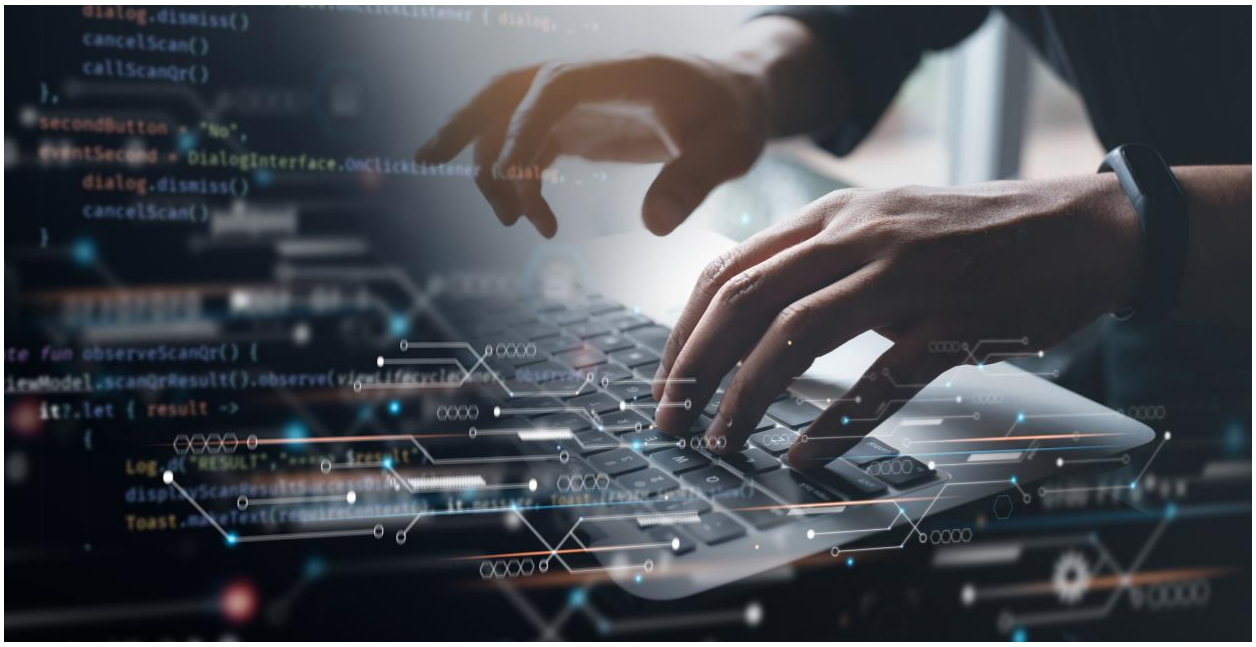
Apex Class and Method Management with Snapshot



Managing Apex Classes and Methods is an important part of keeping any Salesforce org running smoothly. Apex Classes and related metadata assets such as Apex Pages, Apex Components, Apex Triggers, Aura Definition Bundles, and Lightning Web Components can be used to write sophisticated applications on the Salesforce Platform. Keeping these assets operating correctly reduces technical debt and promotes org health.

When it comes to Apex Classes, there are some special challenges. First off, the total size of all Apex Classes is limited to 6 MB. Comments and test classes are not included in the calculation. If you bump up against this limit, then Salesforce Support can potentially relax the constraint, but the limit cannot be expanded indefinitely, and most customers still need to continue developing new applications on the Salesforce platform.

There are many other governor limits associated with Apex development, including the number of SOQL queries, records received, DML statements, records processed, stack depth, HTTP callouts, queued jobs, CPU time, push notifications, and sent emails. Violating any of these constraints can result in Apex Classes that throw errors or stop operating correctly.

Another consideration is the impact of poorly written code that causes reduced agility, performance problems, or security vulnerabilities. Apex Triggers that invoke too many Apex Classes or Flows are a major source of mischief. Writing unit tests and enforcing Code Coverage is another common requirement for managing Apex Classes.

This whitepaper was written to document the many ways that Metazoa Snapshot can improve Apex Class and Method management. We also discuss solutions to the Apex limit problem in the short term and strategies to deal with this issue in the long run. We present various challenges for Apex Class and Method management and where to find the right solution in the Metazoa Snapshot product.

Apex Classes That Are Not Invoked

There are about 15 metadata assets that can invoke an Apex Class, including other Apex Classes. The Snapshot Forgotten Assets report can look for Apex Classes that are not invoked by any other asset. These classes cannot be run and should probably be decommissioned. This report will return information on the different ways that every class is currently invoked and highlight all the classes that are not invoked. This report also provides information on when the class was created and last modified. That information might also be relevant.

Apex Methods That Are Not Invoked

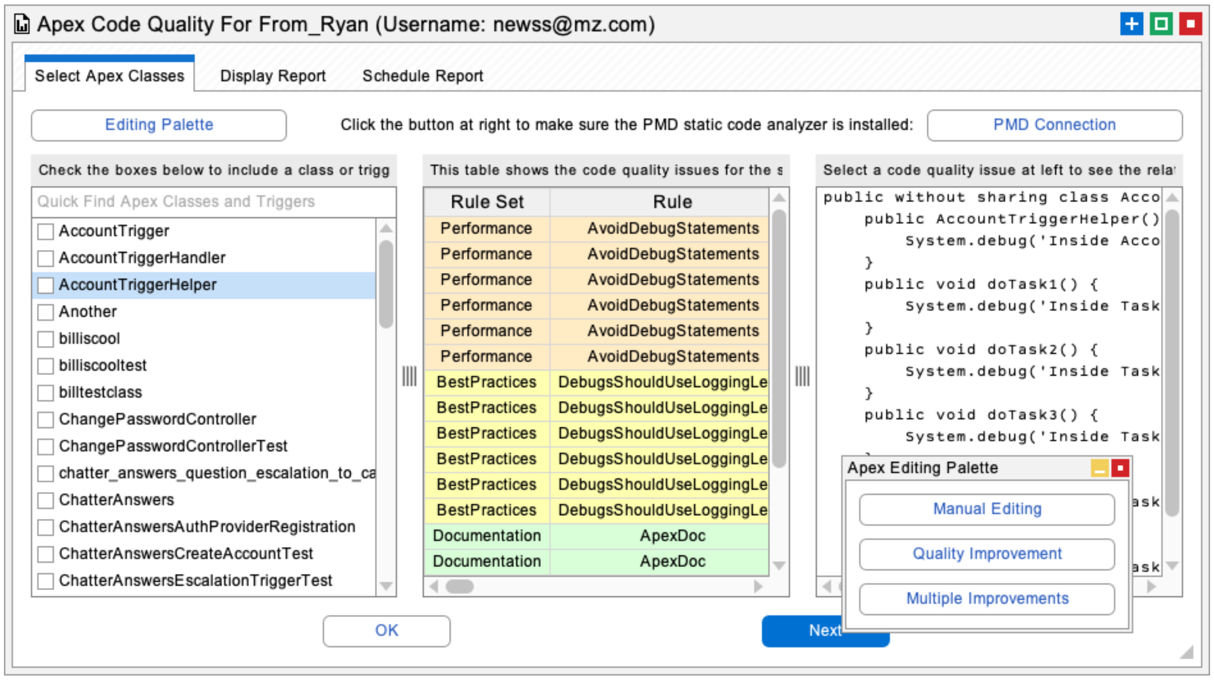
Some Apex Methods are Public and can be invoked by other metadata assets. Other Apex Methods are Private and can be invoked only within the Apex Class file that contains them. Either way, the Forgotten Assets report will search for Apex Methods that are not being invoked. These methods can potentially be removed from the Apex Class file to reduce technical debt and reduce the character size limit problem.

Duplicate Apex Methods

Developers often copy a familiar Apex Method and include it in the Apex Class file they are working on. Sometimes these methods are eventually customized, but other times they are just duplicate code that could be shared. The Forgotten Assets report will search for duplicate Apex Methods and report on all the Apex Classes that contains them. The report does not compare comments, capitalization, or white space when looking for Duplicate Methods. The report can also be scheduled to run during off hours, this is useful when there are many methods.

Apex Classes with Low Code Quality

The Snapshot Apex Code Quality Report uses PMD to generate very extensive information on over 60 different code quality factors. The quality categories include best practices, code style, design, documentation, error prone, performance, and security. This report can be exported in multiple formats including PDF, HTML, and native Excel documents. There are accompanying graphical dashboards that can be exported in multiple formats. The report can also be scheduled to run during off hours, this is useful when there are many class files. The Code Quality Report can be used for org monitoring and as a deployment gateway.



Click the button at right to make sure the PMD static code analyzer is installed:

Check the boxes below to include a class or trigg

- ☐ AccountTrigger
- ☐ AccountTriggerHandler
- ☐ AccountTriggerHelper
- ☐ Another
- ☐ billiscool
- ☐ billiscooltest
- ☐ billtestclass
- ☐ ChangePasswordController
- ☐ ChangePasswordControllerTest
- ☐ chatter_answers_question_escalation_to_ca
- ☐ ChatterAnswers
- ☐ ChatterAnswersAuthProviderRegistration
- ☐ ChatterAnswersCreateAccountTest
- ☐ ChatterAnswersEscalationTriggerTest

This table shows the code quality issues for the s

Rule Set	Rule
Performance	AvoidDebugStatements
Performance	AvoidDebugStatements
Performance	AvoidDebugStatements
Performance	AvoidDebugStatements
Performance	AvoidDebugStatements
Performance	AvoidDebugStatements
BestPractices	DebugsShouldUseLoggingLe
BestPractices	DebugsShouldUseLoggingLe
BestPractices	DebugsShouldUseLoggingLe
BestPractices	DebugsShouldUseLoggingLe
BestPractices	DebugsShouldUseLoggingLe
BestPractices	DebugsShouldUseLoggingLe
Documentation	ApexDoc
Documentation	ApexDoc

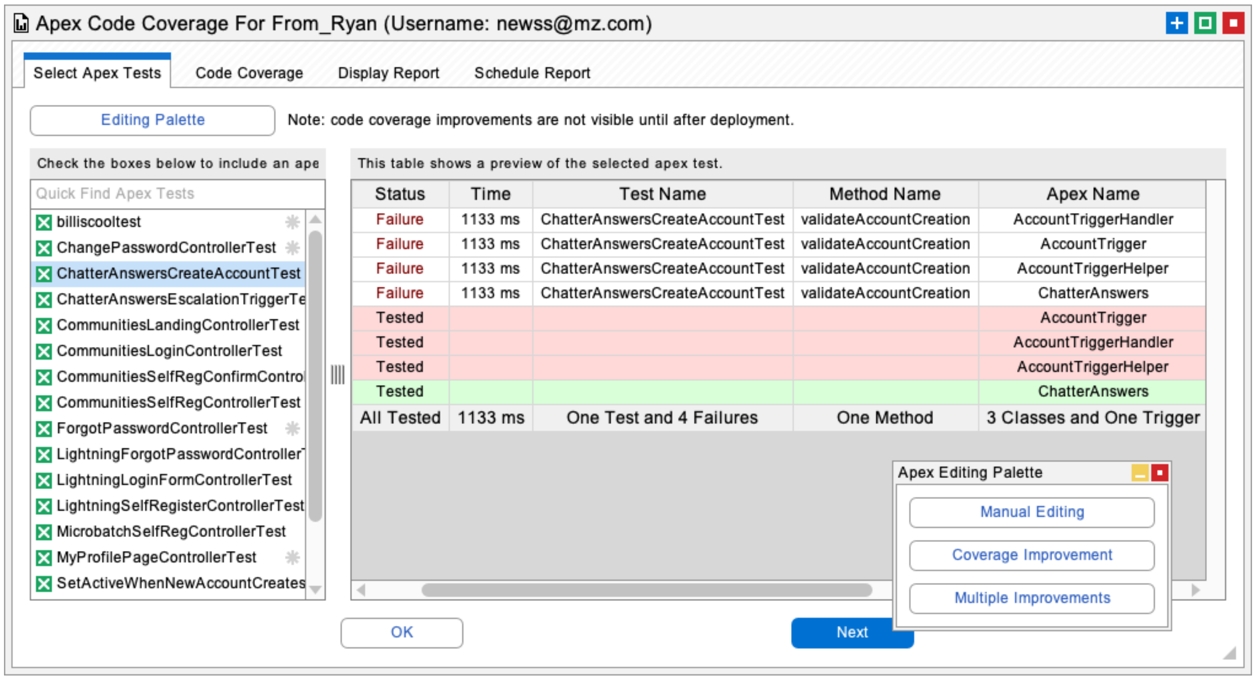
Select a code quality issue at left to see the rela

```
public without sharing class Acco
    public AccountTriggerHelper()
        System.debug('Inside Acco
    }
    public void doTask1() {
        System.debug('Inside Task
    }
    public void doTask2() {
        System.debug('Inside Task
    }
    public void doTask3() {
        System.debug('Inside Task
                                ask
                                ask
```

The Apex Code Quality Report uses PMD static code analysis to list out Code Quality Issues for any selected Apex Class or Trigger. Now with the use of Artificial Intelligence, you can automatically repair and improve the Code Quality Issues that were discovered. The improvement can start with a single Class or Trigger. You can edit the prompts and make sure that the results conform to your business practices and policies. Then the improvements in Code Quality can be automated for any number of Classes or Triggers in the Org with the Multiple Improvements button. The changes can be reviewed and deployed right from the same interface

Apex Classes with Low Code Coverage

Code coverage indicates how many executable lines of code in your Apex Classes and Triggers have been exercised by Test Methods. You must write Test Methods for your Apex Classes and Triggers, and then run those tests to generate code coverage information. Currently Salesforce requires that at least 75% of the code is covered in a production org. The Code Coverage Report can also be used for org monitoring and as a deployment gateway.



Note: code coverage improvements are not visible until after deployment.

Check the boxes below to include an ape

- billiscooltest
- ChangePasswordControllerTest
- ChatterAnswersCreateAccountTest
- ChatterAnswersEscalationTriggerTe
- CommunitiesLandingControllerTest
- CommunitiesLoginControllerTest
- CommunitiesSelfRegConfirmContro
- CommunitiesSelfRegControllerTest
- ForgotPasswordControllerTest
- LightningForgotPasswordController
- LightningLoginFormControllerTest
- LightningSelfRegisterControllerTest
- MicrobatchSelfRegControllerTest
- MyProfilePageControllerTest
- SetActiveWhenNewAccountCreates

This table shows a preview of the selected apex test.

Status	Time	Test Name	Method Name	Apex Name
Failure	1133 ms	ChatterAnswersCreateAccountTest	validateAccountCreation	AccountTriggerHandler
Failure	1133 ms	ChatterAnswersCreateAccountTest	validateAccountCreation	AccountTrigger
Failure	1133 ms	ChatterAnswersCreateAccountTest	validateAccountCreation	AccountTriggerHelper
Failure	1133 ms	ChatterAnswersCreateAccountTest	validateAccountCreation	ChatterAnswers
Tested				AccountTrigger
Tested				AccountTriggerHandler
Tested				AccountTriggerHelper
Tested				ChatterAnswers
All Tested	1133 ms	One Test and 4 Failures	One Method	3 Classes and One Trigger

The Apex Code Coverage Report shows the Code Coverage for each Test Class in the Org. Now with the use of Artificial Intelligence, you can automatically repair and improve Code Coverage for any Test Class. The improvement can start with a single Test Class. You can edit the prompts and make sure that the results conform to your business practices and policies. Then the improvements in Code Coverage can be automated for any number of Test Classes in the Org with the Multiple Improvements button. The changes can be reviewed and deployed right from the same interface.

Apex Classes with Too Many Duplicate Lines

Sometimes busy developers add lines to an Apex Class intended to boost the percentage of code coverage. This practice bypasses the quality safeguards that real test methods can provide. The Org Health Scan Report has a test that will flag Apex Classes with lots of duplicate lines. This test is especially good at finding duplicate lines intended to bypass the code coverage requirements. Duplicate lines should be replaced with legitimate tests to ensure code quality.

Apex Classes That Have Not Been Executed Recently

The Org Health Scan Report has a test that will flag Apex Classes that have not been executed recently. This test uses the Event Monitoring information available with the Salesforce Shield platform encryption product. This information captures entry points into the Apex metadata container but does not capture subsequent calls to internal Apex Classes. Still, the report is a good starting point to look for problems.

Best Practices for Apex Triggers

Apex Triggers that do not follow best practices can cause many problems. The order of trigger execution is not guaranteed, so objects with multiple triggers are a bad idea. Another problem is objects that mix triggers, workflows, and flows. These technologies do not work well together and can cause unexpected behavior and maintenance problems. Lastly, Apex Triggers should be relatively simple and just call one Apex Class. They should not reference multiple assets or implement complex logic. The Org Health Scan Report has a variety of tests that detect problems with Apex Triggers. The tests for best practices include:

- Objects With Too Many Apex Triggers
- Objects With Too Many Record Trigger Flows
- Objects That Mix Triggers, Workflows, and Flows
- Apex Triggers That Reference Multiple Assets

Best Practices for Apex Classes

The Org Health Scan Report has a variety of additional test designed to encourage best practices. Apex Classes that call or are called by external web services are a source of potential security vulnerabilities. Be sure this functionality is by design. Low version numbers in deployed Apex Classes may indicate deprecated code. The total number of Apex Classes should track closely with the number of Apex Tests. An imbalance here should be investigated. Apex Classes with too many lines of code should be divided into multiple classes. The relevant best practice tests available in the Org Health Scan Report include:

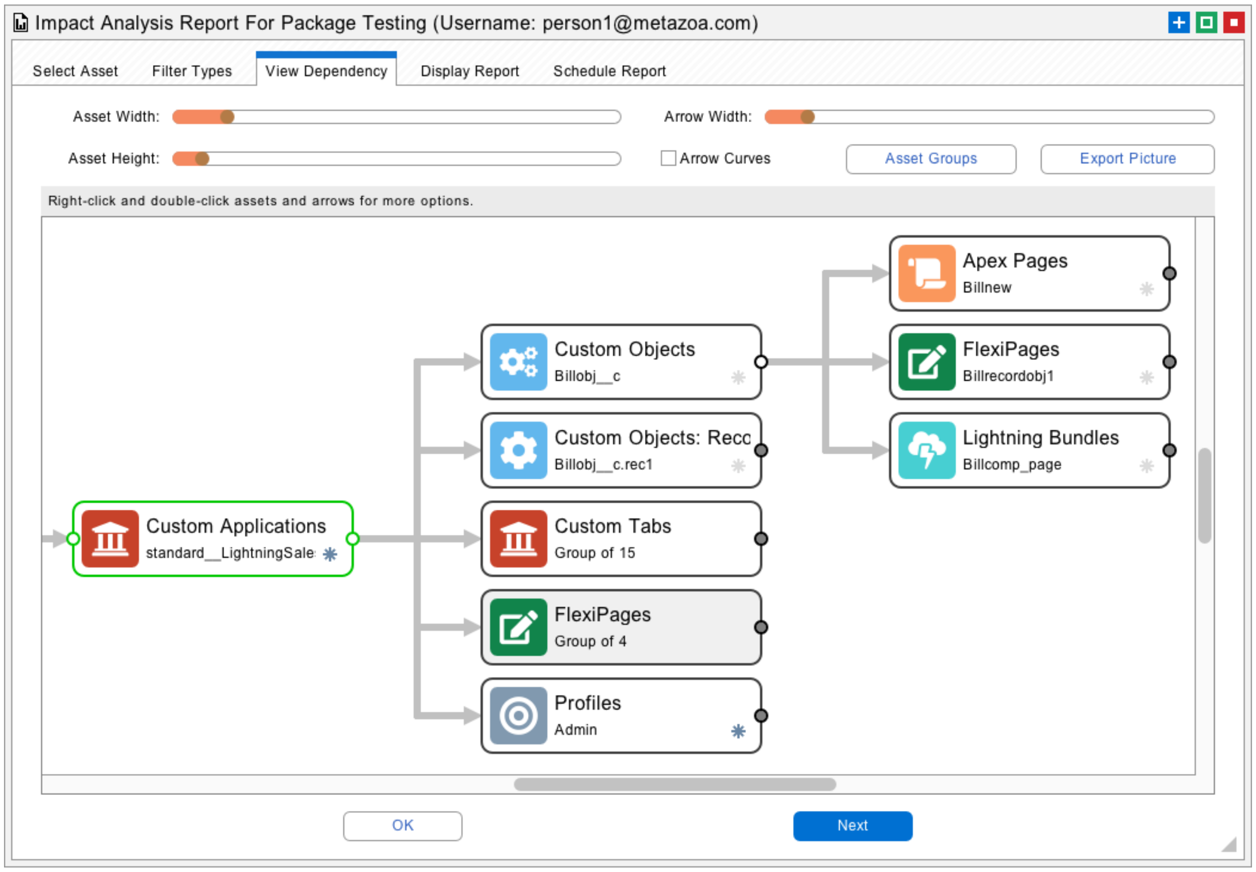
- Apex Classes That Call External Services
- Apex Classes Called by External Services
- Apex Classes with Too Many Lines of Codes
- Too Many Apex Classes Compared to Test Classes
- Apex Classes and Triggers with Low Version Numbers

Apex Class Permissions

The Profiles and Permissions Sets Report will display Apex Classes that are not enabled in any Profile or Permission Set. Salesforce permissions are tricky, and sometimes classes are invoked with a running user, but this report provides information on Apex Classes that cannot be run in the normal manner. This information should be compared with the Forgotten Assets Report to locate deprecated Apex Classes.

Impact Analysis

Snapshot includes our best of breed Impact Analysis report that covers all 250 metadata types and all 1240 dependent relationships between them. This report does not rely on the Salesforce dependency analysis API which only covers about 80 dependencies. This is a great report for investigating how Apex Classes are invoked and what other assets they reference.



Test Deletions

All the reports mentioned so far generate job lists with Apex Classes that either need to be investigated or are candidates for deletion. The Snapshot Smart Deploy interface will show the results of test deletions for any class. If there are dependent references to the class, then the test deployment will trigger an error. This is a great indication of when a class is not in fact being used.

Sandbox Migration

Working in a Sandbox, the best practice is to delete Apex Classes that are not being used and then take another Snapshot to see if there are now subsequent classes that are not being used. Once this process fails to produce incremental results then we can be pretty sure that the cleanup effort has been maximized. The Sandbox can then go through user acceptance testing and deployment back to the production org.

The Apex Limits Problem

As mentioned in the introduction, unpackaged Apex Classes are limited to a total of 6 MB of character space in an Enterprise org. As more and more orgs approach the 10-year mark, this fundamental limit can constrain future application development. The cleanup capabilities in Snapshot can go a long way to reducing unused Apex Classes and Methods in any org. But ultimately, Salesforce customers may need other solutions to address the Apex limits problem.

The most aggressive solution to any technical debt problem is an org clone, split, or reboot. Embarking on this strategy is a major undertaking that requires broad consensus for success. Reducing the size or Apex storage can be accomplished in the context of an org split. For example, when Sales, Marketing, and Support are divided into multiple orgs the amount of Apex required for any one of them can be greatly reduced. The technical debt reduction activities mentioned above would hopefully be more effective after the split, although this is not guaranteed. Metazoa provides professional services for org clone, split, or reboots.

A more practical solution to the Apex limits problem entails the use of Second Generation Managed Packages. Apex Code in managed packages does not count against the unpackaged limit. The development of new projects in the packaged format is also considered a best practice recommended by Salesforce. Many resources have been developed for this purpose including Salesforce DX, Visual Studio Code, Dev Hubs, and Scratch Orgs.

Existing projects can be repackaged as Managed Packages and new development can also take advantage of this technology. The packages can all have the same namespace or different ones. This will require adding a namespace to some class and method names. Another change is that public classes and methods in the unpackaged namespace will have to become global classes and methods once packaged.

A complication with this plan is that internal static references to custom objects and fields must be converted to dynamic sObject references or the custom objects and fields need to be included in the package. In that case, all unpackaged references to these custom objects and fields would need to be updated, and data would need to be migrated from the old unpackaged object to the new packaged object. The upshot here is that some of your Apex Classes will need to be rewritten when they are packaged.

Difficulties aside, Managed Packages are a permanent solution to the Apex Limits problem. For customers who have unlimited need for application development on the Salesforce platform they are the only game in town. Snapshot can also help administrators migrate to Managed Packages. I have written extensively about this issue at the link below.

https://www.metazoa.com/best-practices-breaking-your-org-into-packages/

Conclusion

Metazoa Snapshot has a suite of tools that can help you manage Apex Classes and Methods. We can help improve code quality and coverage as well. Unused classes and methods can be detected and decommissioned in the context of the Release Management process. In the long run, Snapshot can also help you avoid the Apex limits problem. We provide professional services for org splits and clones. Snapshot also supports application development with Managed Packages.

Bill Appleton
CTO Metazoa