Keywords Filters

The **Keywords Filters** in Replyke allow developers to filter entities in an entity list based on keywords. This filtering mechanism provides a way to include or exclude entities whose keywords match specific criteria. Let's dive into how it works and how you can utilize it effectively.

Overview of Keywords Filters

The `keywordsFilters` property can be passed to the `EntityListProvider` as an object with the following structure:

```
export interface KeywordsFilters {
  includes?: string[];
  doesNotInclude?: string[];
}
```

- **`includes`**: Specifies keywords that must be present in an entity's keywords array for it to be included in the list.
- **`doesNotInclude`**: Specifies keywords that must not be present in an entity's keywords array for it to be included in the list.

You can dynamically update these filters using the `updateKeywordsFilters` function provided by the `useEntityList` hook.

Dynamically Updating Keywords Filters

The `updateKeywordsFilters` function allows developers to modify the state of the keywords filters. It provides three actions:

1. **`add`**: Adds new keywords to the specified filter array (`includes` or `doesNotInclude`).
2. **`remove`**: Removes specified keywords from the specified filter array.
3. **`reset`**: Resets the specified filter array to its initial state. You can also reset both lists simultaneously.
4. **`replace`**: Replaces the entire array of keywords with a new set.

> *Important**: When calling `updateKeywordsFilters("add", "both", ...)` or `updateKeywordsFilters("replace", "both", ...)`, no action is taken, as adding/replacing

> *across both lists simultaneously is not supported.*

Here is an example of how to use `updateKeywordsFilters`:

```
const { updateKeywordsFilters } = useEntityList();

// Adding a keyword to the includes list
updateKeywordsFilters("add", "includes", "React");

// Removing a keyword from the doesNotInclude list
updateKeywordsFilters("remove", "doesNotInclude", "Vue");

// Resetting both lists
updateKeywordsFilters("reset", "both");

// Replacing the includes array with a new array
updateKeywordsFilters("replace", "includes", ["Angular"]);
```

How Keywords Filters Work

When the entity list is queried, the keywords filters are applied as follows:

1. **Includes**: If the `includes` array contains keywords, only entities whose `keywords` property contains all these keywords will be included in the list.
2. **Does Not Include**: If the `doesNotInclude` array contains keywords, entities whose `keywords` property contains any of these keywords will be excluded from the list.
3. **Combined**: When both `includes` and `doesNotInclude` are specified, entities must satisfy both conditions—they must include all keywords from `includes` and exclude all keywords from `doesNotInclude`.

Example Use Cases

Passing Static Filters to the EntityListProvider

If you want to set static filters as a starting point for your list, you can pass them directly into the `EntityListProvider`:

```
<EntityListProvider
  keywordsFilters={{ includes: ["JavaScript", "React"], doesNotInclude: ["Angular"] }}
>
  <MyFeedComponent />
</EntityListProvider>
```

This setup ensures that the list is pre-configured to show only posts about “JavaScript” or “React” while excluding posts that mention “Angular.”

Dynamically Updating Filters Based on User Interaction

To dynamically change the filters, such as when a user clicks a keyword button, use the `updateKeywordsFilters` function from the `useEntityList` hook:

```
const { updateKeywordsFilters, entities, loadMore } = useEntityList();

const handleKeywordClick = (keyword: string) => {
  updateKeywordsFilters("add", "includes", keyword);
};

return (
  <div>
    <h1>Filtered Blog Posts</h1>
    <div>
      <button onClick={() => handleKeywordClick("JavaScript")}>JavaScript</button>
      <button onClick={() => handleKeywordClick("React")}>React</button>
      <button onClick={() => handleKeywordClick("Angular")}>Exclude Angular</button>
    </div>
    <ul>
      {entities.map((entity) => (
        <li key={entity.id}>
          <h2>{entity.title}</h2>
          <p>{entity.content}</p>
        </li>
      ))}
    </ul>
    <button onClick={loadMore}>Load More</button>
  </div>
);
```

In this example, clicking a button dynamically updates the list by adding a keyword to the `includes` filter.

Important Notes

- The filter updates are applied immediately and reset the list to ensure that the updated filter conditions are reflected in the data.
- Invalid or empty keyword values are ignored.
- Combining `includes` and `doesNotInclude` provides a powerful mechanism for fine-tuning the list results.

Conclusion

The Keywords Filters offer a flexible way to include or exclude entities based on their associated keywords. By leveraging the `keywordsFilters` property and the `updateKeywordsFilters` function, you can dynamically tailor the list to meet your application's needs. Whether you're building a blogging platform, an e-commerce site, or a custom app, this filter can help you surface the most relevant content for your users.

Last updated on May 7, 2025