App Notifications Provider and useAppNotifications Hook

Like other features of Replyke, the notifications feature uses a context provider and an easy-to-use hook.

To enable notifications, wrap the content that needs access to the notifications data and functionality with the **AppNotificationsProvider**.

Once wrapped, any child components can access the notifications data and functionality via the **useAppNotifications** hook.

AppNotificationsProvider Props

The `AppNotificationsProvider` accepts two optional props:

1. **limit**: Specifies the number of notifications to fetch per batch. This strikes a balance between efficiency and the number of API calls. The default value is `10`, which is suitable for most use cases.
2. **notificationTemplates**: An object containing templates to customize notification messages. If not provided, default templates are used. This feature will be explored in the next chapter.

Example usage with custom template

```
<AppNotificationsProvider notificationTemplates={{
    entityComment: {
      "title": "$initiatorName left a comment on your post \"$entityTitle\"",
      "content": "$commentContent"
    }
}}>
    { //* Render User's notifications *// }
</AppNotificationsProvider>
```

Using the useAppNotifications Hook

The **useAppNotifications** hook provides the following data and functions:

```
interface AppNotificationsContextValues {
  appNotifications: UnifiedAppNotification[];
  unreadAppNotificationsCount: number;
  loading: boolean;
  hasMore: boolean;
  loadMore: () => void;
  markNotificationAsRead: (notificationId: string) => Promise<void>;
  resetAppNotifications: () => Promise<void>;
}
```

Available Data:

1. **appNotifications**: An array of all fetched notifications, ordered by creation time.
2. **unreadAppNotificationsCount**: The number of unread notifications.
3. **loading**: A flag indicating if more notifications are currently being fetched.
4. **hasMore**: A flag indicating if there are more notifications to fetch.

Available Functions:

1. **loadMore**: Fetches the next batch of notifications.
2. **markNotificationAsRead**: Expects a notification ID, and marks that notification as “read”.
3. **resetAppNotifications**: Resets and refetches notifications, starting with the first batch.

Creating an Engaging Notifications Experience

With the data and functions provided by the `useAppNotifications` hook, developers can create a dynamic and engaging notifications experience for users. Subsequent chapters will delve deeper into customizing notification templates and advanced integration techniques.

Last updated on May 7, 2025

© Replyke 2025