Evaluation Implementation

Information

These are the architect notes related to the implementation of the evaluation system. Most of the system is self documenting (i.e. notes about the implemenation are clearly written in the APIs and IMPLs), however, some aspects are less clear so those are listed and detailed here.

Implementation

External data handling

- External data is handled via a set of interfaces. Most of this passes through the EvalExternalLogic interface
- The following types of external data are planned to be supported:
    - **External Users**
        - Users are handled in the system as a unique id (referred to as userId)
        - the userId is a String and must be unique and must not change
        - this data can come from anywhere but included implementation is to use the Sakai UserDirectoryService
        - Data requested:
            - Current userId
            - Username from userId
            - User DisplayName from userId
            - Is userId a super admin
    - **External Sites/Permissions**
        - Determining who can take an evaluation is handled by associating a collection of users (a context) with an evaluation via an AssignContext
        - A context is a unique id for a collection of users and must not change
        - Contexts and the users within them are requested via the external interface
        - Determining who can take an evaluation or be evaluated is done via permissions within that context
        - Included implementation uses Sakai Sites and Authz
        - Data requested:
            - Current context
            - Does userId have a permission in a context
            - get DiplayTitle by context
            - Create a Context object from a context
            - Get userIds for all users in a context with a permission
            - Get all contexts which a userId has a specific permission in
    - **External Courses/Enrollments**
        - External courses and enrollments are handled via an interface which sits behind the EvalExternalLogic interface (and is quite close to the EvalExternalLogic interface)
        - This data is passed into the system via the methods in the EvalExternalLogic API
        - This allows courses and enrollments to be passed into the system via an implemented API from any system that can answer the questions asked by the API
        - Example: EvaluationImpl => ExternalLogic => ExternalLogicImpl => ContextProvider => ContextProviderImpl => Sakai CourseManagementProvider
        - Data requested:
            - Get userIds for all users in a course with a permission (take_eval or be_evaluated)
            - Get all courses which a userId has a specific permission in (take_eval or be_evaluated)
            - Get a count of all courses which a userId has a specific permission in (take_eval or be_evaluated)
            - Get a Context from a unique course id
    - **External Hierarchy structure and courses**
        - The hierarchy of courses, departments, colleges, schools, etc. are handled via an interface which sits behind the EvalExternalLogic interface
    - **External Scales/Items/Templates**
        - *TBD*

Hierarchy

- The system should support a hierarchy of evaluation groups and parent groups that is configurable and can have as many layers as necessary
- Ideally this will be provided through Sakai, however, initially, it will have to be done separately

Locking

- The locking logic is designed to make it easier to know if an entity should or should not be changed or removed
    - Locked entities can never be removed via the APIs and should not be removed with direct access to the DB
    - Locked entities can never be modified except in special cases (evaluations are a special case)
        - Some evaluation settings can be adjusted even if the evaluation itself is locked, these settings are detailed in the code comments
    - Locking should never be modified directly on an object via a save method (e.g. saveItem())
        - This means setLocked should almost never be called outside the logic layer
        - There is a circumstance where you might set a lock on an entity from the view layer when creating expert items, but this is a special case and unique in the current design

        - The locking for an entity is controlled by other entities so in most cases entities should not lock themselves
            - The notable exception to this is when an expert item or scale is created that should not be removed
- **Lock methods**
    - Entities must use the **lock** methods to lock other entities
    - DAO Lock methods should not be called outside the logic layer despite the fact that they are accessible via APIs
- **Locking process**
    - The locking process is a chain of entities that lock entities "behind" themselves in the chain when they are locked
    - Entities will use the lock methods to lock other entities
    - When a chain of locks or unlocks is triggered this will be referred to as a **lock cycle**
    - Locking happens when the lock state of an entity not at the end of a chain changes
        - **Locked** - entity will now lock all associated entities below itself in the chain
        - **Unlocked** - entity will now unlock all associated entities below itself that are not locked by other locked entities
            - Unlock cycles are much more intensive that lock cycles
- Chain of entities
    - **Scale** (*Only locked and unlocked by other entities, never locks anything, end of the chain*)
    - **Item** (*Items are locked once they are used in any locked template, they can only unlock if they are not used in any locked templates, lock and unlock associated scales*)
    - **Template** (*Templates are locked once they are used in a single evaluation, they can only unlock if they are not used in any locked evaluations, lock and unlock associated items*)
    - **Evaluation** (*Evaluations are locked once there is a single response associated with them, can only unlock if they have no associated responses, lock and unlock the associated template*)
    - **Responses** (*In general, Responses will be created locked and therefore their creation is the trigger point for most lock cycles, lock and unlock the associated evaluation*)

Expert Templates, Items, and Scales

- Expert entities are supported to make it easier for users to create an evaluation which is well formed and collects the data they want
    - Ideally this allows the user to create a good evaluation without needing to be an expert in writing evaluations
    - Expert entities should be created by an expert in evaluation
- Expert entities will be marked as such and presented to the user in a special way
- Expert entities follow the same rules as normal entities EXCEPT that they cannot be removed, it is important for developers to realize this because an expert entity can be modified based on standard permissions but it cannot be removed so a canControl check will return true on an expert entity but a delete call will cause an exception to be thrown

Sharing

- Many entities in the system can be shared with others, this sharing is hierarchical in nature and currently limited to public and private until the hierarchy is in place
- Sharable entities - Templates, Items, Scales
- Sharing states
    - **OWNER** - sharing is defined by the owner (*this is a special state and should not be used when saving entities, attempting to use it will cause an exception*)
    - **PRIVATE** - visible and controllable to owner and admin only
    - **PUBLIC** - visible to everyone, only editable by owner and admin
    - **SHARED** (*NOT IMPLEMENTED*) - visible and controllable by those at the same hierarchy level
    - **SHARED_BELOW** (*NOT IMPLEMENTED*) - visible and controllable by those at the same hierarchy level, also visible below this level
    - **VISIBLE** (*NOT IMPLEMENTED*) - visible to those at the same hierarchy level, control by admin and owner only
    - **VISIBLE_BELOW** (*NOT IMPLEMENTED*) - visible to those at the same hierarchy level and below, control by admin and owner only
- The sharing states are defined using constants and well documented in the code

System settings

- These are settings that mostly controlled via the Administrate view, they allow an admin to adjust the overall function of the evaluation system for a Sakai installation
    - All updates and retrieval of system settings are done via the Settings API
    - All settings are referenced via constants in the Settings API
- System settings can be removed by updating the setting to a value of null
- Boolean override (ternary) settings are special and should have 3 states
  The override settings are those which basically override a configurable setting elsewhere in the system (e.g.
    - **TRUE** - All settings in the system must match this
    - **FALSE** - All settings in the system must match this
    - **NOT SPECIFIED** (NULL, which is not stored) - Settings may be set to true or false
- Other setting types can be overridden as well as needed
- Boolean and ternary booleans settings are listed in arrays in the EvalSettings API file so the system knows about these settings and can handle them correctly
    - All boolean and ternary settings must be listed in the array constants
    - Boolean settings are guaranteed to return true or false and will not return a null
- Settings are cached to reduce database load
    - The cache is as cluster aware as it can be in Sakai and should flush out the values when they are changed
    - There is also an automated timed process which reloads all settings every hour

Entities

- Entities refer to persistent objects in the evaluation system
    - Entities should be comparable to each other using object equality

Blocks

- Items can be placed into a block which presents a group of scaled items in a compact display format where they all share the same scale labels
- Detailed information about the functioning of blocks is available at Item blocks

Duplicate fields on Item and TemplateItem

- There are some duplicate fields on the Item and TemplateItem tables as indicated below:
    - USES_NA
    - SCALE_DISPLAY_SETTING
    - DISPLAY_ROWS
    - CATEGORY (ITEM_CATEGORY on TemplateItem)
- These fields are there because items can be created independent of a template
- They are to be used as hints for the TemplateItem when it is created and any null fields from this list will inherit the value from the Item when the TemplateItem is saved
- **Note:** You should not use these fields on the Item when rendering an Item in a Template, always use the values from the TemplateItem

Evaluation States

- Evaluations have multiple states which they can be in, these states are determined by the dates set for each specific evaluation Detailed documentation on evaluation states and dates
- An automated job runs every hour to ensure evaluations stay in sync

Protected fields (fields which should not be modified at certain times)

- There are fields on many of the persistent objects that should not be modified once the object has been created (i.e. first saved in the database), the objects and the fields are documented below
- **Note:** ID field is a special case, the **id** field on all persistent objects should never be modified in any way whether the object has been created or not
- Persistent Objects (fields cannot be changed after object is created):
    - **Answer**
        - response
        - item
    - **AssignContext**
        - context
        - evaluation
    - **EmailTemplate**
        - defaultType
    - **Item**
        - locked
    - **Response**
        - owner
        - context
        - evaluation
    - **Scale**
        - locked
    - **Template**
        - locked
    - **TemplateItem**
        - template
        - item
- **Evaluation** is a special case in that the state affects what fields may be modified
  The list belows details what changes are allowed at each state of an evaluation
    - InQueue (cannot modify)
        - locked
        - template
        - instructorOpt
        - termId
    - Active (cannot modify)
        - locked
        - template
        - instructorOpt
        - termId
        - availableEmailTemplate
        - reminderEmailTemplate
        - startDate
    - Due (cannot modify)
        - locked
        - template
        - instructorOpt

        - termId
        - availableEmailTemplate
        - reminderEmailTemplate
        - startDate
        - stopDate
    - Closed (can only modify)
        - viewDate
        - studentsDate
        - instructorsDate
    - Viewable (No changes can be made at all)

Evaluation added items

- Items can be added to evaluations while they are in-queue but not yet active (depending on settings), these items are typically referred to as instructor added items. These items are stored in a separate template from the original one used to create the evaluation (this is to avoid the locked template issue and avoid corrupting the original set of items)
    - The secondary template is created when the first added item is placed in the evaluation
    - Secondary templates are flagged using a field and constant variable
    - Secondary templates should not appear in the list of templates and are managed by the system only
    - Items within the secondary template have 2 fields which are used to store the hierarchical location of the template item
- Eval takers should only see the items applicable to them, a method in the logic layer will return the correct items for an evaluation and user and correctly handle the logic for filtering items
- Question adders (instructors or admins) should only have access to change the items they are adding and yet need to be able to see all items in the evaluation above their level so they know what else exists in this eval, this will allow them to reorder their own items but not mess up the other items in the evaluation template

Adhoc users and groups

- Adhoc groups and users (created as needed) are stored in the eval database
    - these allow the user to define a group of email addresses to send an evaluation to
    - Adhoc users
        - essentially like real users which come from Sakai except that they are stored in the eval system
        - adhoc users have no password at this time so adhoc users get to skip the authentication step
    - Adhoc groups
        - essentially like external groups, sites, or site-groups except that they are stored in the eval system
        - no permissions are associated with adhoc groups except the ability to take evals assigned to that group, evaluatees in adhoc groups do not have the ability to create templates or begin evaluations

Optimizations

- The implementation will take an "optimize last" approach. We will finish the functionality for a milestone before we attempt to optimize the performance.
- Some optimization is designed into the system via the data model and the business logic layer methods