in 28, it has passed testing. Please merge 2699a to trunk.

#43 - 01/23/2017 04:26 PM - Hynek Cihlar

2699a was merged to trunk as revision 11138 and archived.

#44 - 01/23/2017 04:34 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

These two ideas would be useful to implement at some point:

- migrate the ant tasks to gradle and eliminate use of ant
- implement builds/deployment bundles for specific targets (client, application server, database user-defined functions...) and minimize the code and dependencies to just those needed for the target being built

Although there is more that we could do to implement gradle, I'm not going to hold this task open.

Other ideas can be posted here for future reference.

#45 - 01/24/2017 12:09 PM - Hynek Cihlar

In 2699b:

Prevented build failure when building from unsupported bsr location while version.properties missing definitions for repo, branch and rev properties. Also indications added to inform about such a situation.

Please review.

#46 - 01/25/2017 08:47 AM - Greg Shah

Code Review Task Branch 2699b Revision 11138

I'm fine with the change. If you have tested it enough to know it is safe, then you can merge to trunk.

#47 - 01/27/2017 08:24 AM - Greg Shah

Do you know when 2699b will be merged to trunk?

#48 - 01/27/2017 09:20 AM - Hynek Cihlar

Greg Shah wrote:

Do you know when 2699b will be merged to trunk?

I haven't had change to finish testing, that is a successful start of ChUI regression tests. I will do it now. Can I go ahead with the merge when I get a pass?

#49 - 01/27/2017 09:21 AM - Greg Shah

Yes, you can go ahead. You should rebase before testing.

#50 - 01/27/2017 10:39 AM - Hynek Cihlar

2699b rebased, merged to trunk as revision 11140 and archived.

#51 - 01/27/2017 11:13 AM - Eric Faulhaber

Hynek, building p2j with either of the following command lines using trunk rev. 11139 fails for me:

```
gradlew all -Dpost.build=yes -Dspawn.install.folder=/opt/spawner -Dsrv.certs=~/.spawner/hotel-certs.store
ant all -Dpost.build=yes -Dspawn.install.folder=/opt/spawner -Dsrv.certs=~/.spawner/hotel-certs.store
```

This is the error output using gradlew:

```
...
[ant:javac] Note: Some input files use unchecked or unsafe operations.
[ant:javac] Note: Recompile with -Xlint:unchecked for details.
[ant:javac] 20 warnings
:ant-aspectj
:ant-jar
```

```
:ant-native_prepare
> Building 87% > :ant-native[sudo] password for ecf:
:ant-native FAILED
```

FAILURE: Build failed with an exception.

```
* What went wrong:
Execution failed for task ':ant-native'.
> exec returned: 2
```

```
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
```

BUILD FAILED

Total time: 33.248 secs

This is the error output using ant:

```
...
[exec] [ant:javac] Note: Some input files use unchecked or unsafe operations.
[exec] [ant:javac] Note: Recompile with -Xlint:unchecked for details.
[exec] [ant:javac] 20 warnings
[exec] :ant-aspectj
[exec] :ant-jar
[exec] :ant-native_prepare
[sudo] password for ecf:
[exec]
[exec] FAILURE: Build failed with an exception.
[exec]
[exec] * What went wrong:
[exec] Execution failed for task ':ant-native'.
[exec] > exec returned: 2
[exec]
[exec] * Try:
[exec] Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more
log output.
[exec] :ant-native FAILED
[exec]
[exec] BUILD FAILED
[exec]
[exec] Total time: 34.987 secs
```

Note: I do not have the post-build set up to be password-less on my system.

If I run gradlew all or ant all without the post-build, it completes successfully. Something seems to be going wrong with the sudo password prompt. Although I see the password prompt in the console output, it does not pause to allow me to type in my password, just outputs the prompt and fails immediately.

#52 - 01/27/2017 11:31 AM - Greg Shah

The weird thing (with this sudo password prompt issue) is that I don't see this problem. My system is not setup for passwordless sudo. I get prompted and the build pauses just like before. When I authenticate, the post-build runs and it all works fine.

I'm not sure what is different between Eric's system and mine.

#53 - 01/28/2017 01:25 PM - Greg Shah

I've recreated this failure on my system. In my case, I was installing to a directory that did not exist. After I created that directory the subsequent build worked.

Please note that it is possible that my use of sudo to create the /opt/spawner directory, meant that the build (which was run seconds after creating the directory) did not need to prompt for sudo. So if the problem was the prompt, this bypassed it.

Eric: Make sure that all of your parameters are correct and that the target directory exists. If it still fails, try sudo ls -l and then retry the build.

#54 - 01/28/2017 02:27 PM - Hynek Cihlar

Greg Shah wrote:

Eric: Make sure that all of your parameters are correct and that the target directory exists. If it still fails, try sudo ls -l and then retry the build.

Eric, also if you haven't tried already, please run gradlew with --info or --debug, these should give more verbose output.

#55 - 01/29/2017 12:29 PM - Greg Shah

I just tried the build (including the post-build step that needs sudo) in a mode where it needed to prompt me. When I use the correct parameters, it does prompt and it does pause for input. But the input I provide is never accepted. I type my password and then hit enter and nothing happens. I hit enter again and it accepts it but then prompts me again.

There may be a real problem here.

#56 - 01/29/2017 01:03 PM - Eric Faulhaber

I've seen exactly this problem, too. I worked around it by using sudo for something else (as Greg notes in [#2699-53](#)), just before I build.

#57 - 01/30/2017 05:01 AM - Ovidiu Maxiniuc

Greg,

I am trying to do a custom build, and I need to add a new jar to fwd, but gradle will only accept it from <https://proj.goldencode.com/artifacts/com/goldencode/>. Do I need special credentials for that?

Thanks!

#58 - 01/30/2017 06:16 AM - Greg Shah

Are you implementing a customized version of Hibernate or one of the other 3rd party jars we depend upon?

Today we have a manual process for deploying the Hibernate jar to that location. Either Eric or myself can deploy it. Since this should change very infrequently, we were hoping this is OK.

Hynek: what can be done to make local development possible for this scenario?

#59 - 01/30/2017 08:01 AM - Hynek Cihlar

Ovidiu Maxiniuc wrote:

I am trying to do a custom build, and I need to add a new jar to fwd, but gradle will only accept it from <https://proj.goldencode.com/artifacts/com/goldencode/>.

Ovidiu,

if you only need to run FWD with your custom jar, just replace it in \$P2J_HOME/build/lib with your own jar. If you want to add the jar dependency in your IDE, replace the jar also in \$P2J_HOME/lib or where ever your dependencies are located.

You could also modify the build script (gradle.build) to make this work directly when the project is built:

- add the following in the repositories section

```
flatDir {  
    dirs 'lib'  
}
```

- optionally find the dependency you want to change in the dependencies section and comment it out,
- add your jar file to \$P2J_HOME/lib,
- add the following to the dependencies section (the jar file extension should not be needed):

```
p2jCompile name: 'yourjarfile'
```

We have seen the following error in some environments:

```
[exec] * What went wrong:
[exec] A problem occurred evaluating root project 'p2j'.
[exec] > Could not resolve all dependencies for configuration ':p2jCompile'.
[exec] > Could not resolve com.goldencode:p2j-hibernate-core:4.1.8.
[exec] Required by:
[exec] :p2j:unspecified
[exec] > Could not resolve com.goldencode:p2j-hibernate-core:4.1.8.
[exec] > Could not get resource
[exec] 'https://proj.goldencode.com/artifacts/com/goldencode/p2j-hibernate-core/4.1.8/p2j-hibernate-core-4.1.8.pom'.
[exec] > Could not GET 'https://proj.goldencode.com/artifacts/com/goldencode/p2j-hibernate-core/4.1.8/p2j-hibernate-core-4.1.8.pom'.
[exec] > sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
```

The failure reading the pom file is a red-herring. It is being looked up during dependency resolution, if it is not found, then gradle assumes some defaults and fetches the corresponding jar directly. We don't have the pom file there, but the jar is there. The real cause of the build failure is the certificate validation for proj.goldencode.com, for some reason the JVM process in use doesn't trust the web server certificate.

The issue is probably caused by the java installation using a back-level cacerts file. The cacerts file is a keystore that contains the list of trusted certificate authorities (CAs). The CA that we use is listed in the more recent cacerts but it must be missing from the one you have installed.

If you see this, you should check the java key store: `keytool -list -v -keystore <JAVA_HOME>/jre/lib/security/cacerts | grep DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13`.

We are using the OpenJDK 1.8.0_121 on Ubuntu. In my installation, that JDK uses `/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/security/cacerts` which is a symbolic link to `/etc/ssl/certs/java/cacerts` which is dated July 12, 2016. This version of cacerts works. If I run this:

```
keytool -list -v -keystore /etc/ssl/certs/java/cacerts | grep DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
```

Don't put in a password, just enter at the password prompt. This is what you will see if the CA is there:

```
Enter keystore password:
```

```
***** WARNING WARNING WARNING *****
* The integrity of the information stored in your keystore *
* has NOT been verified! In order to verify its integrity, *
* you must provide your keystore password.                *
***** WARNING WARNING WARNING *****
```

```
SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
```

Eugenie hit this same problem and found that he had the newer cacerts was in a file called `/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/security/cacerts.original` but it was not being used. He copied that file into `/etc/ssl/certs/java/cacerts` and then rebooted and it resolved the issue for him. The reboot was important because it unloads the gradle daemon that has some cached knowledge that can cause a problem.

Ubuntu also provides the `ca-certificates-java` package, which installs the `/etc/default/cacerts` file. On my system that file is from May 12, 2014. I'm pretty sure that it does not include the CA we need. But since it is not used in this case, I guess it should not matter. My point here is that installing the `ca-certificates-java` package probably won't solve the problem.

To see where the various cacerts files come from in your Ubuntu installation:

```
dpkg -S cacerts
```

It can also be useful to check the gradle version: `./gradlew --version` in the FWD directory, which will show something like this:

```
-----
Gradle 3.0
-----
```

```
Build time: 2016-08-15 13:15:01 UTC
Revision:   ad76ba00f59ecb287bd3c037bd25fc3df13ca558

Groovy:     2.4.7
Ant:        Apache Ant(TM) version 1.9.6 compiled on June 29 2015
JVM:        1.8.0_121 (Oracle Corporation 25.121-b13)
```

OS: Linux 4.4.0-59-generic amd64

And `./gradlew --status` which will show the status of the currently running gradle daemons:

PID	STATUS	INFO
24239	IDLE	3.0

Only Daemons for the current Gradle version are displayed. See https://docs.gradle.org/3.0/userguide/gradle_daemon.html#sec:status

#61 - 02/01/2017 12:12 PM - Eugenie Lyzenko

Strange gradlew use case:

Assume you have two different FWD source trees. And it is required to make build for both but with different options:

1. Go to the first location and run `ant all` - building is OK.

2. Then it is required to build another tree with spawn generation:

`ant all -Dpost.build=yes -Dspawn.install.folder=/opt/spawn -Dsrv.certs=~/.app/poc/deploy/server/srv-certs.store`

The password to copy spawn files is **not** requesting. The building is just failing with:

```
[exec] :ant-native_prepare
[sudo] password for evl:
[exec] :ant-native FAILED
[exec]
[exec] FAILURE: Build failed with an exception.
[exec]
[exec] * What went wrong:
[exec] Execution failed for task ':ant-native'.
[exec] > exec returned: 2
[exec]
[exec] * Try:
[exec] Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more
log output.
[exec]
[exec] BUILD FAILED
[exec]
[exec] Total time: 1 mins 2.428 secs
```

BUILD FAILED

Solution - kill gradlew daemon and re-run point 2 for another tree - building become OK.

The diagnostics says the native code building issue but it is not a true. gradlew do not ask the password. And it is not obvious for what to do to continue building.

#62 - 02/06/2017 09:45 AM - Eric Faulhaber

- File `h2_synchronization_fix_20160816a.patch` added

Hynek, I see that we are using the latest, public H2 release (1.4.193) in the gradle build. Greg just reminded me that I made a change to H2 1.4.192 last August, but I neglected to store the patch in bzt at that time. I haven't fully tested FWD with 1.4.193, though I don't have any reason to believe it wouldn't work, since you've already run standard regression testing with it. So, I'm happy to stay at the later level. However, we do need to apply the patch. This means we also need to provide the customized jar with our other binary artifacts.

I am attaching the patch here. Please integrate it with the build and let me know if you have any questions or issues. Hopefully, the two affected files haven't changed to the point where applying the patch causes a problem. Sorry for the earlier oversight.

#63 - 02/06/2017 09:48 AM - Eric Faulhaber

- File `deleted (h2_synchronization_fix_20160816a.patch)`

#64 - 02/06/2017 09:49 AM - Eric Faulhaber

- File `h2_synchronization_fix_20160816a.patch` added

Eric Faulhaber wrote:

I am attaching the patch here.

Sorry, wrong one. That had some debug code I meant to remove. Please use this one instead.

#65 - 02/08/2017 06:31 PM - Hynek Cihlar

The problem with the failing native task requesting sudo input is caused by the gradle daemon not handling input redirection correctly on the tasks executed from the ant script. A quick workaround is to use the gradlew parameter `--no-daemon` when executing the task native. I am fixing this by moving the native task body from ant to gradle script.

#66 - 02/09/2017 08:47 AM - Greg Shah

Is there a way to default to `--no-daemon` mode in the build.gradle? It seems like the daemon causes more problems than it solves. Personally, I would prefer it never to run unless I ask for it.

#67 - 02/09/2017 08:53 AM - Hynek Cihlar

Greg Shah wrote:

Is there a way to default to `--no-daemon` mode in the build.gradle? It seems like the daemon causes more problems than it solves. Personally, I would prefer it never to run unless I ask for it.

This is what I am trying to figure out. There is another option which I want to enforce, `--console plain`, which fixes the sudo prompt being overwritten by the gradle status output messages. There are some options to enforce this functionality, but it involves creating config files in the user's home directory. If possible, I'd like to have something more streamlined.

#68 - 02/09/2017 08:55 AM - Greg Shah

If possible, I'd like to have something more streamlined.

Agreed. It seems strange that gradle wouldn't provide a programmable mechanism from the build script since one major value of gradle is its programmability.

#69 - 02/09/2017 10:23 AM - Hynek Cihlar

Please see the attached diff, it fixes the native build task.

The change turns the gradle daemon off by default and forces the plain console (gradle command line argument `--console=plain`). While the daemon mode can be overridden on the command line with `--daemon`, the console mode cannot. Also since the console mode is hard coded in the gradlew wrapper script, future updates to the script will require the change to be merged. I wish there was a better way to force the console mode.

#70 - 02/09/2017 10:24 AM - Hynek Cihlar

- *File build.diff added*

Hynek Cihlar wrote:

Please see the attached diff, it fixes the native build task.

And the attachment.

#71 - 02/09/2017 10:25 AM - Hynek Cihlar

- *File deleted (build.diff)*

#72 - 02/09/2017 10:26 AM - Hynek Cihlar

- *File build.diff added*

A minor cleanup.

#73 - 02/09/2017 11:37 AM - Greg Shah

I'm OK with the approach. You made the right choice to take the merging rather than forcing the user to solve the problem on the command line (which is very error prone).

#74 - 02/09/2017 12:08 PM - Hynek Cihlar

- File `RegularTable.java.rej` added

Eric Faulhaber wrote:

Eric Faulhaber wrote:

I am attaching the patch here.

Sorry, wrong one. That had some debug code I meant to remove. Please use this one instead.

Eric, I am getting one rejection on the patch. This is what I did:

- cloned the h2 repo: `git clone https://github.com/h2database/h2database.git`
- `cd h2database && patch -p 1 < h2_synchronization_fix_20160816a.patch`

The output:

```
hc@gcdntb:~/gcd/h2/h2database$ patch -p 1 < h2_synchronization_fix_20160816a.patch
patching file h2/src/main/org/h2/table/RegularTable.java
Hunk #3 FAILED at 672.
1 out of 4 hunks FAILED -- saving rejects to file h2/src/main/org/h2/table/RegularTable.java.rej
```

The rejection file is attached.

#75 - 02/09/2017 02:23 PM - Hynek Cihlar

The sudo prompt fix of the native build task committed to 3209e.

The check in also contains the ability to provide local jar dependencies. This is to allow to build the patched libraries (hibernate-core and h2) and use them in case the goldencode repository would not be available. The local jars must be placed in the directory `p2j_home/local-repo` following the standard maven repository directory structure, no need to create the metadata files (maven pom).

#76 - 02/10/2017 06:23 AM - Stanislav Lomany

Eric, as Hynek noticed, update 1.4.192 -> 1.4.193 has minor sync-related overlap with your changes. What version should we use?

#77 - 02/10/2017 07:39 AM - Eric Faulhaber

Please use 1.4.192 for the time being, until I have had a chance to review/merge/test with the latest H2 code base. Even if we can drop my patches with a later version of H2, or get an adjusted version of the patches accepted back into H2, we should have the patch mechanism working with the FWD build. I don't expect this to be the last patch we ever will make to H2. Thanks.

#78 - 02/10/2017 07:42 AM - Hynek Cihlar

Eric Faulhaber wrote:

Please use 1.4.192 for the time being.

OK, no problem. If you agree I will rename the artifact group and name so that it is clear it comes with GCD changes. Similarly as we currently do with hibernate-core.

#79 - 02/10/2017 07:58 AM - Eric Faulhaber

Hynek Cihlar wrote:

Eric Faulhaber wrote:

Please use 1.4.192 for the time being.

OK, no problem. If you agree I will rename the artifact group and name so that it is clear it comes with GCD changes. Similarly as we currently do with hibernate-core.

Sounds good.

#80 - 02/10/2017 10:33 AM - Hynek Cihlar

The attached zip file contains the patched h2 version 1.4.192 with the directory structure expected by gradle.

Greg or Eric, please upload the content of the attached zip file to <https://proj.goldencode.com/artifacts/>.

Also I am wondering whether we should rename p2j-h2 to fwd-h2 (and also p2j-hibernate-core to fwd-hibernate-core).

#81 - 02/10/2017 11:16 AM - Greg Shah

Also I am wondering whether we should rename p2j-h2 to fwd-h2 (and also p2j-hibernate-core to fwd-hibernate-core).

Good idea, please do.

#82 - 02/10/2017 11:41 AM - Hynek Cihlar

- File *fwd-hibernate-core.zip* added

- File *fwd-h2.zip* added

Greg, please upload the contents of the attached zip files to <https://proj.goldencode.com/artifacts/>.

#83 - 02/10/2017 01:30 PM - Greg Shah

I've put the files in the artifacts directory. We can defer the removal of the old p2j-hibernate-core until we are sure it is no longer needed.

Let me know if there are any issues.

#84 - 02/16/2017 07:47 AM - Constantin Asofiei

Hynek: building FWD is dependent upon bzt - this should not be a requirement if the person building it doesn't have access to our bzt repo...

Greg: there are the NCURSES patch files, I've linked them in the doc as https://proj.goldencode.com/artifacts/ncurses_curses_h_in_20060828.patch and https://proj.goldencode.com/artifacts/ncurses_lib_getch_c_v5.7_20090512.patch - can you upload them there? Or we re-add them to the FWD repo?

#85 - 02/16/2017 07:53 AM - Hynek Cihlar

Constantin Asofiei wrote:

Hynek: building FWD is dependent upon bzt - this should not be a requirement if the person building it doesn't have access to our bzt repo...

Since trunk revision 11140 the build should support environment without GCD's repository or bzt installed. Gradle build script uses bzt to retrieve the branch, revision and repository name which are then used in assembling the FWD version string. If for whatever reason bzt lookup fails the script uses version.properties solely.

#86 - 02/16/2017 07:59 AM - Constantin Asofiei

Hynek Cihlar wrote:

Constantin Asofiei wrote:

Hynek: building FWD is dependent upon bzd - this should not be a requirement if the person building it doesn't have access to our bzd repo...

Since trunk revision 11140 the build should support environment without GCD's repository or bzd installed. Gradle build script uses bzd to retrieve the branch, revision and repository name which are then used in assembling the FWD version string. If for whatever reason bzd lookup fails the script uses version.properties solely.

I understand, but even if version.properties exists, build fails as it still uses bzd for some reason...

```
fwd@fwd-demo:/opt/fwd$ ant jar
Buildfile: /opt/fwd/build.xml
```

```
check-gradle:
```

```
g-jar:
```

```
[exec] The command attribute is deprecated.
[exec] Please use the executable attribute and nested arg elements.
[exec] Starting a Gradle Daemon (subsequent builds will be faster)
[exec] :buildVersion
[exec] Fetching repository name and branch from bzd info...
[exec] :buildVersion FAILED
[exec]
[exec] BUILD FAILED
[exec]
[exec] Total time: 4.629 secs
[exec]
[exec] FAILURE: Build failed with an exception.
[exec]
[exec] * What went wrong:
[exec] Execution failed for task ':buildVersion'.
[exec] > A problem occurred starting process 'command 'bzd''
[exec]
[exec] * Try:
[exec] Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more
log output.
```

```
BUILD FAILED
```

```
/opt/fwd/build.xml:1188: exec returned: 1
```

#87 - 02/16/2017 08:01 AM - Constantin Asofiei

Ah, I get it: version.properties has these commented values:

```
#repo, branch and rev are attempted to be read from code repository if not defined here
#repo=p2jd
#branch=trunk
#rev=123456
```

And build script wants to determine one of these properties I assume.

#88 - 02/16/2017 08:03 AM - Hynek Cihlar

Constantin Asofiei wrote:

Ah, I get it: version.properties has these commented values:
[...]
And build script wants to determine one of these properties I assume.

It still should succeed, giving warnings as hints. Can you run the script with --info and --debug?

#89 - 02/16/2017 08:06 AM - Constantin Asofiei

- File fwd_bzr_error.txt added

Hynek Cihlar wrote:

Constantin Asofiei wrote:

Ah, I get it: version.properties has these commented values:
[...]
And build script wants to determine one of these properties I assume.

It still should succeed, giving warnings as hints. Can you run the script with --info and --debug?

See attached for the log.

#90 - 02/16/2017 09:05 AM - Hynek Cihlar

Constantin Asofiei wrote:

It still should succeed, giving warnings as hints. Can you run the script with --info and --debug?

See attached for the log.

Fix checked in to 3209e.

#91 - 02/16/2017 09:33 AM - Greg Shah

Greg: there are the NCURSES patch files, I've linked them in the doc as https://proj.goldencode.com/artifacts/ncurses_curses_h_in_20060828.patch and https://proj.goldencode.com/artifacts/ncurses_lib_getch_c_v5.7_20090512.patch - can you upload them there? Or we re-add them to the FWD repo?

I think they should be in the downloads/ncurses/ directory on proj. I have uploaded them there. I'm put all the patches there too:

```
ncurses_curses_h_in_20060828.patch
ncurses_curses_h_in_v5.9_20160219.solaris.patch
ncurses_lib_getch_c_20060828.patch
ncurses_lib_getch_c_v5.7_20090512.patch
ncurses_lib_getch_c_v5.9_20160219.solaris.patch
```

#92 - 02/16/2017 09:35 AM - Hynek Cihlar

Greg Shah wrote:

I think they should be in the downloads/ncurses/ directory on proj. I have uploaded them there. I'm put all the patches there too:

[...]

I already put them to https://proj.goldencode.com/projects/p2j/wiki/Software_Dependencies.

#93 - 02/20/2017 09:11 AM - Hynek Cihlar

Below is a conversation related to a build failure on Windows when bzt was not installed. Note that a fix has been committed to 3209e revision 11211 by Eugenie.

On 19.2.2017 21:18, Eugenie V. Lyzenko wrote:

Committed as 11211 in 3209e repo.

On 19.2.2017 20:41, Eugenie V. Lyzenko wrote:

Hynek,

yes, those warnings are expected when the version components cannot be resolved from bzt and when they are not defined in version.properties. Can you please commit the patch with your changes?

I do not understand what "my changes" you mean. I've made no changes over your patch. Clarify please. Do you mean the patch you have sent me?

Thanks,
Hynek

Regards,
Eugenie.

On 19.2.2017 20:18, Eugenie V. Lyzenko wrote:

Hynek,

ditch the previous patch and use this one.

Confirm your patch is working with message:

```
-----  
:buildVersion  
WARN: Unable to resolve repository name. Make sure you build from a supported bz  
r repository location or the property repo is defined in version.properties.  
WARN: Unable to resolve branch name. Make sure you build from a supported bzt re  
pository location or the property branch is defined in version.properties.  
WARN: Unable to resolve revision number. Make sure you build from a supported bz  
r repository location or the property rev is defined in version.properties.  
Product version: 3.0.0_undefined_undefined_undefined  
-----
```

But I guess it is normal because I do not have BZR installed on this Windows machine.

Thanks,
Hynek

Regards,
Eugenie.

On 18.2.2017 20:12, Hynek Cihlar wrote:

Eugenie,

please apply the attached patch to build.gradle and test in your Windows environment. Unfortunately I don't have a suitable Windows box handy, so I can't test it myself.

Thanks,
Hynek

On 18.2.2017 00:27, Eric Faulhaber wrote:

Eugenie,

Can we have the option to use old legacy(pure ant based) building?

No, we moved to gradle for the dependency management, as we need to not bundle the third party libraries with FWD for legal reasons.

As you noted from the error output, it appears the missing which.exe is the problem. I'm no gradle expert. I think Hynek will have to consider what a good solution for this could be on Windows. Is there a similar command to 'which'?

Thanks,
Eric

On 02/17/2017 06:14 PM, Eugenie V. Lyzenko wrote:

Hynek,

I'm testing server/client installations on Windows.

Have you tried to build FWD project on Windows system with recent build related changes?

I can not build neither with ant all nor with gradlew.bat all. The reason is:

...

:buildVersion FAILED

FAILURE: Build failed with an exception.

- Where:
Build file 'C:\opt\fwd\build.gradle' line: 292
- What went wrong:
Execution failed for task ':buildVersion'.

Cannot run program "which": CreateProcess error=2, The system cannot find the

file specified

- Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED

Total time: 8.611 secs

...

Do I need to install additional software? Bzr? Other command line tools? I have MinGW installed and there is no which.exe there.

We need to have Windows binaries to run FWD server/client on Windows. Either we using cross compiling in Linux to produce Windows binaries or we need to have working Windows build system to do this. What is the current plan for building FWD binaries within new gradle based build system? Can we have the option to use old legacy(pure ant based) building?

Regards,
Eugenie.

#94 - 02/20/2017 09:13 AM - Eugenie Lyzenko

- *File build.gradle.patch added*

The fix body here too.

#95 - 03/08/2017 05:07 AM - Hynek Cihlar

2699c rebased against trunk 11141.

#96 - 03/08/2017 03:50 PM - Hynek Cihlar

2699c revision 11148 contains the final version of the distribution build target and related changes.

Actually there are two new targets, distribution which depends on cleanDistribution. The targets are not triggered by any other build targets, they must be executed manually (i.e. ./gradlew distribution).

When distribution is executed, it will create the directories dist/convert, dist/client and dist/server with dependencies for convert, client and server. I have tested the dependencies with all FWD deployment configurations I could come up with (simple server, customers' servers, ChUI and GUI clients, Web clients, conversion, conversion reports).

Please review.

#97 - 03/13/2017 06:52 AM - Greg Shah

Code Review Task branch 2699c Revision 11148

I'm OK with the changes.

Constantin: Please review.

Have you used this to build FWD for ChUI conversion and regression testing?

#98 - 03/13/2017 06:53 AM - Hynek Cihlar

Greg Shah wrote:

Code Review Task branch 2699c Revision 11148

I'm OK with the changes.

Constantin: Please review.

Have you used this to build FWD for ChUI conversion and regression testing?

Yes, I did the full regression test and it passed.

#99 - 03/13/2017 06:56 AM - Greg Shah

If Constantin has no concerns, then please merge this to trunk.

#100 - 03/13/2017 07:58 AM - Constantin Asofiei

Greg Shah wrote:

Code Review Task branch 2699c Revision 11148

I'm OK with the changes.

Constantin: Please review.

Hynek, two issues:

- Main-Class: com.goldencode.p2j.Version is missing from p2j.mf.template
- I thought we removed the where dependency for Windows?

#101 - 03/13/2017 09:03 AM - Constantin Asofiei

Hynek, one more thing: the distribution folders are missing the p2j.jar file - is this by intent?

#102 - 03/13/2017 09:17 AM - Hynek Cihlar

Constantin Asofiei wrote:

Hynek, two issues:

- Main-Class: com.goldencode.p2j.Version is missing from p2j.mf.template
- I thought we removed the where dependency for Windows?

A merging problem, thanks for catching this. Fix is being checked in.

#103 - 03/13/2017 09:34 AM - Hynek Cihlar

Constantin Asofiei wrote:

Hynek, one more thing: the distribution folders are missing the p2j.jar file - is this by intent?

This is by intention.

#104 - 03/13/2017 10:14 AM - Hynek Cihlar

2699c merged to trunk as rev 11142 and archived.

#105 - 04/06/2017 01:14 PM - Hynek Cihlar

I created new task createLocalRepo. It fetches all build and runtime dependencies and builds local artifact repository in fwd-home/local-repo. The goal of this task is to provide all dependencies for offline work and for backup.

Also copyright year was updated in javadoc task.

Please review 2699d, revision 11147.

#106 - 04/17/2017 11:38 AM - Greg Shah

Code Review Task Branch 2699d Revision 11147

I'm fine with the changes.

Did you test this on both Linux and Windows?

#107 - 04/17/2017 12:01 PM - Hynek Cihlar

Greg Shah wrote:

Code Review Task Branch 2699d Revision 11147

I'm fine with the changes.

Did you test this on both Linux and Windows?

Only on Linux.

I just tested on Windows and it fails deleting the cache dir. Windows locks several jar files needed by the build itself (like Ant jars). I am working on a workaround.

#108 - 04/17/2017 01:47 PM - Hynek Cihlar

Hynek Cihlar wrote:

I just tested on Windows and it fails deleting the cache dir. Windows locks several jar files needed by the build itself (like Ant jars). I am working on a workaround.

Fixed in 2699d. Please review.

#109 - 04/17/2017 04:08 PM - Greg Shah

Code Review Task Branch 2699d Revision 11148

I'm fine with the changes.

Please merge 2699d to trunk.

#110 - 04/17/2017 05:11 PM - Hynek Cihlar

2699d merged to trunk revision 11147 and archived.

#111 - 04/25/2017 06:22 PM - Eric Faulhaber

Hynek, we soon will need to add a handful of JavaScript framework dependencies to the build for [#1514](#). I'm not sure which ones yet. Some quick googling suggests this is not necessarily a standard activity with gradle, so I just wanted to give you a heads up on this pending requirement, so you can begin thinking about how we could integrate this.

#112 - 04/26/2017 07:34 AM - Hynek Cihlar

Eric Faulhaber wrote:

Hynek, we soon will need to add a handful of JavaScript framework dependencies to the build for [#1514](#). I'm not sure which ones yet. Some quick googling suggests this is not necessarily a standard activity with gradle, so I just wanted to give you a heads up on this pending requirement, so you can begin thinking about how we could integrate this.

Eric, my experiences with web development and web frameworks are pretty limited. But AFAIK the web build management tends towards nodejs technologies and npm.

The situation with JavaScript is a bit more complicated than with Java let's say. The build process is usually more complex (this especially holds true with the modern JS frameworks) as there are additional steps. Besides the build and dependency management tasks, the dev stacks today have to handle things like JS modularization, transpiling, minification, bundling, and awful lot of other things mostly to overcome the JS limitations. But the features go beyond just the build workflows and offer others like testing, devel web servers, client debugging, source code change tracking and automatic compilation and application of the changes in the target app. If you have time for a fun read, this article describes it really well (it is not only funny but it can help you understand the complexity of modern web apps)
<https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f>.

So the obvious option, depending on the JS frameworks used, would be to use nodejs/npm. The less obvious option would be to use Gradle with a JS plugin. The advantages for the nodejs way are a substantially greater community and flexibility, the disadvantage of nodejs is the added complexity to our build process and its flexibility (again) of so many ways to achieve the expected results.

My recommendation is to try the Gradle + plugin JS way, if there exists a stable enough plugin that would give us all the needed features.

#113 - 04/26/2017 01:44 PM - Greg Shah

I want to stay away from nodejs and npm.

I think at this point we will just manually package the JS resources into a jar file that we will host in our artifacts area. This can easily be pulled by gradle without any plugins.

Since we won't be just randomly changing JS frameworks or versions, the effort to do this is minimal and has few consequences.

#114 - 04/26/2017 04:15 PM - Hynek Cihlar

Greg Shah wrote:

I want to stay away from nodejs and npm.

I think at this point we will just manually package the JS resources into a jar file that we will host in our artifacts area. This can easily be pulled by gradle without any plugins.

Since we won't be just randomly changing JS frameworks or versions, the effort to do this is minimal and has few consequences.

While this is fine for deploying the final product it may not be optimal for development where the app version changes often.

#115 - 09/07/2017 12:39 PM - Greg Shah

Before trunk revision 11157, if the createLocalRepo target is executed on a Linux system which does not have . in the PATH, the target will fail like this:

Execution failed for task ':createLocalRepo'.

A problem occurred starting process 'command 'gradlew'

The temporary solution is to run the target using PATH=\$PATH:./gradlew createLocalRepo. Starting with revision 11157, this is no longer necessary.

#116 - 07/11/2022 11:40 AM - Greg Shah

- Related to Feature #4645: Migrate Ant build logic into Gradle added

Files

p2j-hibernate-core.zip	3.91 MB	01/09/2017	Hynek Cihlar
h2_synchronization_fix_20160816a.patch	5.76 KB	02/06/2017	Eric Faulhaber
build.diff	1.64 KB	02/09/2017	Hynek Cihlar
RegularTable.java.rej	5.63 KB	02/09/2017	Hynek Cihlar
fwd-h2.zip	1.66 MB	02/10/2017	Hynek Cihlar
fwd-hibernate-core.zip	3.91 MB	02/10/2017	Hynek Cihlar
fwd_bzr_error.txt	754 KB	02/16/2017	Constantin Asofiei
build.gradle.patch	679 Bytes	02/20/2017	Eugenie Lyzenko