Trending Articles (Bonus)

1. Task Description

(Bonus): Implement a feature to display the top 5 trending articles based on the number of stars given by all users. This should be implemented efficiently, avoiding excessive computation or storage overhead.

2. Implementation Details

2.1 Overview

A new "Trending Articles" feature was added, allowing users to view the five articles with the highest star counts. This involved adding a new UI section, client-side JavaScript to handle interactions and display results, and a server-side Java REST resource to manage star counts and provide the trending data.

2.2 Trending Articles

- **UI:**
    - A new section (`trending-articles-container`) was added to display the trending articles. This section includes:
        - A heading ("Trending Articles").
        - A "Refresh" button (`refresh-trending-articles`).
        - A table (`trending-articles-table`) to display the articles, with columns for title, star count, description, and URL.
    - A new menu item was added to the settings qTip (`qtip-settings`) to navigate to the trending articles view (`#/trendingarticles/`).
- **Client-Side Logic (r.trending.js):**
    - **`init()`:** Initializes the module, attaching a click handler to the "View Trending Articles" link and calling `setupContainer`.
    - **`setupContainer()`:** Sets up event listeners for the "Refresh" button and shows a "Click Refresh to view trending articles" message initially. This separation ensures initialization only happens once.
    - **`showTrendingArticles()`:** Resets the main view (`r.main.reset()`), shows the `trending-articles-container`, and sets up initial state.
    - **`fetchAndDisplayTrendingArticles()`:** Makes an AJAX GET request to `/api/trending/top` to retrieve the trending articles data. On success, it calls `displayTrendingArticles` to render the data in the table. On error, it displays an error message.
    - **`displayTrendingArticles(articles)`:** Clears the `trending-articles-table` body and populates it with the provided article data. Handles cases where no trending articles are found. Creates links for the article URLs.
    - **`handleStarClick(articleElement, starButton)`** and **`handleDestarClick(articleElement, starButton)`:** No Longer needed as this is being handled by `r.article.js`.

    - `getArticleData(articleElement)`: No longer needed.

- **Modified** `r.article.js`:
    - **Delegate on star buttons:** The event handler is modified.
        - Store the original state *before* toggling.
        - Choose the correct endpoint based on the *original* starred state (`../api/trending/destar` or `../api/trending/star`).
        - Call the Trending API and the original star API.
        - If the Trending API call fails, **DO NOT** revert UI changes here, only log/alert errors.
        - If the original star API call fails, revert the UI.

- **Server-Side Logic (TrendingArticleResource.java):**
    - Provides REST endpoints for managing star counts and retrieving trending articles.
    - Uses in-memory data structures (`ConcurrentHashMap`) to store article details and star counts. This is efficient for reads, but data will be lost if the server restarts.
    - Maintains a cached list (`top5ArticlesCache`) of the top 5 articles, sorted by star count (descending) and then article ID (ascending) for tie-breaking. The cache is updated whenever a star count changes.
    - `starArticle` **(POST to** `/api/trending/star`**):**
        - Receives the full article object, including a unique `id` (which is the article title).
        - Stores/updates the article details in `articleDetailsMap`.
        - Atomically increments the star count for the article in `starCountsMap`.
        - Updates the `top5ArticlesCache`.
    - `destarArticle` **(POST to** `/api/trending/destar`**):**
        - Receives the full article object, including the `id` (article title).
        - Decrements the star count for the article in `starCountsMap`, ensuring it doesn't go below zero.
        - Updates the `top5ArticlesCache`.
    - `getTop5Articles` **(GET to** `/api/trending/top`**):**
        - Retrieves the top 5 articles from the `top5ArticlesCache`.
        - Fetches the corresponding article details from `articleDetailsMap`.
        - Returns a JSON response containing an array of article objects, each with the article detail and star count.
    - `updateTop5Cache` **(private, synchronized):**
        - Updates the sorted cache of top 5 articles. It is `synchronized` to ensure thread safety, as multiple users could be starring/destarring articles concurrently.
        - Handles adding new articles to the cache, updating existing ones, and maintaining the correct sort order.

3. API Endpoints

Method	Endpoint	Description	Request Body	Response Body

Method	Endpoint	Description	Request Body	Response Body
POST	`/api/trending/star`	Increment the star count for an article.	`{ "id": "string", "articleObject": { "title": "string", "url": "string", ... } }`	`{ "status": "success", "articleId": "string", "newStarCount": number }` or `{ "message": "string" }`
POST	`/api/trending/destar`	Decrement the star count for an article.	`{ "id": "string", "articleObject": { "title": "string", "url": "string", ... } }`	`{ "status": "success", "articleId": "string", "newStarCount": number }` or `{ "message": "string" }`
GET	`/api/trending/top`	Get the top 5 trending articles (highest star count).	None	`{ "topArticles": [ { "articleObject": { "title": "string", ... }, "starCount": number }, ... ] }` or `{ "message": "string" }`

4. Design Patterns Used

- **Module Pattern:** Client-side code is organized into modules (e.g., `r.Trending`).
- **Observer Pattern:** Used for event handling (button clicks, link clicks).
- **Asynchronous Request (AJAX):** Client-server communication is asynchronous.
- **Client-Side MVC Elements:** Similar to the other features, with separation of data, view, and controller logic.
- **RESTful API:** The backend is a RESTful API.
- **Caching:** The `top5ArticlesCache` on the server-side provides a simple caching mechanism to improve performance for retrieving the trending articles. This avoids recalculating the top 5 on every request.
- **Concurrency Control:** `ConcurrentHashMap` is used for thread-safe storage of star counts and article details. The `updateTop5Cache` method is synchronized to prevent race conditions when updating the cache.

5. Changes in Files and Description

File	Changes

File	Changes
`index.html`	Added a new container (`trending-articles-container`) with a table (`trending-articles-table`) and a refresh button (`refresh-trending-articles`). Added a new menu item in `qtip-settings` for "View Trending Articles". Added navigation binding for `#/trendingarticles/`.
`r.trending.js`	(New) Implements client-side logic for fetching and displaying trending articles. Handles user interactions and updates the UI.
`r.article.js`	Modified the star button click handler to make *two* API calls: one to the trending API (`/api/trending/star` or `/api/trending/destar`) and another to original star API. Handles success and failure cases, reverting UI changes only if the original API call fails.
`r.main.js`	Added line to hide new trending articles container.
`TrendingArticleResource.java`	(New) Implements the server-side REST endpoints for managing star counts and retrieving the top 5 trending articles. Uses `ConcurrentHashMap` for in-memory storage and maintains a sorted cache of the top 5 articles.