**PackageController Documentation**

**Overview**

The PackageController manages subscription packages and their features in a Laravel application. It handles package creation, updates, deletions, and settings management. The controller supports multilingual packages and includes features for bulk operations.

**Class Dependencies**

php

Copy

use App\Http\Controllers\Controller;

use App\Http\Requests\Package\PackageStoreRequest;

use App\Http\Requests\Package\PackageUpdateRequest;

use App\Models\BasicExtended;

use App\Models\Language;

use App\Models\Package;

**Core Features**

**1. Package Settings Management**

**Settings Methods**

* settings():
  + Displays package settings view
  + Retrieves basic extended settings
* updateSettings(Request $request):
  + Updates package expiration reminder settings
  + Handles session flash messages

**2. Package Features Management**

**Feature Methods**

* features():
  + Retrieves and decodes package features from JSON
  + Displays features management view
* updateFeatures(Request $request):
  + Encodes and updates package features
  + Updates all BasicExtended records
  + Supports multiple languages

**3. Package CRUD Operations**

**List Packages**

php

Copy

public function index(Request $request)

* Handles language-specific package listings
* Supports search functionality
* Orders packages by creation date
* Returns view with package data

**Create Package**

php

Copy

public function store(PackageStoreRequest $request)

Features:

* Uses custom request validation
* Handles featured status
* JSON encodes features
* Creates slug from title
* Uses database transaction
* Returns success message

**Edit Package**

php

Copy

public function edit($id)

* Retrieves specific package
* Handles language selection
* Includes error handling
* Returns edit view

**Update Package**

php

Copy

public function update(PackageUpdateRequest $request)

Features:

* Custom request validation
* Handles trial package settings
* Updates featured status
* JSON encodes features
* Uses database transaction
* Returns success message

**4. Delete Operations**

**Single Delete**

php

Copy

public function delete(Request $request)

Features:

* Deletes single package
* Removes associated memberships
* Deletes related receipts
* Uses database transaction
* Handles error cases

**Bulk Delete**

php

Copy

public function bulkDelete(Request $request)

Features:

* Handles multiple package deletions
* Removes associated memberships
* Deletes related receipts
* Uses database transaction
* Returns success message

**Important Features**

**1. Multilingual Support**

* Handles different language settings
* Supports default language fallback
* Language-specific content management

**2. Transaction Management**

* Uses DB::transaction for data integrity
* Proper error handling and rollback
* Safe deletion of related records

**3. File Management**

* Handles receipt file deletions
* Proper file path management
* Clean up of associated files

**Data Models**

**Package Model Relations**

php

Copy

- memberships(): Relationship with membership records

- features: JSON encoded package features

- slug: URL-friendly package title

**Required Fields**

php

Copy

- title: Package title

- featured: Featured status flag

- is\_trial: Trial package indicator

- trial\_days: Number of trial days

- features: JSON encoded features list

**Error Handling**

1. Model Not Found:

php

Copy

try {

$package = Package::query()->findOrFail($id);

} catch (ModelNotFoundException $e) {

return $e;

}

1. Transaction Errors:

php

Copy

try {

return DB::transaction(function () use ($request) {

*// Transaction logic*

});

} catch (\Throwable $e) {

return $e;

}

**Best Practices Used**

1. Request Validation

* Custom request classes (PackageStoreRequest, PackageUpdateRequest)
* Separation of validation logic

1. Database Operations

* Transaction wrapping for data integrity
* Proper model querying
* Efficient bulk operations

1. File Operations

* Safe file deletion with @ operator
* Proper path handling
* Cleanup of associated files

1. Session Handling

* Flash messages for user feedback
* Proper session management

**Usage Examples**

**Creating a Package**

php

Copy

$request->validate([...]);

$features = json\_encode($request->features);

Package::create([

'title' => $request->title,

'slug' => make\_slug($request->title),

'features' => $features,

'featured' => $request->featured ?? "0"

]);

**Bulk Deleting Packages**

php

Copy

$ids = $request->ids;

foreach ($ids as $id) {

$package = Package::findOrFail($id);

*// Delete associated memberships and files*

$package->delete();

}

**Security Considerations**

1. Input Validation

* Custom request classes
* Proper data sanitization
* Type checking

1. File Operations

* Safe file deletion
* Proper path verification
* File type validation

1. Database Operations

* Transaction usage
* Proper error handling
* Safe query building

**Related Components**

1. Models:

* Package
* BasicExtended
* Language
* Membership

1. Request Classes:

* PackageStoreRequest
* PackageUpdateRequest

1. Views:

* admin.packages.index
* admin.packages.edit
* admin.packages.settings
* admin.packages.features