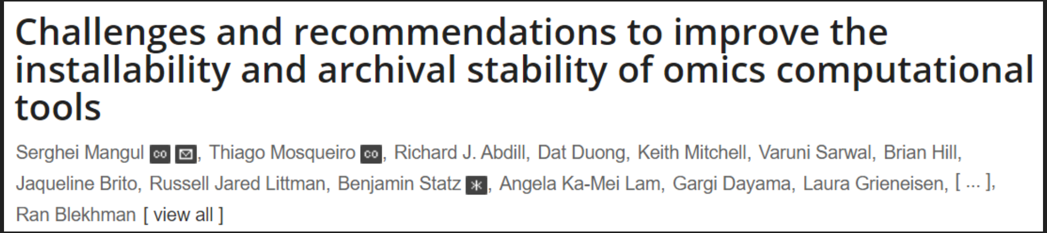
Code Review

Bioinformatics software has a problem

“27.8% of 36,702 omics software resources examined in this study are not currently accessible via the original published URLs. Among the 98 software packages selected for our installability test, 49.0% of omics tools failed our “easy-to-install” test. In addition, 27.6% of surveyed tools could not be installed because of severe problems in the implementation process.”



Challenges and recommendations to improve the installability and archival stability of omics computational tools

Serghei Mangul, Thiago Mosqueiro, Richard J. Abdill, Dat Duong, Keith Mitchell, Varuni Sarwal, Brian Hill, Jaqueline Brito, Russell Jared Littman, Benjamin Statz, Angela Ka-Mei Lam, Gargi Dayama, Laura Grieneisen, [...], Ran Blekhman [view all]

Q: How do we avoid being part of the problem?

A: Write good code

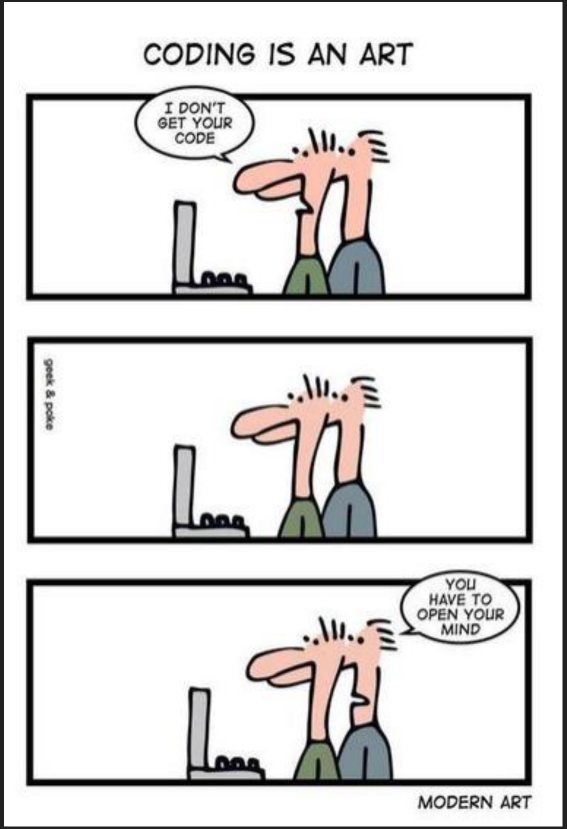
Code review helps us write reproducible, useful code!

Code review:

1. Improves the *code*

2. Improves the *coder*

3. Improves the *reviewer*



What are important things to look out for?

Three pillars of good code:

Safe From Bugs **Easy to Understand** **Ready for Change**

What are important things to look out for?

Three pillars of good code:

Safe From Bugs **Easy to Understand** **Ready for Change**

non-repetitive

descriptive names

encapsulation

whitespace

single-use variables

includes user manual

comments

follows code standard style

fail fast

What is a code review

- It is
  - Reviewing a peers code
  - Looking for potential bugs, flaws, security risks
  - Looking for areas to reduce time or space complexity
- It isn’t
  - A replacement for unit tests, QC, integration test
  - A reason to say “this code is bad someone else will fix it”
  - A time to bash on your peers

A good code review

- Helps a coder identify inconsistencies in styling
- Helps a coder write better documentation
- Teaches peers’ knowledge that you have that they might not
- Creates an environment where peers teach, help, and encourage each other
- Improves the quality of the codebase
- Takes time to review code (200 to 400 lines of code reviewed per hour, ~1 line every 10 seconds)

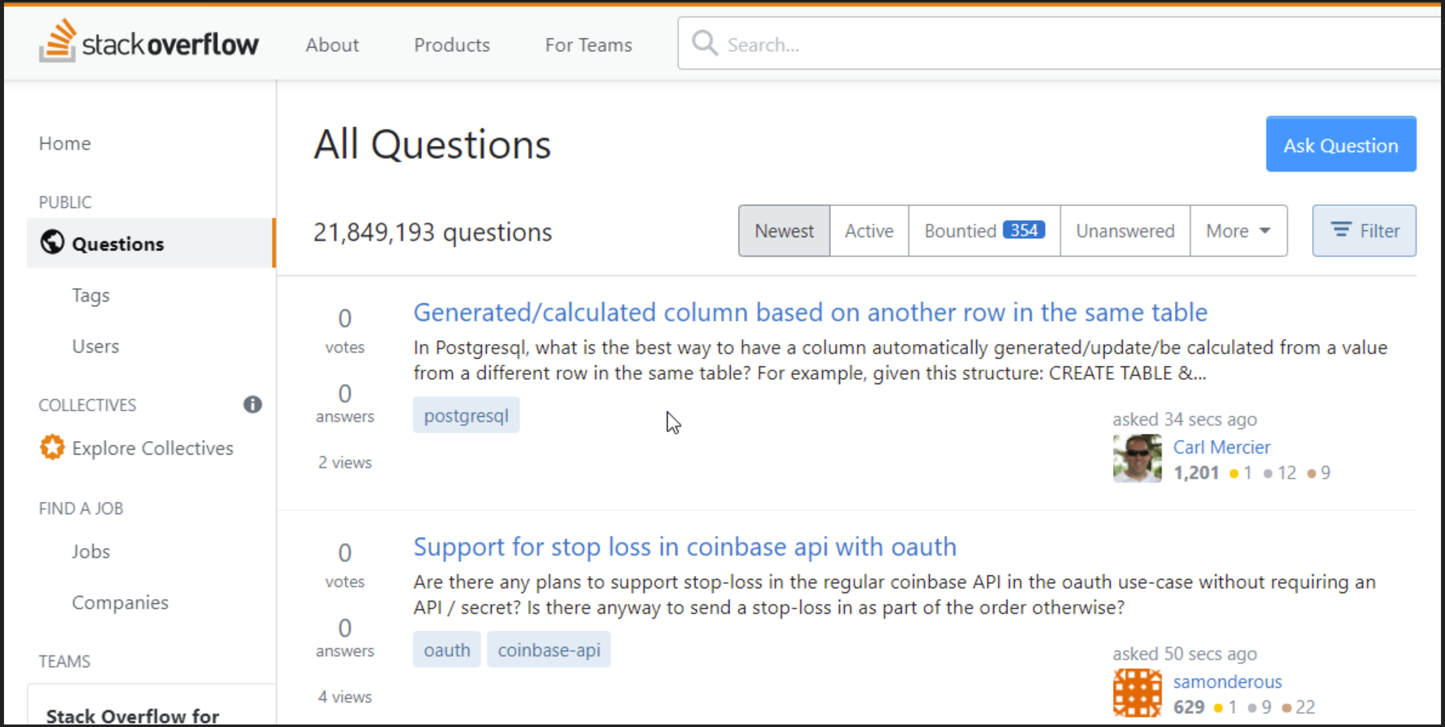
A bad code review

- Destroys a coder’s confidence
- Encourages a toxic environment of competition not cooperation
- Does not improve the quality of the code base
- Isolates peers from each other
- Is skimming over code and saying “meh good enough”

Inclusive Code Review

- ❑ Be aware of the tone of your review (avoid opinionated language)
- ❑ Offer concrete alternatives without insisting that your way is the Right Way
- ❑ Offer encouragement and positive comments (this is a learning process!)

Frequent counterpoint: stackoverflow



All Questions

21,849,193 questions

0 votes
0 answers
2 views

Generated/calculated column based on another row in the same table

In Postgresql, what is the best way to have a column automatically generated/update/be calculated from a value from a different row in the same table? For example, given this structure: CREATE TABLE &...

postgresql

asked 34 secs ago
Carl Mercier
1,201 ● 1 ● 12 ● 9

0 votes
0 answers
4 views

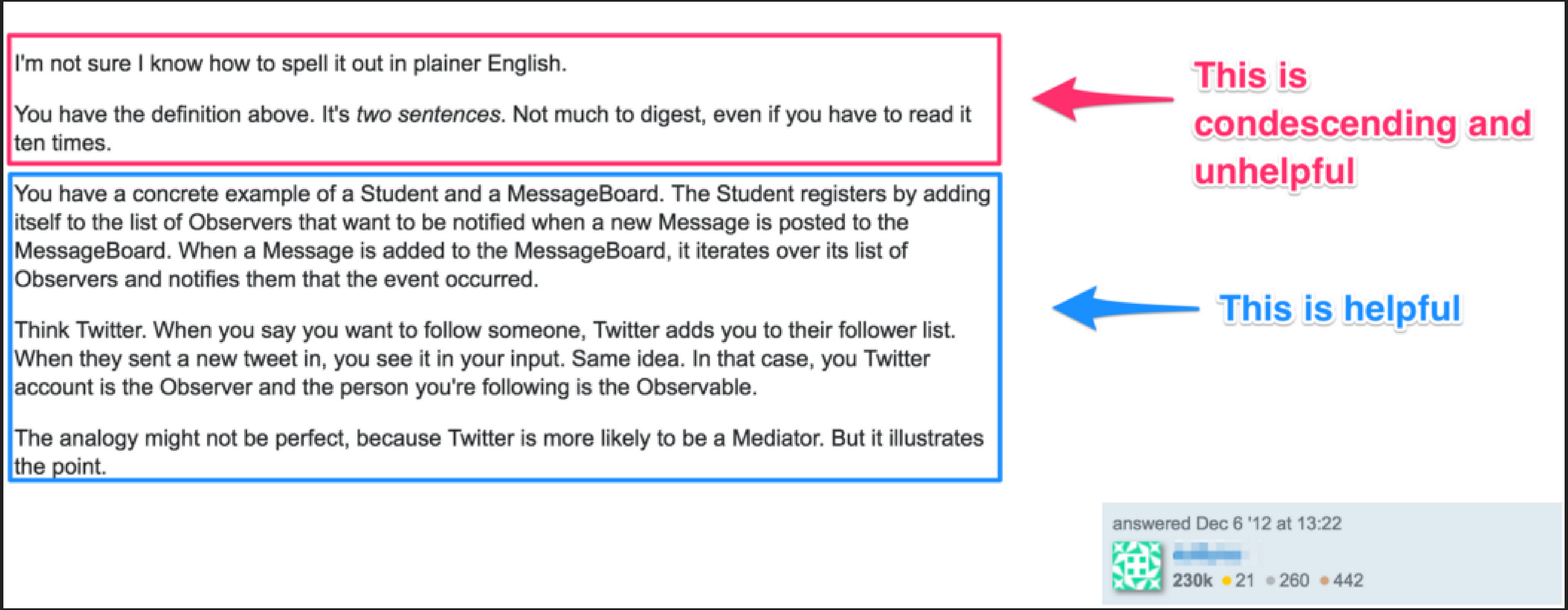
Support for stop loss in coinbase api with oauth

Are there any plans to support stop-loss in the regular coinbase API in the oauth use-case without requiring an API / secret? Is there anyway to send a stop-loss in as part of the order otherwise?

oauth
coinbase-api

asked 50 secs ago
samonderous
629 ● 1 ● 9 ● 22

Avoid opinionated language



I'm not sure I know how to spell it out in plainer English.

You have the definition above. It's *two sentences*. Not much to digest, even if you have to read it ten times.

This is condescending and unhelpful

You have a concrete example of a Student and a MessageBoard. The Student registers by adding itself to the list of Observers that want to be notified when a new Message is posted to the MessageBoard. When a Message is added to the MessageBoard, it iterates over its list of Observers and notifies them that the event occurred.

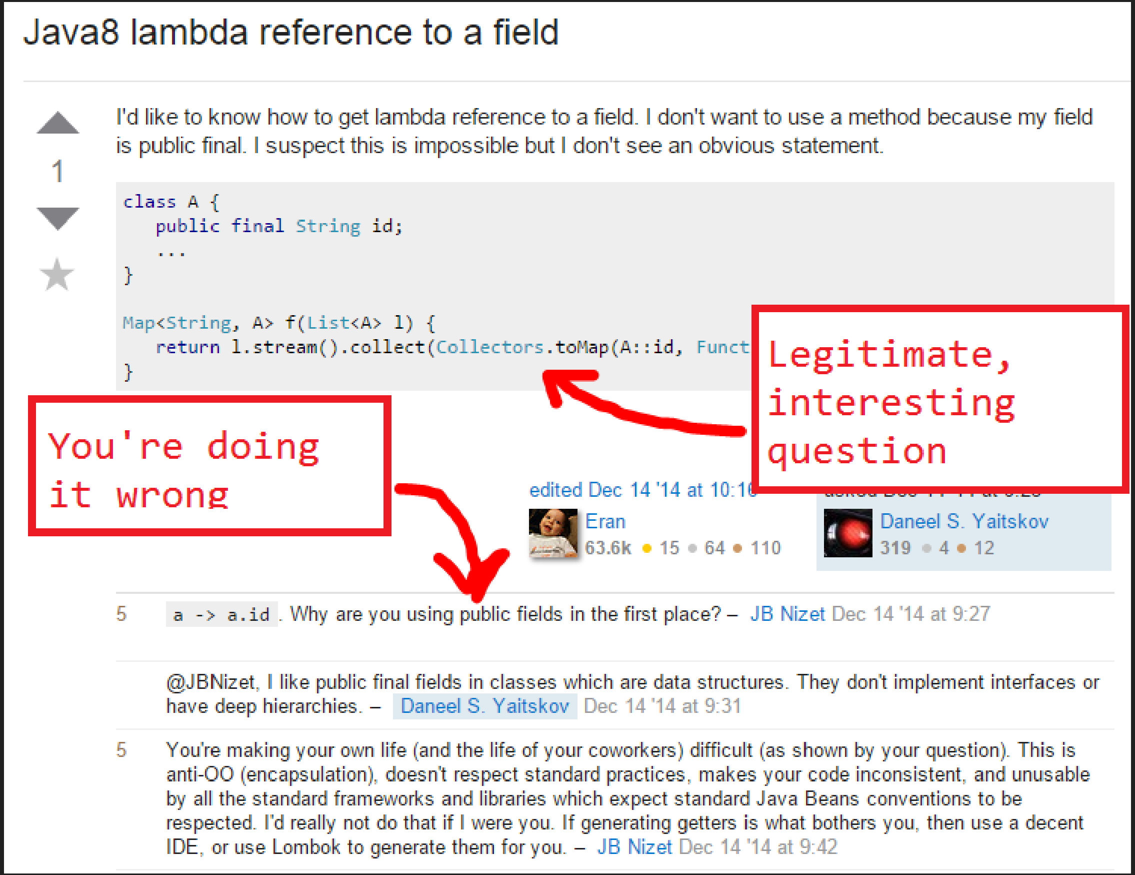
Think Twitter. When you say you want to follow someone, Twitter adds you to their follower list. When they sent a new tweet in, you see it in your input. Same idea. In that case, you Twitter account is the Observer and the person you're following is the Observable.

The analogy might not be perfect, because Twitter is more likely to be a Mediator. But it illustrates the point.

This is helpful

answered Dec 6 '12 at 13:22
[redacted]
230k ● 21 ● 260 ● 442

Don’t insist your way is the Right Way



Java8 lambda reference to a field

1

I'd like to know how to get lambda reference to a field. I don't want to use a method because my field is public final. I suspect this is impossible but I don't see an obvious statement.

```
class A {
   public final String id;
   ...
}

Map<String, A> f(List<A> l) {
   return l.stream().collect(Collectors.toMap(A::id, Funct
}
```

Legitimate, interesting question

You're doing it wrong

edited Dec 14 '14 at 10:1
Eran
63.6k ● 15 ● 64 ● 110

Daneel S. Yaitskov
319 ● 4 ● 12

5 `a -> a.id`. Why are you using public fields in the first place? – JB Nizet Dec 14 '14 at 9:27

@JBNizet, I like public final fields in classes which are data structures. They don't implement interfaces or have deep hierarchies. – Daneel S. Yaitskov Dec 14 '14 at 9:31

5 You're making your own life (and the life of your coworkers) difficult (as shown by your question). This is anti-OO (encapsulation), doesn't respect standard practices, makes your code inconsistent, and unusable by all the standard frameworks and libraries which expect standard Java Beans conventions to be respected. I'd really not do that if I were you. If generating getters is what bothers you, then use a decent IDE, or use Lombok to generate them for you. – JB Nizet Dec 14 '14 at 9:42

Offer encouragement

1. Everyone loves compliments!
2. You can note parts of the code you recommend against changing
3. You establish welcoming, open communication

HOW TO MAKE A GOOD CODE REVIEW
geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Humanizing the process -- the reviewer

Remember that it’s your peers hard work.

You will be on the receiving end of a review at some point, so help others as you would want to be helped.

Instead of making statements such as “You didn’t declare those variables”, ask a question like “I didn’t see where those variables were declared. Could you point it out”.

Allow the author of the code to help discover, don’t tear them down

Humanizing the process -- the coder

You don’t know everything so be open to suggestions

“Egoless programming”: separate your idea of self worth from your code

Any suggestion or improvement is not an attack on you, it is being helpful

Take the opportunity to learn from your peers that know things you don’t

When to do a code review

Depends on the environment

- Large organization with many feature, bugfix, or hotfix branches, code reviews should be done before any merge to production or master branches (PRs)
- After unit tests, builds, and integration tests

Coder’s responsibility (1/3)

This isn’t an excuse to write bad code. Your peers are here to *help* you, not do your work

You should make it as easy as possible for them to review

Coder’s responsibility (2/3)

- Write meaningful commit messages
  - Avoid ambiguous one liners
  - Small fixes: specific one liner
  - Large fixes: tweet or moderate email length
- Narrow down the scope
  - No one wants to review 5 files with 100 + lines of changes
  - Break it down
- Review your own code
  - Did you just write it and forget about it?
  - Look over it with your own checklist before passing it on

Coder’s responsibility (3/3)

- Make sure your code actually works
  - Run all unit tests, local builds, integration tests
  - Don’t ask for others to review broken code
- Documentation
  - Function descriptions, class descriptions, inline comments
  - Make sure anyone who looks at your code understands what your code does
  - Plus its just good practice

Reviewer’s responsibility

Don’t use this as a time to make yourself “look better” by having lots of edits

Build your peer’s confidence and ask meaningful, thoughtful questions

This is not a zero sum game. You do not gain anything by finding the most incorrect things. Quality over quantity

How to do a code review

Larger organizations may have style guides or standards. Follow those if given

For smaller projects, consistency is the goal. Make sure that the coding style remains consistent and clear.

Consider suggesting making use of the language’s official style guide (like PEP8 for python or other standards (GNU styling for C)

For all other aspects of the code base, find or create a checklist of things to review

Example checklist (1/4)

Legibility and Style

- Does the code adhere to any guidelines set out by your organization?
- Is the code easy to read and understand?
- Are tabs, curly braces, operators evenly spaced and consistent in their look and position?
- Do lines of code extend too far and go off screen?

Documentation

- Is there any?
- Are functions adequately explained and parameters outlined?
- Are there inline comments that explain what complex lines or blocks of code do?
- If the project is hosted in a repository, is there a readme that explains what the project does and how to get it up and running?

Example checklist (2/4)

Architecture

- Are blocks of reusable code put into submodules?
- Does the program have an easy to follow flow?
- Are front end and backend modules adequately separated?
- If a middle layer or message passing module exists, is it easy to find and separated from the rest of the project?

Testability

- Are there tests? If so, do the tests cover all test cases?
- Is code broken into easy to test modules?

Example checklist (3/4)

Maintainability

- Is the code easy to update and follow along?
- Is it backwards compatible if it needs to be?
- Are things hard coded where constants should be used instead?
- Going along with modularity, is the code too tightly coupled with another module?
- Can modules be easily swapped without re-writing the entire code base?

Performance

- Does the module do what it is supposed to do?
- Do the modules do what they do in a reasonable amount of time?
- Are modules too time or resource intensive? Could the problem be simplified?
- If you had to write the module, what would you do differently and why?

Example checklist (4/4)

Security

- Do you require too much user information?
- Do you carry around user data into modules that do not need it?
- Think like an adversary. If you wanted to steal data from this code, could you do it easily? How would you do it?

Conclusion

If done correctly, code reviews:

- Distributes knowledge
- Improves the code base
- Build a culture that values teamwork