Routing System Description

Overview

The routing system in this application serves two primary purposes: 1. **Initial page load**: When the application first loads, routing determines which page/pane to display 2. **Navigation between panes**: When users click links or buttons that need to switch to different panes or specific elements within panes

The router is configured with `updateLocation: false`, meaning it does NOT update the browser's URL bar or history. Instead, it maintains its own internal history stack and only updates a custom location bar display.

When Routing is Performed

1. Application Initialization

- **When**: On page load, when `router.start()` is called
- **Trigger**: `handlePopState()` is called automatically
- **Purpose**: Determines the initial route based on the current URL path
- **Location**: `router.js:253` - `router.start()` calls `this.handlePopState()`

2. Tab Click (when pane doesn't exist)

- **When**: User clicks a tab link (`[data-tab-link]`) and the corresponding pane doesn't exist in the DOM
- **Trigger**: Tab click handler in `App.initializeTabs()` (line 488-523)
- **Condition**: `targetPanel` is null but `location` attribute exists on the link
- **Action**: Calls `this.router.navigate(location)`
- **Location**: `ui/app.js:519-521`
- **Note**: If the pane exists, routing is NOT performed - the pane is simply shown/hidden via CSS

3. Login Success

- **When**: User successfully authenticates via the login form
- **Trigger**: `setupInlineLogin()` success handler
- **Action**: `app.router.navigate(\`$BASE_PATH/account/{accountName}`)`
- **Location**: `ui/pages/index.js:2178`
- **Purpose**: Navigate to the account pane after authentication

4. Repository Creation

- **When**: User creates a new repository
- **Trigger**: Repository creation form submission success
- **Action**: `app.router.navigate(\`$BASE_PATH/account/{accountName}/repositories/${repositoryNam`

- **Location**: `ui/pages/index.js:2592`
- **Purpose**: Navigate to the newly created repository pane

5. Repository Deletion

- **When**: User deletes a repository
- **Trigger**: Repository deletion confirmation
- **Action**: `app.router.navigate(\`$BASE_PATH/account/${accountName}\`)\``
- **Location**: `ui/pages/index.js:1072`
- **Purpose**: Navigate back to the account pane after deletion

6. Logout

- **When**: User clicks the logout button
- **Trigger**: `App.handleLogout()`
- **Action**: `this.router.navigate("/login", { replace: true })`
- **Location**: `ui/app.js:228`
- **Purpose**: Navigate to login pane and clear session

7. My Account Redirect

- **When**: User navigates to `/account` without a specific account name
- **Trigger**: `App.renderMyAccount()` when no current account exists
- **Action**: `this.router.navigate("/login", { replace: true })` if not logged in, or `this.router.navigate(\`$basePath/account/${current.friendlyId}\`, { replace: true })\`` if logged in
- **Location**: `ui/app.js:163, 167`
- **Purpose**: Redirect to login or the current account's pane

8. Browser Back/Forward Navigation

- **When**: User clicks browser back/forward buttons
- **Trigger**: `popstate` or `hashchange` events
- **Action**: `router.handlePopState()` → `router.handlePath(path)`
- **Location**: `router.js:179-184, 247-250`
- **Purpose**: Restore previous route state

9. Direct Link Clicks (non-tab links)

- **When**: User clicks any `<a>` tag that:
  - Has an `href` starting with `/`
  - Does NOT have `data-tab-link` attribute
  - Does NOT have `data-action` attribute
  - Does NOT have `data-external` attribute
  - Is not a mailto link
  - Is not a hash link (`#...`)
  - Is not an external link (different origin)

  - Does not point to a file (e.g., `.html`, `.css`, `.js`)
- **Trigger**: Global click listener on document (bubble phase)
- **Action**: `router.handleLinkClick()` → `router.navigate(href)`
- **Location**: `router.js:187-245, 252`
- **Purpose**: Handle navigation from regular links in content

When Routing is NOT Performed

1. Tab Click (pane exists)

- **When**: User clicks a tab and the pane already exists in the DOM
- **Action**: Direct DOM manipulation (show/hide via CSS)
- **Location**: `ui/app.js:503-518`
- **Reason**: No need to route - just change visibility

2. Tab Actions (close, save)

- **When**: User clicks tab close or save buttons
- **Action**: Direct DOM manipulation
- **Location**: `ui/app.js:526-580`
- **Reason**: These are UI actions, not navigation

3. View Editor Operations

- **When**: Opening/closing view editors in repository panes or separate panes
- **Action**: Direct DOM manipulation via `openViewPane()`, `createEditorPanel()`, etc.
- **Location**: `ui/pages/index.js` - various editor functions
- **Reason**: Editors are managed within existing panes or created dynamically

4. Form Submissions (except login/repository creation)

- **When**: Most form submissions (edit forms, etc.)
- **Action**: Direct API calls and DOM updates
- **Reason**: These update existing panes, not navigate to new ones

5. Buttons with `data-action` or `data-tab-action`

- **When**: Any element with these attributes
- **Action**: Handled by specific event handlers
- **Location**: Various handlers in `ui/pages/index.js` and `ui/app.js`
- **Reason**: These have explicit handlers that prevent routing

Route Handler Behavior

Each route handler in `ui/routes.js` follows this pattern:

1. **Check if pane exists**: Look for the pane element in the DOM by ID
2. **If exists**: Call `app.activateTab()` to show it (no routing needed)
3. **If doesn't exist**:
    - For panes (login, info, account, repository, view): Call the corresponding `open*Pane()` function to create it dynamically
    - For full pages: Call `app.renderPage()` to render the entire page

This pattern ensures that: - Existing panes are simply shown/hidden (fast, preserves state) - New panes are created dynamically (no full page re-render) - Only true "pages" trigger full re-renders

Key Design Decisions

1. **`updateLocation: false`**: The router doesn't update the browser URL, maintaining its own history. This allows for a SPA experience without URL changes.
2. **Tab links skip router**: Links with `data-tab-link` are handled by `App.initializeTabs()`, not the router. This allows tabs to work even when panes exist.
3. **Pane existence check**: Route handlers check if panes exist before creating them, enabling fast tab switching.
4. **Direct DOM manipulation**: Most UI operations (showing/hiding panes, creating editors) use direct DOM manipulation rather than routing, for performance and state preservation.

Pages That Trigger Full Re-Renders

A full re-render occurs when `app.renderPage()` is called, which replaces the entire DOM (`this.root.innerHTML = html`). The following routes **always** trigger full re-renders:

Authentication & Account Management Pages

- **`/`** → `HomePage` - Initial landing page
- **`/signup`** → `SignupPage` - User registration
- **`/reset_password`** → `ResetPasswordPage` - Password reset
- **`/confirmations/new`** → `ConfirmationsPage` - Email confirmation
- **`/unlocks/new`** → `UnlocksPage` - Account unlock
- **`/account/new`** → `AccountNewPage` - Create new account
- **`/account/:account_name/edit`** → `AccountEditPage` - Edit account metadata
- **`/account/:account_name/auth_token`** → `AccountAuthTokenPage` - View auth token

Repository Management Pages

- `/account` → `RepositoriesIndexPage` - List repositories
- `/repositories` → `RepositoriesIndexPage` - List repositories (alternative)
- `/account/:account_name/repositories` → `RepositoriesIndexPage` - List repositories for account
- `/account/:account_name/repositories/:repository_name/edit` → `RepositoryEditPage` - Edit repository metadata
- `/account/:account_name/repositories/:repository_name/import` → `RepositoryImportPage` - Import data
- `/account/:account_name/repositories/:repository_name/query` → `SparqlPage` - SPARQL query interface
- `/account/:account_name/repositories/:repository_name/query_logs` → `RepositoryQueryLogsPage` - Query history

View/Query Pages

- `/account/:account_name/repositories/:repository_name/views/:view_name/execute` → `ViewRoute` - Execute view
- `/account/:account_name/repositories/:repository_name/views/:view_name/meta` → `ViewRoute` - View metadata

Repository Status Pages

- `/account/:account_name/repositories/:repository_name/status` → `RepositoryRoute` - Repository status
- `/account/:account_name/repositories/:repository_name/size` → `RepositoryRoute` - Repository size
- `/account/:account_name/repositories/:repository_name/meta` → `RepositoryRoute` - Repository metadata

Invitation Pages

- `/invite` → `InvitationsNewPage` - Create invitation
- `/invite/success` → `InvitationsSuccessPage` - Invitation success
- `/invitations` → `InvitationsIndexPage` - List invitations

Special Pages

- `/standalone-editor` → `StandaloneEditorPage` - Standalone query editor
- `/_maintenance` → `MaintenancePage` - Maintenance mode
- `/_rpc_test` → `RpcTestPage` - RPC testing
- `/_template` → `TemplatePage` - Template page
- `/404` → `NotFoundPage` - 404 error page

Conditionally Full Re-Render (only if no DOM structure exists)

- `/login` → `LoginPage` - Only if `#content-container` doesn't exist (otherwise uses `openLoginPane()`)
- `/info` → `HomePage` - Only if `#content-container` doesn't exist (otherwise uses `openInfoPane()`)

Routes That Never Full Re-Render (use pane management)

- `/account/:account_name` → Uses `openAccountPane()` or `activateTab()` (direct DOM manipulation)
- `/account/:account_name/repositories/:repository_name` → Uses `openRepositoryPane()` or `activateTab()` (direct DOM manipulation)
- `/account/:account_name/repositories/:repository_name/views/:view_name` → Uses `openViewPane()` or `activateTab()` (direct DOM manipulation)

Summary

Routing is primarily used for: - Initial page load - Creating new panes that don't exist - Navigation after significant state changes (login, logout, create/delete resources) - Browser history navigation - **Full page re-renders for "true pages"** (forms, special interfaces, status pages)

Routing is NOT used for: - Switching between existing panes (handled by tab clicks with direct DOM manipulation) - UI actions within panes (handled by specific event handlers) - Editor operations (handled by direct DOM manipulation) - **Main workspace navigation** (account/repository/view panes use direct DOM manipulation)

This design minimizes unnecessary routing and page re-renders while maintaining a clean navigation model. Full re-renders are reserved for: 1. **Form pages** (edit, create, import) 2. **Special interfaces** (SPARQL query, standalone editor) 3. **Status/metadata pages** (status, size, meta, query logs) 4. **Authentication flows** (signup, reset password, confirmations) 5. **Initial page load** (when no DOM structure exists)