

BULK FETCHING OF DB RECORDS

WITHOUT3ND.COM

m2sprite66@gmail.com



PROBLEM:

The current way one of our major apps worked was based on PL/SQL procedures. There was a managerial directive to move the logic from the OracleDB back to C/C++ code. This required that a new way of fetching records was implemented, there was some draft code I was given to follow.

ACTION:

Implemented bulk fetching approach.
Created testing suite that would create syntehtic data, insert it into the appropriate db tables And then send synthetic authorizations that were expected to either match the data inserted (positive result on test) or be false in order to test how my impelementation would handle it.

RESTULT:

- * My bulk fetching approach was alot more performant than the original draft code.
Reducing the fetching of 10000 rows from 1 second to 10 ms.
- * Was asked to create a base class that other row fetches can inherit from.
- * My testing suite found more bugs than QA.

UPGRADING AN OLD VISUAL STUDIO SOLUTION



PROBLEM:

One of the two major codebases I was working on had a visual studio solution that got retargeted upwards from visual studio 2008 to 2017. It was time to move it to 2022.

- * Build times were slow
- * The people who had retargeted it for all those years had quit the company and had left behind a messy repo filled with template files, scripts that all had codependent relationships that no one in my team understood.

ACTION:

I took the time to understand the entire complexity of the .sln (7 .vcproj files)

This allowed me to:

- * Delete unused files
- * Removed redundancy in downloading of dependencies from our servers for each individual project by creating a shared folder.
- * Removed any support for x86 that we werent supporting anymore yet the pipeline would still build binaries for it.
- * Created documentation for how all the post build and pre build events worked and what libraries each .vcproj depended on.

RESTULT:

- * Reduced the full solution build times from 5 mins to 1 min.
- * Improved readability of repo

DECYPHERING JENKINS PIPELINE



PROBLEM:

Initially my team would use other teams' kube namespaces to conduct QA on their developments. This wasted alot of time on environment configuration as residuals from the previous config would lead to errors so a lot of useless debugging would take place that increased development time by alot. This did not only include back and fourths with the previous team that'd used the given environment, but also alot of back and fourths with the devops team, to add to our already long build times.

ACTION:

Convinced my team lead to push to get us our own namespace.
He did and I was tasked with maintaining it.
Got in contact with the devops team and in 1 hour they'd expalined to me how changes were loaded into the namespace and how the docker container configs where each controlled.
I created a diagram that visually depicted all the information they'd given me.

RESTULT:

- * Granted my team full autonomy over our kubernetes namespace.
- * Reduced time to make a change by eliminating the back and fourth time with the devops team.
- * Relieved DevOps team from mineal work since my chart was added to the companies wiki and was widely shared with other teams.