

Drupal Europe Darmstadt, Germany Sep 10-14-2018

www.drupaleurope.org





Drupal + Technology

TRACK SUPPORTED BY







Entity access for lists

A crucially missing piece of the puzzle

Kristiaan Van den Eynde





Kristiaan Van den Eynde

Senior Drupal developer







Kristiaan Van den Eynde

- Work at Factorial GmbH in Hamburg
- Live near Antwerp, Belgium
- Group module maintainer
- Happily married, recently a dad
- Highly sensitive person





Checks access **before** entities are loaded



- Checks access before entities are loaded
- Takes caching into account

- Checks access before entities are loaded
- Takes caching into account
- Supported by Views





Drupal Europe Darmstadt, Germany Sep 10 - 14, 2018

But we already have that, so what's the big deal?

"

Someone in the audience



Current implementation

The node grants system





Saves business logic to the database upon node manipulation

Drupal Europe Darmstadt, Germany XP-14 September ROLE

- Saves business logic to the database upon node manipulation
- Alters queries tagged with node_access to check for access against this saved business logic

- Saves business logic to the database upon node manipulation
- Alters queries tagged with node_access to check for access against this saved business logic
- Also used as fallback if regular access checks are indecisive



- Saves business logic to the database upon node manipulation
- Alters queries tagged with node_access to check for access against this saved business logic
- Also used as fallback if regular access checks are indecisive

Bad metaphor: Locks and keys



- Saves business logic to the database upon node manipulation
- Alters queries tagged with node_access to check for access against this saved business logic
- Also used as fallback if regular access checks are indecisive

- Bad metaphor: Locks and keys
- Better metaphor: Bouncers at a night club







Only work for "content" (aka nodes)

- Only work for "content" (aka nodes)
- Only work for view, update and delete actions



- Only work for "content" (aka nodes)
- Only work for view, update and delete actions
- As a result does not scale well for other entities



- Only work for "content" (aka nodes)
- Only work for view, update and delete actions
- As a result does not scale well for other entities
- Might try and store extremely complex access logic in the DB



Can it be fixed?



Can it be fixed?

- Not really, the concept is past its due date
- Would either require a new column on existing table or one table per entity type, both are far from ideal
- Would require a new column per supported operation, which again would lead to unwieldy (and buggy) code





Intermezzo

Possible approaches

Pros and cons



Pros and cons

Pro: We already have ENTITY_TYPE_access query tags



Pros and cons

Pro: We already have ENTITY_TYPE_access query tags

Pro: People are already used to this approach



Pros and cons

- Pro: We already have ENTITY_TYPE_access query tags
- Pro: People are already used to this approach
- Con: Complicated use cases may find themselves limited by SQL



Scalable pagination

Access checks in code



Scalable pagination

Access checks in code

- Proposed by catch early 2017
- Based on a Four Kitchens blog post from 2009
- https://www.fourkitchens.com/blog/article/anticipage-scalablepagination-especially-acls/







You ask for more results than you need and pull them through your access logic

- You ask for more results than you need and pull them through your access logic
- If you do not have enough results, go back to the database for more



- You ask for more results than you need and pull them through your access logic
- If you do not have enough results, go back to the database for more
- Keep track of the first and last item and use them for paging



- You ask for more results than you need and pull them through your access logic
- If you do not have enough results, go back to the database for more
- Keep track of the first and last item and use them for paging
- Works best on sites where most content is accessible to everyone





Pros and cons

Pro: Same access logic for both individual entities and entity lists



Pros and cons

Pro: Same access logic for both individual entities and entity lists

Pro: No "content drift" due to Reddit-style pagers (next/previous)



- Pro: Same access logic for both individual entities and entity lists
- Pro: No "content drift" due to Reddit-style pagers (next/previous)
- Con: Poor performance on sites with more complex access set-ups



- Pro: Same access logic for both individual entities and entity lists
- Pro: No "content drift" due to Reddit-style pagers (next/previous)
- Con: Poor performance on sites with more complex access set-ups
- Con: No indication of amount of possible results



- Pro: Same access logic for both individual entities and entity lists
- Pro: No "content drift" due to Reddit-style pagers (next/previous)
- Con: Poor performance on sites with more complex access set-ups
- Con: No indication of amount of possible results
- Con: People are not familiar with this approach





Drupal Europe Darmstadt, Germany Sep 10 - 14, 2018

You don't seem to be a fan of scalable pagination

Someone else in the audience



Query altering: Part Deux





Extend the entity access system with a new grants API (and deprecate the query-alter-based node grants API) <u>https://www.drupal.org/project/drupal/issues/777578</u>

- Extend the entity access system with a new grants API (and deprecate the query-alter-based node grants API) <u>https://www.drupal.org/project/drupal/issues/777578</u>
- Entity access policies

https://www.drupal.org/project/entity_access_policies



- Extend the entity access system with a new grants API (and deprecate the query-alter-based node grants API) <u>https://www.drupal.org/project/drupal/issues/777578</u>
- Entity access policies <u>https://www.drupal.org/project/entity_access_policies</u>
- Implement a query-level entity access API <u>https://www.drupal.org/project/entity/issues/2909970</u>





A collection of access plugin, e.g.: is_published





A collection of access plugin, e.g.: is_published
Used in policy config entities that list which entity types and operations they apply to

- A collection of access plugin, e.g.: is_published
- Used in policy config entities that list which entity types and operations they apply to
- When an entity query is launched, this system kicks in, finds all applicable policies and compiles them into one query alter



- A collection of access plugin, e.g.: is_published
- Used in policy config entities that list which entity types and operations they apply to
- When an entity query is launched, this system kicks in, finds all applicable policies and compiles them into one query alter
- You can build a UI showing all of the active access policies for your website and even allowing you to edit them





Pros and cons

Pro: Supports any operation and entity type



Pros and cons

Pro: Supports any operation and entity typePro: Option to have an access overview UI



- Pro: Supports any operation and entity type
- Pro: Option to have an access overview UI
- Pro: Works alongside node grants (until hopefully removed in D9)



- Pro: Supports any operation and entity type
- Pro: Option to have an access overview UI
- Pro: Works alongside node grants (until hopefully removed in D9)
- Pro: Easy to work around a problematic module





- Pro: Option to have an access overview UI
- Pro: Works alongside node grants (until hopefully removed in D9)
- Pro: Easy to work around a problematic module
- Con: Too big of a change at once to go into core



- Pro: Supports any operation and entity type
- Pro: Option to have an access overview UI
- Pro: Works alongside node grants (until hopefully removed in D9)
- Pro: Easy to work around a problematic module
- Con: Too big of a change at once to go into core
- Con: Loads a list of config entities to decide access to another list of entities

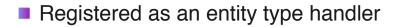




- Pro: Supports any operation and entity type
- Pro: Option to have an access overview UI
- Pro: Works alongside node grants (until hopefully removed in D9)
- Pro: Easy to work around a problematic module
- Con: Too big of a change at once to go into core
- Con: Loads a list of config entities to decide access to another list of entities
- Con: Nowhere near finished









- Registered as an entity type handler
- Generates conditions based on the user's permissions which are ultimately added to the query



- Registered as an entity type handler
- Generates conditions based on the user's permissions which are ultimately added to the query
- Fires an event to alter the conditions (so before query alter)



- Registered as an entity type handler
- Generates conditions based on the user's permissions which are ultimately added to the query
- Fires an event to alter the conditions (so before query alter)
- Works for Views and entity queries





Pros and cons

Pro: Familiar approach (entity handlers)





- Pro: Familiar approach (entity handlers)
- Pro: Works well for most scenarios out of the box



- Pro: Familiar approach (entity handlers)
- Pro: Works well for most scenarios out of the box
- Pro: Small enough in scope to make it into core



- Pro: Familiar approach (entity handlers)
- Pro: Works well for most scenarios out of the box
- Pro: Small enough in scope to make it into core
- Con: Relies on a permission handler that hasn't made it to core yet

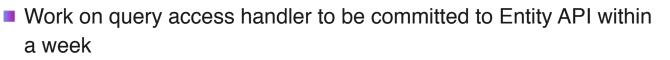


- Pro: Familiar approach (entity handlers)
- Pro: Works well for most scenarios out of the box
- Pro: Small enough in scope to make it into core
- Con: Relies on a permission handler that hasn't made it to core yet
- Con: Hard to alter or prevent another module's access logic



Good news everyone

Roadmap



- Permission handler on a fast-track to be added to core <u>https://www.drupal.org/node/2809177</u>
- This paves the way for adding the query access handlers to core
- Likely in Drupal 8.7.0



Become a Drupal contributor Friday from 9am

First timers workshop
Mentored contribution
General contribution



Thank you!



Questions?