Cheatography

Rails Refactoring Recipes Cheat Sheet

by Andrzej Krzywda (pawelpacana) via cheatography.com/20181/cs/3027/

Explicitly render views with locals

1. Go to the view, replace all `@ivar` with `var`
2. Do the same in all partials that are called from the view and always pass the params to partials explicitly with `render "products/form", {product: product}`
3. At the end of the action add an explicit render call with a full path and the locals: `render "products/new", :locals => {product: product}`
4. Find all controllers that were using the views/partials that you changed and apply the same.

Extract render/redirect methods

1. Identify all `render` and `redirect` calls in your controller's actions.
2. Extract a private method for each `render` and `redirect` call you found with descriptive name that shows your intention.
3. Find and remove any duplicated methods you might created during this refactoring in the same controller.

Extract a Single Action Controller class

1. A new route declaration above the previous (first wins)
2. Create an empty controller which inherits from the previous
3. Copy the action content to the new controller
4. Remove the action from the previous controller
5. Copy the filters/methods that are used by the action to the new controller
6. Make the new controller inherit from the `ApplicationController`
7. Change routes to add `except: [:foo_action]`

Extract routing constraint

1. Go to the controller, duplicate existing action method under different name.
2. Create a constraint that can recognize which action should be called. Put it in `routes.rb`
3. Duplicate the relevant routing rule in `routes.rb`
4. Protect first routing rule with the constraint.
5. Change the second routing rule so it delegates to the new controller action. If necessary, protect it with similar constraint.
6. Remove the irrelevant code from controller actions. Make them do only one thing.
7. (Optionally) Move the constraint(s) to separate file(s).

Extract a repository object

1. Create a class called `ProductsRepository` inside the same file as the controller
2. Find all calls to typical `Product.find/all/new/save/create` methods in the controller
3. Create those methods in the repo object
4. Add a private method, called repo in the controller (possibly in the `ApplicationController`) where you instantiate the repo.
5. Move the repository class to `app/repos/`

Extract a service object using the SimpleDelegator

1. Move the action definition into new class and inherit from `SimpleDelegator`.
2. Step by step bring back controller responsibilities into the controller.
3. Remove inheriting from `SimpleDelegator`.
4. (Optional) Use exceptions for control flow in unhappy paths.

Extract an adapter object

1. Extract external library code to private methods of your controller
2. Parametrize these methods - remove explicit request / params / session statements
3. Pack return values from external lib calls into simple data structures.
4. Create an adapter class inside the same file as the controller
5. Move newly created controller methods to adapter (one by one), replace these method calls with calls to adapter object
6. Pack exceptions raised by an external library to your exceptions
7. Move your adapter to another file (ex. app/adapters/your_adapter.rb)

By **Andrzej Krzywda** (pawelpacana)
cheatography.com/pawelpacana/
rails-refactoring.com/recipes

Published 11th December, 2014.
Last updated 12th December, 2014.
Page 1 of 1.