Community

yegor256.com

27 Comments • Created a year ago

Fluent Java Http Client

I created a simple fluent Java HTTP client to make things easier with HTTP interactions

(yegor256.com)

27 Comments

Recommend Share Sort by Newest

Join the discussion…

Riccardo Muzzì • 4 months ago

What I like about this:
- Slick and easy syntax

What I don't like about this, and the reason I won't use this:
- Too many dependencies that are forced to be included in my classpath. Remember that when you write a library you are forcing everyone who use it, to import your dependencies in classpath, the result could be a "dependency hell" project with many problems like library versions conflict etc...
A library should be dependency free or as close as possible.

Here some good examples:

https://github.com/jOOQ/jOOQ/b...
https://github.com/lviggiano/o...
https://github.com/revetkn/res...

1 • Reply • Share ›

Yegor Bugayenko author ↪ Riccardo Muzzì • 4 months ago

That's a good point, but on the other hand, why dependency management was invented in the first place? I think a project with multiple dependencies is a good practice, since it is more flexible and light-weight. Don't you think that this is the purpose and the goal of dependency management?

• Reply • Share ›

Riccardo Muzzì ↪ Yegor Bugayenko • 4 months ago

Wait, don't get me wrong: One thing is an application, another thing is a library.
Of course there is nothing wrong for an application to have multiple dependencies, but for a library the point is different because a library is always used as a dependency. A dependency management tool like Maven surely helps a lot, but it cannot solve every conflict problem. I'll make you a few examles:

I'm working on Java 8 right now, as you probably know, libraries like Guava or LambdaJ are fairly useless because of the native support for Lambda expression and the Stream API. So if I'm using jcab, I am forced to import 1,5MB of Guava in my .war package.. for nothing!

But there are more, what if I'm using a newer version of Guava where a method X(); jcab is calling has been removed and replaced by another method Y(); I'm calling in my code? Even worse, what if method Y(); is called by another library I'm using?.... And that's an example for just one dependency!

Maven can do nothing to solve situations like these because you can only have 1 version of Guava in your application classpath.

Conclusion: To avoid this, the only thing a library programmer can do is to avoid as much as possible to have external dependencies.. And the only thing an application programmer can do is to use as much as possible libraries that are (almost) free of external dependencies.

Sorry for the long reply :)

• Reply • Share ›

Yegor Bugayenko author ↪ Riccardo Muzzì • 4 months ago

You're right, in a practical and short-term sense. And you're wrong in a long-term strategic sense. When libraries are "dirty" and tightly coupled the best way is to remove any connections between them and treat them as self-sufficient black boxes. This approach will give immediate results. But we all start thinking like this, we'll lose the idea of "composability", initially introduced by JAR packages and Maven repository. We'll just get back to C world where everything is a black box and you can't compose libraries from libraries. Don't you think?

• Reply • Share ›

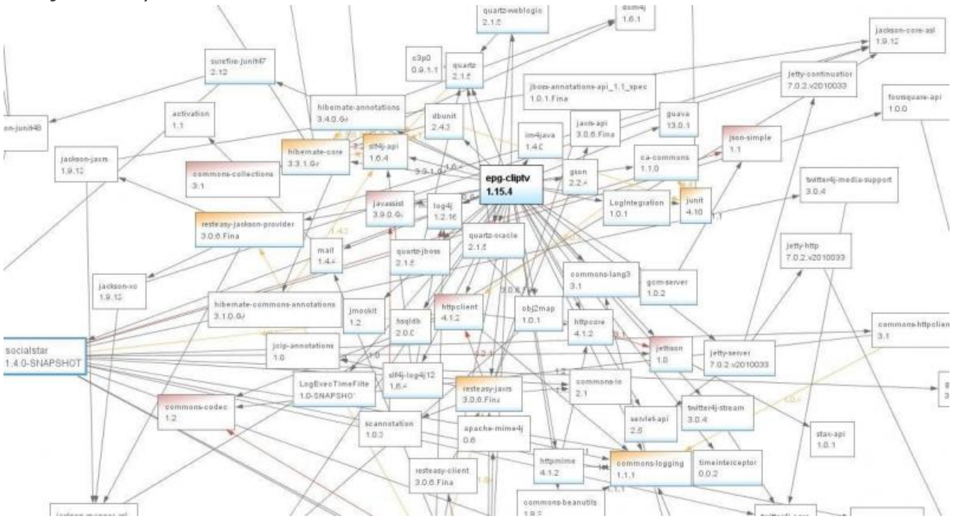
Riccardo Muzzì ↪ Yegor Bugayenko • 4 months ago

Software architecture & software engineering are not exact sciences, so it's not a universal rule, but just a good practice. But there are reason why it is considered a good practice:

The attached image refer to an integrated server I've worked in the past. All the orange and the red one are library in conflict. Try to

migrate this to a newer version of the web container you are using (in this case JBoss...) and see how many problems and compilation issues you'll get :) I spent about 3 weeks for that migration.
While migrating another web app with a much less deeper dependency graph was 1 day of work.
Note that due some nasty tightly coupled library I have 4, i repeat FOUR different libraries in my classpath for JSON parsing: Jettison, Jackson, json-simple and GSON while the only one I'm actually using is Jackson. This is insane! And this is what I'm talking about...
Furthermore migrating just a single library to a newer version (maybe because of a new function that you need), when that library is tightly coupled with other libraries that you are using in your code can be a very painful operation.

In conclusion I think that a library can have external dependencies only when they are actually needed for the user of that library (e.g. it's fine for jOOQ to have JPA as a dependency). Or when the external dependency can ensure a good backward compatibility (e.g. Log4J, slf4j... etc.).



^ | v • Reply • Share ›

Yegor Bugayenko author ➦ Riccardo Muzzì • 4 months ago

What you're saying makes perfect sense and I'm seeing exactly the same problems in every project I'm working with. However, I'm suggesting you to look at this from a different angle. I'm saying that the root cause of this dependency management nightmare is someone's stupidity, but not the dependency management idea in itself. Creators of all that libraries just don't know how to design them correctly, so that you can compose them in your application.

You're saying that you have four different JSON libraries, because they are used by other four different libraries. But it's good that you see this! Thanks to dependency management this information is transparent and visible. If that four libraries would implement JSON parsing internally, you would never know who and how is parsing JSON inside your app. And you wouldn't be able to replace one broken JSON implementation with another, without replacing the dependency that is using it.

In conclusion, I think that dependency management is a brilliant idea, which is abused in most of the libraries. As a workaround, to prevent complains like this one from yourself, library designers just get rid of dependencies and implement everything inside their code. What is the result of this? No real dependency management. Instead, our libraries are **big** and **flat** and we can't control what they are composed of. Makes sense?

By the way, you can use this jcabi-http library in a single big fat JAR, like this:

```
<dependency>
  <groupid>com.jcabi</groupid>
  <artifactid>jcabi-http</artifactid>
  <version>1.10.3</version>
  <classifier>jar-with-dependencies</classifier>
</dependency>
```

• Reply • Share ›

Riccardo Muzzì Yegor Bugayenko • 4 months ago

"I'm saying that the root cause of this dependency management nightmare is someone's stupidity, but not the dependency management idea in itself." - No doubt about that.

"In conclusion, I think that dependency management is a brilliant idea, which is abused in most of the libraries." - That is exactly the point! You summarized my 2 long replies in 1 statement :-).

"What is the result of this? No real dependency management. Instead, our libraries are big and flatand we can't control what they are composed of. Makes sense?"

Well IMHO the fact that we can't control what's inside makes some sense because it's not up to the application programmer to decide what jcabi uses internally but it's up to you (and the jcabi contributors). Take RestFB (which is a dependency free library) for instance: in order to serialize Facebook Graph API json response it uses internally some Json-Simple classes (with a different package name from the original in order to avoid conflict) not exposed externally. But that is ok to me, since I just call RestFB Façade inteface as a black box and I don't care how it returns my resulting POJO that maps Facebook JSON response.
The fact that I can use a fat-jar of Jcabi, as you said, just "sweep the dirt under the carpet" since all the dependency you are using are still in my classpath and unfortunately with the same original package name and this will not solve a possible version conflict of a Jcabi external dependency.

• Reply • Share ›

Yegor Bugayenko author ➦ Riccardo Muzzì • 4 months ago

Looks like we're on the same page, conceptually. Let's meet somewhere in the middle. How about you submit issues to our issue tracker in Github, for every dependency you want us to get rid of. We'll try to do it, one by one. I think some of them are there for no good reason and can be replaced with some small custom code. Let's try?

1 • Reply • Share ›

Riccardo Muzzì ➦ Yegor Bugayenko • 4 months ago

Maybe in my spare time I can also fork and try directly to contribute in isolating some dependencies.

1 • Reply • Share ›

Yegor Bugayenko author ➦ Riccardo Muzzì • 4 months ago

That would be absolutely great!

• Reply • Share ›

Oliver Doepner • 4 months ago

I like it. :o)

• Reply • Share ›

Yegor Bugayenko author ➦ Oliver Doepner • 4 months ago

Finally you found something to like here :)

• Reply • Share ›

Oliver Doepner ➦ Yegor Bugayenko • 4 months ago

Well, I looked through some of your jcabi libraries over the last few days and like them much better than your sometimes exaggerated and over-zealous labeling of "anti-patterns" and "evil" coding practices - accompanied by simplistic or misleading code examples - in your opinion style blog posts. So I like your "here is a useful class / API and how it works" posts, but not the ones with the religious attitude.

I actually maintain a bunch of similar libraries myself - mostly still of alpha or beta status - at https://github.com/guppy4j/lib.... Interestingly my "messaging-impl" module contains an Xml class that is very similar to your XmlDocument in jcabi-xml. I did not call my module "xml-impl", because I am trying to keep the interface (I call it Tree), generic enough to also work for JSON and JsonPath expressions. If you are interested, I'd appreciate feedback:

https://github.com/guppy4j/lib...
https://github.com/guppy4j/lib...

• Reply • Share ›

Yegor Bugayenko author ➦ Oliver Doepner • 4 months ago

Without that "religious attitude" I wouldn't be able to create that useful classes/API that you (and many others) like. Principles are much more important that specific implementation of them. I understand that you don't like some of them (or all of them), but I appreciate that you read my articles and give your feedback. The principles/ideas are not written in stone, I'm interested in improving and learning.

If you're looking for a positive feedback, I would recommend to 1) get rid of setters/getters, 2) make all objects immutable, 3) use strict static analysis.

• Reply • Share ›

Oliver Doepner Yegor Bugayenko • 4 months ago

Well, maybe my criticism then boils down to: Use more realistic code examples in your opinionated blog posts, maybe directly from jcabi.

• Reply • Share ›

Oliver Doepner Yegor Bugayenko • 4 months ago

Not sure what you mean. I don't use setters. Almost all my classes are immutable, with all dependencies passed in as constructor parameters. I use a pretty strict IntelliJ inspection profile. I will actually soon add .idea folder to Git with the shared inspection profile to my parent module for easy sharing. All other subfolders of .idea will be excluded using .gitignore.

• Reply • Share ›

Yegor Bugayenko author Oliver Doepner • 4 months ago

Try qulice.com as a static analyzer, here is an article about it: http://www.yegor256.com/2014/0... Run it on your code and you'll see how much improvement can be made. Here is where I found setters/getters: https://github.com/guppy4j/lib...

I agree, more realistic code examples are always helpful. Thanks for correction.

• Reply • Share ›

Yann Banan • 5 months ago

Just wanted to say thank you! This library is exactly what I was looking for. Simple and concise! I am actually amazed/chocked by the useless verbosity and "imperativeness" of other libraries.

• Reply • Share ›

Yegor Bugayenko author Yann Banan • 5 months ago

You are very welcome :)

• Reply • Share ›

Invisible Arrow • 5 months ago

I started using this library in a new project that I've just started to work.
I'm completely amazed by how simple and easy it is to use this client compared to Apache HttpClient.
It took literally 2 statements to get the job done. One for creating the JSON string input stream, the other to create and execute the `ApacheRequest` and return the JSON response!
I loved the `FakeRequest` object as well which was very helpful for unit testing.
With Apache HttpClient, it used to take several statements to just prepare the request and several more to execute and parse the response in the desired format. Not to mention the mutable states in between :)

Currently I'm using it for APIs that return JSON responses, but I was looking at the '`Response`' interface's '`as`' method documentation: http://http.jcabi.com/apidocs-...
From what I could make out, any class implementing '`Response`' could be the result of the conversion. Is my understanding correct?
If so, shouldn't declaration of '`as`' method be restrictive of the hierarchy, meaning it can only accept classes that implement '`Response`'?
Something like: `<T extends Response> T as(Class<T> type)`
instead of the current one `<T> T as(Class<T> type)`?
1 • Reply • Share ›

Yegor Bugayenko author Invisible Arrow • 5 months ago
I'm glad you liked the library :) And it is all built from immutable objects. Regarding your suggestion, you're totally right. Please submit it as a Github issue and we'll fix the interface: https://github.com/jcabi/jcabi...
1 • Reply • Share ›

Vach • a year ago
I cant reach HttpResponse class. Where is it?
• Reply • Share ›

Yegor Bugayenko author Vach • a year ago
My bad, it's `RestResponse`, it's here: http://http.jcabi.com/apidocs-.... I'll fix the article right now. Thanks!
1 • Reply • Share ›

Vach Yegor Bugayenko • a year ago
Ok now Its failing at runtime.
java.lang.IllegalStateException: class com.jcabi.http.request.BaseRequest is not immutable, can't use it (jcabi-aspects 1.0-SNAPSHOT/822fb2d)

I had switched it to javac (1.8) and its still failing with that exception, not sure that i understand how that even possible, I just disabled aspects.
• Reply • Share ›

Yegor Bugayenko author Vach • a year ago
That looks really strange. Can you please post it to Github: https://github.com/jcabi/jcabi... Together with a full stacktrace. We'll investigate and fix (if it's a bug). We'll help in any case :)

• Reply • Share ›

More from yegor256.com

yegor256.com recently published

How to Protect a Business Idea While Outsourcing

8 Comments Recommend

yegor256.com recently published

My Favorite Software Books - Yegor Bugayenko

13 Comments Recommend

yegor256.com recently published

There Can Be Only One Primary Constructor

33 Comments Recommend