Day 20 - Cache API and Cache Tags

Objective

- Drupal 8's cache API, although fairly similar in its usage to D7, has a few fascinating additions to it. Mainly the cache tags, that would allow you to invalidate caches automatically whenever related content has changed.

Primary Tutorial

- http://www.sitepoint.com/exploring-cache-api-drupal-8/
- https://dev.acquia.com/blog/drupal-8-performance-render-caching

Additional Resources

- Cache contexts are worth exploring as well. Check out **this writeup**. As another bonus exercise, extend the block such that it displays the titles of current logged in user's latest 5 published nodes (instead of all published nodes) and apply the cache context such that the block is cached for each user than globally for all users.

Debug

- If your block is not caching, it is probably because you have disabled your caching and have unset cache bins as prescribed on card Day-10.

Exercise

- Task in hand is to build a small custom block. The block is fairly simple whose content is a concatenated string with the titles of the latest 5 published nodes (of any type). Title could be set to anything arbitrary. The body of the block should be of the format - [Title of Node A]-[Title of Node B]-[Title of Node C]-[Title of Node D]-[Title of Node E] where A, B, C, D, E are the latest 5 published nodes on the site.
- Additional requirement to improve performance is that this block should be cached and this cache should be automatically invalidated when any of these 5 nodes are updated. We are not bothered about new nodes added, but the block should not display an outdated title at any point of time.
- Hint: Cache the output of the block's build() such that function doesn't compute the string if it already exists in cache. Set cache tags as node:a, node:b, node:c, node:d, node:e where a,b,c,d,e are nids of the retrieved nodes - A,B,C,D,E
- Explore a more cleaner way of implementing this. You could avoid all the logic of checking if the cache exists, and if the cache tags are valid etc, by just passing the cache tags as "#cache" to the block's render array.

Bonus Exercise

- Modify the cache tags to add something more such that the block cache is invalidated even when a new node is added (published). As of the earlier implementation, the block cache is implemented only when any of the nodes represented in the cached block are modified.