```ruby

#
# Copyright (C) 2011 - present Instructure, Inc.
#
# This file is part of Canvas.
#
# Canvas is free software: you can redistribute it and/or modify it under
# the terms of the GNU Affero General Public License as published by the Free
# Software Foundation, version 3 of the License.
#
# Canvas is distributed in the hope that it will be useful, but WITHOUT ANY
# WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FC
# A PARTICULAR PURPOSE. See the GNU Affero General Public License for more
# details.
#
# You should have received a copy of the GNU Affero General Public License
along
# with this program. If not, see <http://www.gnu.org/licenses/>.
#

class QuestionBanksController < ApplicationController
  before_action :require_context, :except => :bookmark
  add_crumb(proc { t('#crumbs.question_banks', "Question Banks") }, :except =>
:bookmark) { |c| c.send :named_context_url,
c.instance_variable_get("@context"), :context_question_banks_url }

  include Api::V1::Outcome

  def index
    if @context == @current_user || authorized_action(@context, @current_user,
:read_question_banks)
      @question_banks =
@context.assessment_question_banks.active.except(:preload).to_a
      if params[:include_bookmarked] == '1'
        @question_banks += @current_user.assessment_question_banks.active
      end
      if params[:inherited] == '1' && @context != @current_user
        @question_banks += @context.inherited_assessment_question_banks.active
      end
      @question_banks = @question_banks.select{|b| b.grants_right?
(@current_user, :manage) } if params[:managed] == '1'
      @question_banks = Canvas::ICU.collate_by(@question_banks.uniq) { |b|
b.title || CanvasSort::Last }
      respond_to do |format|
        format.html
        format.json { render :json => @question_banks.map{ |b|
b.as_json(methods: [:cached_context_short_name, :assessment_question_count]) }
      end
```

```
41      end
42    end
43
44    def questions
45      find_bank(params[:question_bank_id], params[:inherited] == '1') do
46        @questions = @bank.assessment_questions.active
47        url = polymorphic_url([@context, :question_bank_questions],
:question_bank_id => @bank)
48        @questions = Api.paginate(@questions, self, url, default_per_page: 50)
49        render :json => {:pages => @questions.total_pages, :questions =>
@questions}
50      end
51    end
52
53    def reorder
54      @bank = @context.assessment_question_banks.find(params[:question_bank_id])
55      if authorized_action(@bank, @current_user, :update)
56
@bank.assessment_questions.active.first.update_order(params[:order].split(',')
57        render :json => {:reorder => true}
58      end
59    end
60
61    def show
62      @bank = @context.assessment_question_banks.find(params[:id])
63      js_env({
64        :CONTEXT_URL_ROOT => polymorphic_path([@context]),
65        :ROOT_OUTCOME_GROUP => outcome_group_json(@context.root_outcome_group,
@current_user, session)
66      })
67      rce_js_env
68
69      add_crumb(@bank.title)
70      if authorized_action(@bank, @current_user, :read)
71        @alignments = Canvas::ICU.collate_by(@bank.learning_outcome_alignments)
|a| a.learning_outcome.short_description }
72        @questions = @bank.assessment_questions.active.paginate(:per_page => 50,
:page => 1)
73      end
74
75      js_bundle :quizzes_bundle, :question_bank
76      css_bundle :quizzes, :learning_outcomes, :tinymce, :question_bank
77      @page_title = @bank.title
78    end
79
80    def move_questions
81      @bank = @context.assessment_question_banks.find(params[:question_bank_id])
82      @new_bank =
AssessmentQuestionBank.find(params[:assessment_question_bank_id])
```

```
83      if authorized_action(@bank, @current_user, :update) &&
    authorized_action(@new_bank, @current_user, :manage)
84        ids = []
85        params[:questions].each do |key, value|
86          ids << key.to_i if value != '0' && key.to_i != 0
87        end
88        @questions = @bank.assessment_questions.where(:id => ids)
89        if params[:move] != '1'
90          attributes = @questions.columns.map(&:name) - %w{id created_at
    updated_at assessment_question_bank_id}
91          connection = @questions.connection
92          attributes = attributes.map { |attr| connection.quote_column_name(attr
    }
93          now = connection.quote(Time.now.utc)
94          connection.insert(
95              "INSERT INTO #{AssessmentQuestion.quoted_table_name}
    (#{(%w{assessment_question_bank_id created_at updated_at} + attributes).join('
    ')}) " +
96              @questions.select(([@new_bank.id, now, now] + attributes).join(',
    ')).to_sql)
97        else
98          @questions.update_all(:assessment_question_bank_id => @new_bank.id)
99        end
100
101       [ @bank, @new_bank ].each(&:touch)
102
103       render :json => {}
104     end
105   end
106
107   def create
108     if authorized_action(@context.assessment_question_banks.temp_record,
    @current_user, :create)
109       @bank = @context.assessment_question_banks.build(bank_params)
110       respond_to do |format|
111         if @bank.save
112           @bank.bookmark_for(@current_user)
113           flash[:notice] = t :bank_success, "Question bank successfully
    created!"
114           format.html { redirect_to named_context_url(@context,
    :context_question_banks_url) }
115           format.json { render :json => @bank }
116         else
117           flash[:error] = t :bank_fail, "Question bank failed to create."
118           format.html { redirect_to named_context_url(@context,
    :context_question_banks_url) }
119           format.json { render :json => @bank.errors, :status => :bad_request
120         end
121       end
```

```ruby
    end
  end

  def bookmark
    @bank = AssessmentQuestionBank.find(params[:question_bank_id])

    if params[:unbookmark] == "1"
      render :json => @bank.bookmark_for(@current_user, false)
    elsif authorized_action(@bank, @current_user, :update)
      render :json => @bank.bookmark_for(@current_user)
    end
  end

  def update
    @bank = @context.assessment_question_banks.find(params[:id])
    if authorized_action(@bank, @current_user, :update)
      if @bank.update(bank_params)
        @bank.reload
        render :json => @bank.as_json(:include => {:learning_outcome_alignment
=> {:include => {:learning_outcome => {:include_root => false}}}})
      else
        render :json => @bank.errors, :status => :bad_request
      end
    end
  end

  def destroy
    @bank = @context.assessment_question_banks.find(params[:id])
    if authorized_action(@bank, @current_user, :delete)
      @bank.destroy
      render :json => @bank
    end
  end

  private

  def bank_params
    params.require(:assessment_question_bank).permit(:title, :alignments =>
strong_anything)
  end
end
```