SCASUBJI User Interface

SCASUBJI User Interface

Please note that this page and related pages are being developed as part of the CIM Courses Project and are subject to change.

This article describes the functionality of the SCASUBJI form. This form allows power Registrar users to directly edit Subject data in the Container/Template Subject Structure.

- SCASUBJI User Interface
  - Accessing SCASUBJI
    - Links
    - Authorizations
  - Restrictions/Validations
  - FAQ
  - Developer info
    - Thin apache proxy
      - Location and configuration
      - `/service/apis/user` and `/service/apis/tier` endpoints
      - Architecture diagram
    - Frameworks used
    - APIs
    - Setup
    - Two main views
      - 1: Search
        - MIT subjects
        - Non-MIT subjects
        - Other functionality on the page
          - Deactivate
          - Reactivate
          - Backdate
          - Create New Subject link
          - Show All Templates for Container link
      - 2: Create/Edit Subject
        - Supporting data
          - Tier
          - Department info
          - Requisites
          - DisplayControls
    - CSS
  - Subject Management Documentation Index

Accessing SCASUBJI

Links

User access SCASUBJI user interface through the SMD forms collection:

- Dev - https://sis-data-maint-dev.mit.edu/sdm/
- Test - https://sis-data-maint-test.mit.edu/sdm/
- Prod - https://sis-data-maint.mit.edu/sdm/

Direct links to the SCASUBJI UI:

- Dev - https://subjectmgmt-dev.mit.edu/#/
- Test - https://subjectmgmt-test.mit.edu/#/
- Prod - https://subjectmgmt.mit.edu/#/

Authorizations

Please refer to the Subject Management Authorizations KB page. The roles that pertain to CIM Courses access are sent via the User Provisioning Feed.

Restrictions/Validations

The majority of validations performed by SCASUBJI can be found on the Subject Management Business Rules KB page. Many of the restrictions/validations are performed using the MIT Subject Management API. This article lists the restriction/validations that are performed on the front-end by SCASUBJI only.

- Perform a check on save for a Subject Number = "URN" and for attribute "UROP". If these conditions are true, check for a "URN" attribute. If "URN" is not present, "URN" will automatically be included in the JSON to the API upon save.
- If no value is provided for Roll_Faculty and no value already exists, set to "P" for "Previous Term"

FAQ

Q: **Why are the input boxes related to Equivalencies and Scheduling Relationships grayed out except for the on a container's most recent template?**
A: TBD (Tim to add)

Developer info

Thin apache proxy

All requests from the UI are routed through an apache server which acts as a thin proxy to the back-end APIs.

The apache server:

- Serves the files that make up the angular application (css, html, js, etc)
- Integrates with Touchstone
- Relays requests to API, appending client id and secret and referred user info
- Exposes endpoints for getting property and user info

Location and configuration

The apache server lives on a virtual machine in our VMWare Cloud. It is managed by the App Delivery team and configured via Puppet.

The puppet config for the non-production tiers can be found at https://github.mit.edu/ops/puppet-environment. File: hiera:nodes: <servername>: common.yaml. Contact the App Delivery team for information about the production configuration.

Production is load balanced between two servers.

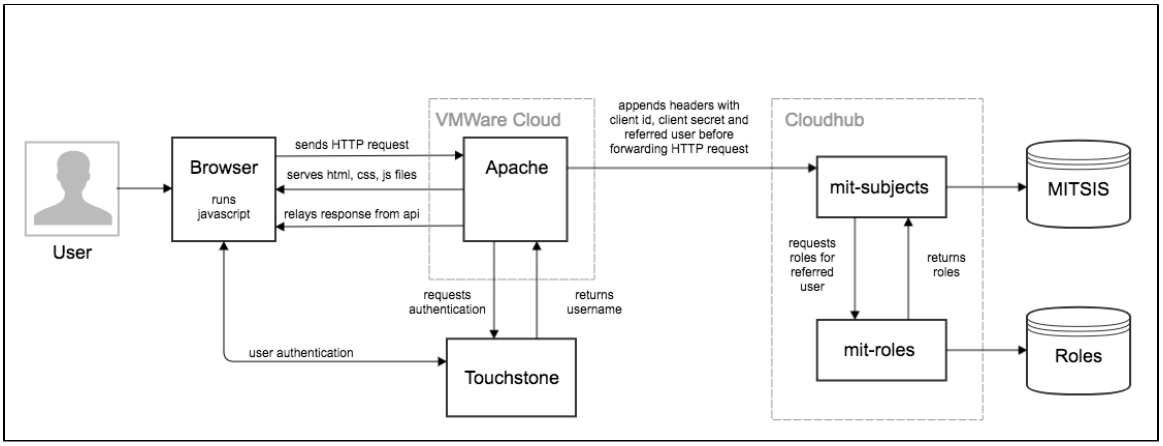
`/service/apis/user` and `/service/apis/tier` endpoints

Two endpoints are configured directly in the apache configuration.

- The `/service/apis/user` endpoint returns information provided by Touchstone about the authenticated user
- The `/service/apis/tier` endpoint returns tier-specific properties to be used by the UI

Architecture diagram

Show/hide diagram



[Image source: https://wikis.mit.edu/confluence/display/EduSys/Thin+Apache+Proxy+Diagram]

Frameworks used

- AngularJs 1.63
- Bootstrap 3
- angular-query-builder for requisites editing (https://mfauveau.github.io/angular-query-builder/)
- Build tools: Grunt, Bower, Sass CSS preprocessor, husky/prettier (on commit)

APIs

https://ist-developer.mit.edu/academics/mit-subjects-v1

APIs start with: /service/apis/subjectmanagement/

Calls to APIs also have a unique token appended as a cachebuster (will appear as 'token_' plus a timestamp). This will prevent the browser from using cached versions of API results.

Setup

Clone repository:
```
git clone git@github.mit.edu:ist-devops/subject-managment-ui.git
```

To build local version, from inside root folder of the repo you just cloned, run:
```
npm install
```

To run local server, from inside same folder:
```
grunt serve
```

Work from inside the /app folder. /dev folder will be created upon any saves.

Two main views

SCASUBJI comprises two main views: 1. a search page that enables user to manage templates/containers, and 2. a create/edit page to enter details on a subject, including cross-listings, equivalents, requisites, etc.
(Note that the Home button takes user to the main SDM search page, not the main SCASUBJI page.)

1: Search

- View: views/home.html
- Controller: scripts/controllers/homeCtrl.js
- Factory: Factories/homeFactory.js

MIT subjects

Clicking search button calls searchSubj() in controller. If this is an MIT subject, this then calls:

```
homeFactory.getSubjectDtls($scope.course, $scope.number.toUpperCase())
```

which calls API endpoint:

```
/subjects/courses/{subjectCode}/numbers/{subjectNumber}
```

If a term is included, it calls a different method, from different factory:

```
viewFactory.getSubjectDtls($scope.course, $scope.number.toUpperCase(), $scope.offeringTerm.toUpperCase())
```

API endpoint is called to make sure department entered as part of subject is valid:

```
/subjectCodes/{departmentCode}
```

This returns metadata about the department, e.g. for department 22,

```
departmentCode: " 22"
departmentName: "Nuclear Engineering"
schoolCode: "E"
schoolDescription: "Engineering"
```

Then processSearchResult (in controller homeCtrl.js) is called, whether a term was included or not in search. This sets the data ($root.searchResults) used to render the main table.

Non-MIT subjects

If it's a non-MIT search, it calls the same method, with or without a term provided:

```
homeFactory.getNonMitSubjectDtls($scope.nonMitCourseNumber, $scope.offeringTerm.toUpperCase() )
```

the factory method then calls the API:

```
/subjectmanagement/subjects/homeSubjects/{nonMitCourseNumber}
```

This retrieves a list of templates and stores it in the page controller to render the main table. (For non-MIT subjects this will usually be 1 match.)

Other functionality on the page

These functions enable to user to manage a template's metadata.

Deactivate

Confirmation of term code entered in modal runs deactivateSubject(), which calls the API:

```
/subjects/containers/{containerId}
```

After the user confirms the deactivate request, the subjects list is updated.

Reactivate

Confirmation of term code entered in modal runs reactivateSubject(), which redirects to the edit page for that subject in that term (using the containerId in the container list):

```
/#/editSubj/{course}/{number}/{containerId}/reactivate
```

Backdate

Similar to Deactivate, confirmation of term code entered in modal runs backdateContainer(), which then calls the API:

```
/subjects/containers/{containerId}/resetEffectiveTerm
```

Create New Subject link

Redirects to

```
/#/newSubj
```

This takes user to Edit Subject screen, described below.

Show All Templates for Container link

Displays all templates for a container. Each link should expand to show all templates for only that containerId.

In a few cases (e.g. 6.802) there are multiple containerIds.

uiSettings is a scope var that keeps track of user's Show All Templates choice, key is containerId.

2: Create/Edit Subject

This is the main SCASUBJI subject editing screen.

- views/editSubject.html
- scripts/controllers/editSubjCtrl.js
- factories/editFactory.js

Clicking on link in table calls Subjects endpoint with course/number/term params (and cache buster token) and calls API:

```
/subjects/courses/{department}/numbers/{subject}/terms/{termCode}
```

Supporting data

On page load, the following lookups are made (some APIs, some from apache config):

Tier

```
/service/apis/tier
```

Sets metadata for current tier (DEV, TEST, SCHED-TEST, or PROD) e.g.:

```
attributesUrl: "https://sis-data-maint-dev.mit.edu/sdm/subjectAttributes/list",
gradingModesUrl: "https://sis-data-maint-dev.mit.edu/sdm/gradingModeCodes/list",
sdmUrl: "https://sis-data-maint-dev.mit.edu/sdm/",
tier: "DEV"
```

Department info

`/subjectCodes/{departmentCode}`
Sets metadata for this department; see example above.

Requisites

Retrieval API:

```
/subjects/containers/{containerId}/templates/{templateId}/requisites
```

Save API:

```
/subjects/containers/{containerId}/requisites
```

Renders querybuilder component, which adds a node to requisites. Querybuilder calls itself recursively to render nested groups.

The save button is in the editRequisites modal, markup for this modal is in /views/editSubject.html. The save function, saveRequisitesData(), is in the edit page's controller (editSubjectCtrl.js).

DisplayControls

Gets display controls (user selected data), options (to build dropdowns), and reference point (templateId and version number). API:

```
/subjects/containers/{containerId}/terms/{termCode}/displayControlsAndUrls?optionsValues=Y
```

Under Term Plan, when user selects Fall, IAP, Spring, and/or Summer, a corresponding section will appear in the Online Listing section that follows.

CSS

Mostly uses bootstrap default CSS.

styles/main.scss comprises all other files; add to top if need to add new files.

Some media queries should be removed (use bootstrap's responsive breakpoints).

Subject Management Documentation Index

The Subject Management Documentation Index is the central listing for documentation pertaining to Subject Management.