**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

I'll explain the feature management in detail and help you add shipping features to the package system.

Package Features Implementation

Click to open code

Let me explain how the feature management works and the JSON structure:

1. **Why JSON is Used:**
   * Flexible storage: JSON allows storing complex data structures in a single database column
   * Easy modification: Can add/remove features without changing database structure
   * Efficient querying: Can search and filter features when decoded
   * Hierarchical data: Can store nested information about features
2. **Current JSON Structure:**

json

Copy

[

{

"name": "10 Products",

"type": "basic"

},

{

"name": "2GB Storage",

"type": "basic"

}

]

1. **New JSON Structure with Shipping:**

json

Copy

[

{

"name": "10 Products",

"type": "basic"

},

{

"name": "Express Shipping",

"type": "shipping",

"max\_weight": 20,

"price\_per\_kg": 10.50,

"allowed\_methods": ["express", "standard"],

"zones": ["domestic", "international"]

}

]

To implement this in your views, you'll need:

Package Features Form

Click to open code

To use this shipping feature system:

1. **Database Changes Required:**
   * No schema changes needed since we're using JSON
   * Existing features column can store the new structure
2. **Usage in Application:**

php

Copy

*// Get shipping features for a package*

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

$features = json\_decode($package->features, true);

$shippingFeatures = array\_filter($features, fn($f) => $f['type'] === 'shipping');

*// Check if package has international shipping*

$hasInternational = collect($shippingFeatures)->contains(function($feature) {

return in\_array('international', $feature['zones']);

});

*// Calculate shipping cost*

$weight = 15; *// kg*

$shippingFeature = $shippingFeatures[0];

$cost = $weight \* $shippingFeature['price\_per\_kg'];

1. **Benefits of This Approach:**
   * Flexible feature management
   * Easy to add new shipping methods
   * Supports complex pricing rules
   * Can be extended for different shipping providers
   * Maintains backward compatibility

Would you like me to explain any specific part in more detail or help with implementing additional shipping features?

<form id="packageFeatureForm" method="POST" action="{{ route('admin.package.features.update') }}">

@csrf

<!-- Basic Features -->

<div class="card">

<div class="card-header">

<h4>Basic Features</h4>

</div>

<div class="card-body">

<div class="feature-inputs">

@foreach($features as $feature)

@if($feature['type'] === 'basic')

<div class="form-group">

<input type="text" name="features[]" class="form-control" value="{{ $feature['name'] }}">

</div>

@endif

@endforeach

</div>

</div>

</div>

<!-- Shipping Features -->

<div class="card mt-4">

<div class="card-header">

<h4>Shipping Features</h4>

</div>

<div class="card-body">

<div class="shipping-features">

@foreach($features as $feature)

@if($feature['type'] === 'shipping')

<div class="shipping-feature-item">

<div class="form-group">

<label>Feature Name</label>

<input type="text" name="shipping\_features[][name]" class="form-control" value="{{ $feature['name'] }}">

</div>

<div class="form-group">

<label>Max Weight (kg)</label>

<input type="number" name="shipping\_features[][max\_weight]" class="form-control" value="{{ $feature['max\_weight'] }}">

</div>

<div class="form-group">

<label>Price per KG</label>

<input type="number" step="0.01" name="shipping\_features[][price\_per\_kg]" class="form-control" value="{{ $feature['price\_per\_kg'] }}">

</div>

<div class="form-group">

<label>Shipping Methods</label>

<select name="shipping\_features[][allowed\_methods][]" class="form-control" multiple>

<option value="express" @if(in\_array('express', $feature['allowed\_methods'])) selected @endif>Express</option>

<option value="standard" @if(in\_array('standard', $feature['allowed\_methods'])) selected @endif>Standard</option>

</select>

</div>

<div class="form-group">

<label>Shipping Zones</label>

<select name="shipping\_features[][zones][]" class="form-control" multiple>

<option value="domestic" @if(in\_array('domestic', $feature['zones'])) selected @endif>Domestic</option>

<option value="international" @if(in\_array('international', $feature['zones'])) selected @endif>International</option>

</select>

</div>

</div>

@endif

@endforeach

</div>

<button type="button" class="btn btn-info" id="addShippingFeature">Add Shipping Feature</button>

</div>

</div>

<button type="submit" class="btn btn-primary mt-3">Update Features</button>

</form>

<script>

document.getElementById('addShippingFeature').addEventListener('click', function() {

const template = `

<div class="shipping-feature-item">

<div class="form-group">

<label>Feature Name</label>

<input type="text" name="shipping\_features[][name]" class="form-control">

</div>

<div class="form-group">

<label>Max Weight (kg)</label>

<input type="number" name="shipping\_features[][max\_weight]" class="form-control">

</div>

<div class="form-group">

<label>Price per KG</label>

<input type="number" step="0.01" name="shipping\_features[][price\_per\_kg]" class="form-control">

</div>

<div class="form-group">

<label>Shipping Methods</label>

<select name="shipping\_features[][allowed\_methods][]" class="form-control" multiple>

<option value="express">Express</option>

<option value="standard">Standard</option>

</select>

</div>

<div class="form-group">

<label>Shipping Zones</label>

<select name="shipping\_features[][zones][]" class="form-control" multiple>

<option value="domestic">Domestic</option>

<option value="international">International</option>

</select>

</div>

</div>

`;

document.querySelector('.shipping-features').insertAdjacentHTML('beforeend', template);

});

</script>