



Dojo Secrets • 33%
(/m/9/4311)

Overview (/m/

TDD (/m/

Dojo Secrets (/m/

Set Up (/m/

Encryption (/m/

User (/m/

Authentication (/m/

Authentication I (/m/

User Features (/m/

User Settings (/m/

Secrets (/m/

Secrets (/m/

Like (/m/

Secret Features (/m/

Like Features (/m/

Filters (/m/

Authentication II

Advanced Authentication

Authorization

Advanced Authorization

Like Authorization

Controller Filters

Filters are methods that run *before*, *after* or *around* a controller action. To authenticate a user, we use filters to run methods before our controller actions.

Let's look at a quick example of using filters before a controller action.

```
class HellosController < ApplicationController
  before_action :say_hello

  def welcome
    end

  private

  def say_hello
    puts "Hello!"
  end
end
```

Before every action in the Hello Controller, the `say_hello` method will run, which will output "how many actions we add to this controller, say_hello will run before all of them. We have set this up because it will only act as a filter within this controller.

What if we only want our filter to run before some actions and not others?

```
class HellosController < ApplicationController
  before_action :say_hello, only: [:welcome]

  def welcome
    end

  def goodbye
    end

  def denied
    end

  private

  def say_hello
    puts "Hello!"
  end
end
```

By passing in `only: [:welcome]` to our `before_action` filter, we made it so that the `say_hello` method only runs before the `welcome` action of our `HellosController`. We could have instead passed in `except: [:goodbye, :denied]` to our filter. In that case, the `say_hello` method would run on all of the actions except `goodbye` and `denied`.

What if I have multiple controllers and I want `say_hello` to run before every action?

We could add the same `say_hello` method to each one of our controllers, but our code would be repetitive. Instead, we can use inheritance. All of the controllers inherit from the `ApplicationController`. So we can add our filter to the `ApplicationController`.

```
class ApplicationController < ActionController::Base
  before_action :say_hello

  private

  def say_hello
    puts "Hello Everybody!"
  end
end
```

Whenever any controller action is invoked, the `say_hello` method will run. The same way that our controllers inherit attributes from the `ApplicationController`, they will inherit filters as well.

Dojo Secrets

• 33%

(/m/9/4311)

Overview

(/m/

TDD

(/m/

Dojo Secrets

(/m/

Set Up

(/m/

Encryption

(/m/

User

(/m/

Authentication

(/m/

Authentication I

(/m/

User Features

(/m/

User Settings

(/m/

Secrets

(/m/

Secrets

(/m/

Like

(/m/

Secret Features

(/m/

Like Features

(/m/

Filters

(/m/

Authentication II

Advanced Authentication

Authorization

Advanced Authorization

Like Authorization

Ok, but what if I want the say_hello method to run on every action, except for two controllers?

Let's go back to the HellosController and see if we can make that happen.

```
class HellosController < ApplicationController
  skip_before_action :say_hello, except: [:welcome]

  def welcome
    end

  def goodbye
    end

  def denied
    end
end

skip_before_action allows me to remove a before filter that has been inherited from another controller. This way, we can skip the filter on specific actions. In this case, we skip the filter on the welcome action. This way, we ensure that the say_hello method is called on every action within this controller except for on the welcome action. This way, we ensure that a User is logged in throughout our website.
```

[PREVIOUS \(/M/9/4311/31525\)](#)