Updated: 2/6/2017

We should have POJOS and controller.js states and behaviours here. Please don’t forget to add content for changes you make to these files throughout the course of the project.

All getters and setters exist if necessary, but not listed for document brevity.

- **Plain Old Java Objects** -

Batch.java

*Properties*

* int **ID**
* String **name**
* java.sql.Timestamp **startDate** // May change to java.time.LocalDate
* java.sql.Timestamp **endDate**
* Curriculum **curriculum** // From here down is project-defined types
* Room **room**
* BatchStatusLookup **batchStatus**
* Trainer **trainer**
* Trainer **cotrainer**
* List<Skill> **skills**

*Behaviours*

* **Batch()** - no args empty constructor
* **Batch(int iD, String name, Curriculum curriculum, Location location, Room room, Trainer trainer, Trainer cotrainer, Timestamp startDate, Timestamp endDate, BatchStatusLookup status)** - runs super() and sets all.
* **Batch(int iD, String name, Timestamp startDate, Timestamp endDate, Curriculum curriculum, Room room, Location location, BatchStatusLookup batchStatus, Trainer trainer, Trainer coTrainer)** - runs super() and sets all. This is the same as above, but in a different order.
* **toString()** - returns all info in String format

BatchStatusLookup.java

*Properties*

* int **batchStatusID**
* String **batchStatusName** // e.g. in progress, complete, in staging, etc.

*Behaviours*

* **BatchStatusLookup()** - noarg empty constructor
* **BatchStatusLookup(int batchStatusID, String batchStatusName)** - calls super() and initializes both properties

Building.java

***Implements Activatable***

*Properties*

* int **ID**
* String **name**
* int **location** - This perhaps should be changed to locationID to avoid confusion? Make sure to see where all references are being made before making this change.
* Boolean **active**
* List<Room> **rooms**

*Behaviours*

* **Building()** - noargs empty constructor
* **Building(int ID, String name, List<Room> rooms, boolean active, int location)** - constructor, sets all fields
* **Building(int location)** - constructor, sets only location’s id of reference
* **toString()** - returns all info as String

Certification.java

*Properties*

* int **id**
* String **file**
* String **name**
* int **trainer**

*Behaviours*

* **Certification()** - noargs empty constructor
* **Certification(int id, String file, String name, int trainer)** - constructor, sets all
* **toString()** - returns all info as String

Curriculum.java

***Implements Activatable***

*Properties*

* int **currId**
* String **name**
* List<Skill> **skills**
* Boolean **active** - not in constructor or toString
* Boolean **core** - is this skill a core curriculum? (if false, sub-focus)

*Behaviours*

* **Curriculum()** - noargs empty constructor
* **Curriculum(int currId, String name, List<Skill> skills)** - calls super() and initializes info
* **toString()** - returns all info as String

Location.java

***Implements Activatable***

*Properties*

* int **ID**
* String **name**
* String **city**
* String **state**
* Boolean **active**
* List<Building> **buildings**

*Behaviours*

* **Location()** - noargs empty constructor
* **Location(int iD, String name, String city, String state, List<Building> buildings, Boolean active)** - calls super() and initializes all info.
* **toString()** - returns all info as a String

Room.java

***Implements Activatable***

*Properties*

* int **roomID**
* String **roomName**
* int **building** - This perhaps should be changed to buildingID to avoid confusion? Make sure to see where all references are being made before making this change.
* List<Unavailable> **unavailabilities** - represents unavailable date ranges
* List<Batch> **batches**
* Boolean **active** - not in constructor or toString

*Behaviours*

* **Room()**
* **Room(int roomID, String roomName, List<Unavailable> unavailabilities)**
* **Room(int roomID, String roomName, List<Unavailable> unavailabilities, Boolean active)**
* **toString()** - returns all info as a String

Setting.java

*Properties*

* int settingId
* String settingName
* Double settingValue

*Behaviours*

* **Setting()** - noargs empty constructor
* **Setting(int settingId, String settingName, Double settingValue)**
* **toString()** - returns all info as a String

Skill.java

***Implements Activatable***

*Properties*

* int skillId
* String name
* Boolean active - not in constructor or toString

*Behaviours*

* **Skill()**
* **Skill(int skillId, String name)** - calls super() and initializes ID and Name
* **toString()** - returns all info as String

Trainer.java

***Implements Activatable***

*Properties*

* int **trainerId**
* String **firstName**
* String **lastName**
* String **resume** - (résumé)
* List<Unavailable> **unavailabilities** - represents unavailable date ranges
* List<Skill> **skills**
* List<Certification> **certifications**
* Boolean **active** - not in the constructor

*Behaviours -* there is no toString() defined

* **Trainer()** - noargs empty constructor
* **Trainer(int trainerID, String firstName, String lastName, List<Unavailable> unavailable, List<Skill> skill, List<Certification> certification)** - calls super() and initializes all info.

Unavailable.java

*Properties*

* int **ID**
* java.sql.Timestamp **startDate**
* java.sql.Timestamp **endDate**

*Behaviours*

* **Unavailable()** - noargs empty constructor
* **Unavailable(int iD, Calendar startDate, Calendar endDate)** - calls super() and initializes all info.
* **toString()** - returns all info as String

- **Controllers.js** -

\*\*Note: Documentation may still not be complete for all controllers due to the out-of-place nature of JavaScript.

authController.js

This is called aCtrl in other files.

*Properties* - none

*Behaviours*

* //$scope.$on(‘$viewContentLoaded’, function(){...}) - commented out, but looks like it is validating user via token
* **showToast(function(message){...})** - note says: global function available to all other controllers (as they are all children of authCtrl) - creates toast messages.

batchController.js

This is called bCtrl in other files.

-- *Batches Tab* --

*Properties*

* **bc** = this; - prefixes all other properties for clarity
* **oldBatchEndDate** - used for start date validation
* **oldRoom** - needed to update room in db when editing a batch / trainer
* **oldTrainer** - needed to update room in db when editing a batch / trainer
* **availableTrainers** - determined by batch dates vs trainer unavailable dates
* **availableRooms** - determined by batch dates vs room unavailable dates
* **weeksSpan** = “spans 0 weeks”; - shows up as a calculation for length of batch determined by startDates and endDates
* **batchOrder** = “startDate”;
* **batch** - batch being edited / worked on
* **batchesSelected[ ]**
* **trainerSkillRatios[ ]** = array that contains the skill ratios to the curriculum for each trainer
* **state** - mode state (e.g. create, edit, clone)
* **stateMux** - message which changes based on state.
* **findHQ** - number representing id of default location
* **findHQBuilding** - number representing id of default building
* **trainers**
* **buildings**
* **selectedSkills**

*Behaviours*

* **convertUnavailability(incoming)** - converts incoming date to new Date(incoming) not currently in use.
* **changeState(newState, (Batch) incomingBatch)** - Populates / unpopulates fields on state change.
* **updateTrainers((list of trainers) trainers, (Date) batchStart, (Date) batchEnd)** - sets property availableTrainers, utilizing trainerSelection filter by comparing all unavailable dates of trainers in list against batch dates.
* **updateCurriculumRatios()** - updates the skill match ratio for all the trainers.
* **calcTrainerSkillRatio((Trainer) trainer)** - puts a number next to ‘%’ in trainer dropdown for selection based on core batch skill selected.
* **selectEndDate()**
* **enableMondays(date)**
* **enableFridays(date)**
* **filterBuildings(locationID)**
* **filterRooms(locationID)**
* **updateWeeks()**
* **defaultName()**
* **calcProgress(paramLow, paramHigh)**
* **highlightBatch(batch)**
* **selectedBatchRow(batch)**
* **resetForm()** - clears info on page
* **toggleAll()**
* **allSelected()** - checks if all batches have been selected. May not be needed.
* **exists(batch)**
* **toggle(batch)**
* **repull()** - get all info from the db again
* **edit(batch)**
* **clone(batch)**
* **delete(batch)**
* **deleteMultiple()**
* **deleteMultipleHelper(delList)**
* **saveBatch(isValid)**
* **subtractUnavailabilities()** - temporarily removes unavailability for room and trainer if selected for batch edit. Needs optimization.
* **deleteUnavailabilities()** - subtracts unavailabilities, then persists them.
* **validateBatchEndDate()**
* **updateTrainersAndRooms(trainers, rooms, batchStart, batchEnd)**
* **updateRooms(rooms, batchStart, batchEnd)**
* **updateSelectedSkills()**
* **updateBatchSkills()**
* **updateSkillRatios()**
* **saveUnavailabilities()** - saves unavailabilities to trainer and room upon batch save

deleteDialogController.js

*-- For ‘deletion’ on Buildings Tab --*

*Properties*

* **dc** = this;
* **thinking** = (deleted == summary.rooms + summary.locations);
* **summary** - summary of what is to be deleted
* **desc** - description of what is to be deleted
* **list** - list of buildings / locations / rooms selected

*Behaviours*

* **formatText()**
* **cancel()** - closes modal
* **delete()**
* **deletehelper(delList)** - called by delete
* **deleteBuildings(arr)** - inactivates a list of buildings
* **deleteRooms(arr)** - inactivates a list of rooms

homeController.js

*-- Overview Tab (landing page) --*

*Properties*

* **hc** = this;
* **batchOrder** = “startDate”;
* **batchFilter** = “All”;
* **trainerOrder** = “firstName”;
* **trainerFilter** = “All”;
* **locationOrder** = “name”;
* **locationFilter** = “Current”;
* **filterMux** - sets up json for filter mode selection
* **trainers**
* **locations**
* **buildings**
* **batches**

*Behaviours*

* **calcProgress(paramLow, paramHigh)** - calculates progress of batch as of current date
* **checkAvailability(dates)** - may not be used
* **findRoomsAvailable(rooms)** - may not be available here?
* **formatBatches**

locationController.js

*-- Locations Tab --*

*Properties*

* **lc** = this;
* **selectedList[ ]**
* **locations**

*Behaviours*

* **openLocation(location)** - expands selected location to reveal buildings
* **openBuilding(building)** - expands selected building to reveal rooms
* **addLocation()**
* **addBuilding()**
* **addRoom()**
* **removeBuildings(location)** - removes buildings from selected list on location close
* **editSelected()**
* **deleteSelected()**
* **formatMessage(summary)** - formats toast message based on deletion summary
* **categorizeSelected()**
* **exists(obj)** - may not be used
* **toggle(obj)**
* **visible(location)** - may not be used
* **repull()** - repulls all locations from the db

locationDialogController.js

*-- Locations Modal --*

*Properties*

* **ldc** = this;
* **stateMux** - message displayed based on state
* **title**
* **USstates**

*Behaviours*

* **cancel()**
* **save()**

menuController.js

*-- Menu Bar --*

*Properties*

* **mc** = this;
* **currentPage**

*Behaviours*

* **findCurrentPage()** - sets navbar to current page even on refresh

profileController.js

*-- Profile Tab --*

*Properties*

* **pc** = this;
* **tId** - trainer ID grabbed from URL
* **skills**
* **myFile**
* **creds**
* **certName**
* **skillsList[ ]**
* **trainer**

*Behaviours*

* uploadResume()
* SaveTSkills()
* addSkill()
* removeSkill()
* uploadCertification()
* removeCertification()
* rePullSkills()
* rePullTrainer()
* getAllSkills()

reportController.js

*-- Reports Tab --*

\*Needs clarity and optimization\*

*Properties*

* **rc** = this;
* **chart1**
* **chart2**
* **canSubmit**
* **data[ ]**
* **newTable[ ]**
* **year**
* **today**
* **reqDate**
* **startDate**
* **totalNetBatch**
* **totalSDETBatch**
* **totalJavaBatch**
* **totalCumulativeBatches**
* **newObj**
* **cardArr**
* **currOrder**
* **monthList**
* **toggleBatch**
* **toggleGrad**
* **toggleIncoming**

*Behaviours*

* **export()**
* **export2()**
* **currSummary(curriculum)**
* **currSummary2(curriculum)**
* **sumCurrYear(total, num)**
* **sumYear**
* **sumYear2**
* **sumMonth**
* **sumMonth2**
* **calcStartDate(requiredDate, index)**
* **calcReqBatch(requiredTrainees, index)**
* **assignCurr(bType, index)**
* **genCard()**
* **removeCardClick(index)**
* **cumulativeBatches()**
* **submittionValidityAssertion(index)**
* **createBatchClick(index)**
* **createAllBatchClick()**
* **toggleCrateBatchToolbar()**
* **toggleGradToolbar()**
* **toggleIncomingToolbar()**
* **graphData()**
* **graphData2()**
* **$scope.myGraph()**
* **$scope.myGraph2()**

roomDialogController.js

*-- Room Modal --*

*Properties*

* **rdc** = this;

*Behaviours*

* **cancel()**
* **save(roomForm)**

timelineController.js

*-- Bottom of Batches Tab --*

*Properties*

* **minDate** = minimum date range on the timeline graph.
* **oldMinDate** = previous valid minDate that was set.
* **maxDate** = maximum date range on the timeline graph.
* **oldMaxDate** = previous valid maxDate that was set.
* **currentDate** = today’s current date.
* **options** = setting for the date picker dropdowns.
* **timelineFormatting** = object that contains properties for formatting the timeline.
* **selectedCurriculum** = used for filtering batches based on the curriculum view.
* **selectedFocus** = used for filtering batches based on the focus view.
* **selectedLocation** = used for filtering batches based on the location view.
* **selectedBulding** = used for filtering batches based on the building view.
* **trainersPerPage** = input for how many trainers to display per page.
* realTrainersPerPage = the actual value for trainers to display per page, under the constraints.
* **trainerPage** = input for the current page of trainers viewed.
* **realTrainerPage** = actual value of the current page of trainers viewed.
* **maxTrainerPages** = total number of pages for trainers on the timeline.
* **trainerListStartIndex** = start index of the list on the trainers for pagination.
* **trainerListEndIndex** = end index of the list on the trainers for pagination.
* **previousPageButtonDisabled** = whether or not the previous page button is disabled.
* **nextPageButtonDisabled** = whether or not the next page button is disabled.
* **hideConcludedBatches** = checkbox entry for hiding batches that have already finished.
* **hideBatchlessTrainers** = checkbox entry for hiding trainers with no batches.
* **filteredTrainers** = final list of trainers to display on timeline after all the filtering.
* **filteredBatches** = final list of batches to display on timeline after all the filtering.

*Behaviours*

* **isUndefinedOrNull(obj)** - function for validating an object exists.
* **isNaN(obj)** - function for validating the value is a valid number.
* **removeNoTrainer(batch)** - filter for removing batches with no trainer.
* **removeConcluded(batch)** - filter for removing batches that have finished.
* **removeDateless(batch)** - filter for removing batches with a null date.
* **removeOutOfDateRange(batch)** - filter for removing batches that fall outside the viewed date range.
* **removeUnmatchingCurriculum(batch)** - filter for removing batches that don’t match the selected curriculum view.
* **removeUnmatchingLocation(batch)** - filter for removing batches that don’t match the selected location view.
* **removeUnmatchingBuilding(batch)** - filter for removing batches that don’t match the selected building view.
* **removeUnmatchingFocus(batch)** - filter for removing batches that don’t match the selected focus view.
* **removeIrrelevantBatches(batch)** - filter for removing batches not displayed on the current page.
* **removeTrainersOutOfPage(trainer)** - filter for removing trainers that are outside the current page.
* **removeBatchlessTrainers(trainer)** - filter for removing trainers that don’t have any assigned batches.
* **validateStartDate()** - ensures the start date can’t be set after the end date.
* **validateEndDate()** - ensures the end date can’t be set before the start date.
* **getDateRange()** - Gets the initial value on the minimum and maximum dates for the timeline, based on the batches.
* **getAllBatches()** - Promise that retrieves all the batches.
* **getSelectedLocation()** - searches the master list of locations and returns the location object that matches the location view, if found.
* **getAllTrainers()** - Promise that retrieves all the trainers.
* **repullPromise()** - Promise that re-retrieves the data needed to re-project the timeline.
* **repull()** - calls the repull Promise, and upon its completion, re-projects the timeline.
* **projectTimelineOnly()** - re-projects the timeline, without repulling anything.
* **changeTrainersPerPage(useFilteredTrainers)** - updates the amount of trainers displayed per page.
* **previousTrainerPage(useFilteredTrainers)** - goes to the previous page of trainers.
* **nextTrainerPage(useFilteredTrainers)** - goes to the next page of trainers.
* **firstTrainerPage(useFilteredTrainers)** - jumps to the first page of trainers.
* **lastTrainerPage(useFilteredTrainers)** - jumps to the last page of trainers.
* **goToTrainerPage(useFilteredTrainers)** - jumps to a specific page of trainers.
* **previousPageButtonStatus(useFilteredTrainers)** - the conditions required for the previous page buttons to be enabled/disabled.
* **nextPageButtonStatus(useFilteredTrainers)** - the conditions required for the next page buttons to be enabled/disabled.
* **useFilteredBatches()** - the conditions on which the list of filtered batches should be used for logic.
* **filterTimelineData()** - filters the data to be displayed on the timeline as necessary, based on all the settings from the user.
* **projectTimeline()** - filters the trainers and batches, and calls the project timeline function.
* **trainerColumnName(trainer)** - formatting of the string displayed in each column header on the timeline.
* **projectTimeline(timelineFormatting, minDate, maxDate, yCoord, timelineData, parentScope, numWeeks, trainerNames)** - projects the timeline, using the d3 library for the display.

trainerController.js

*-- Trainers Tab --*

*Properties*

* **tc** = this;

*Behaviours*

* **addTrainer()**
* **removeTrainer(trainerRM)**
* **grabS3Resume(trainer)**
* **activateTrainer(trainer)**
* **joinObjArrayByName(elem)** - reformats how an array of objects is joined
* **goToTrainer(event)** - redirects the url to selected trainer’s profile page
* **rePullTrainers()**
* **convertUnavailability(incoming)**
* **showCalendar()**
* **hideCalendar()**
* **showPTODialog()**

trainerDialogController.js

*-- Trainer Modal --*

*Properties*

* **tdc** = this;
* **stateMux** - currently only one state, so may be unnecessary

*Behaviours*

* **cancel()**
* **save(isValid)**

*Shared link* in Google Docs, for real-time editing viewable / editable by all: <https://docs.google.com/document/d/1bXGjaT-Bp82DzU8nNKpFsBF9EF9QiuEIUpGLt_I_P48/edit?usp=sharing>

Make sure to update the documentation within the project folder as well after significant changes have been made, preferably by one person.