

# In this lecture, we will discuss...

- ✧ Strong parameters
- ✧ Flash
- ✧ How create action works

# Strong Parameters

 guides.rubyonrails.org/action\_controller\_overview.html#strong-parameters



Search

## 4.5 Strong Parameters

With strong parameters, Action Controller parameters are forbidden to be used in Active Model mass assignments until they have been whitelisted. This means you'll have to make a conscious choice about which attributes to allow for mass updating and thus prevent accidentally exposing that which shouldn't be exposed.



# create action

```
class PostsController < ApplicationController

  # POST /posts
  # POST /posts.json
  def create
    @post = Post.new(post_params)

    respond_to do |format|
      if @post.save
        format.html { redirect_to @post, notice: 'Post was successfully created.' }
        format.json { render :show, status: :created, location: @post }
      else
        format.html { render :new }
        format.json { render json: @post.errors, status: :unprocessable_entity }
      end
    end
  end
end

private
# Never trust parameters from the scary internet, only allow the white list through.
def post_params
  params.require(:post).permit(:title, :content)
end
end
```

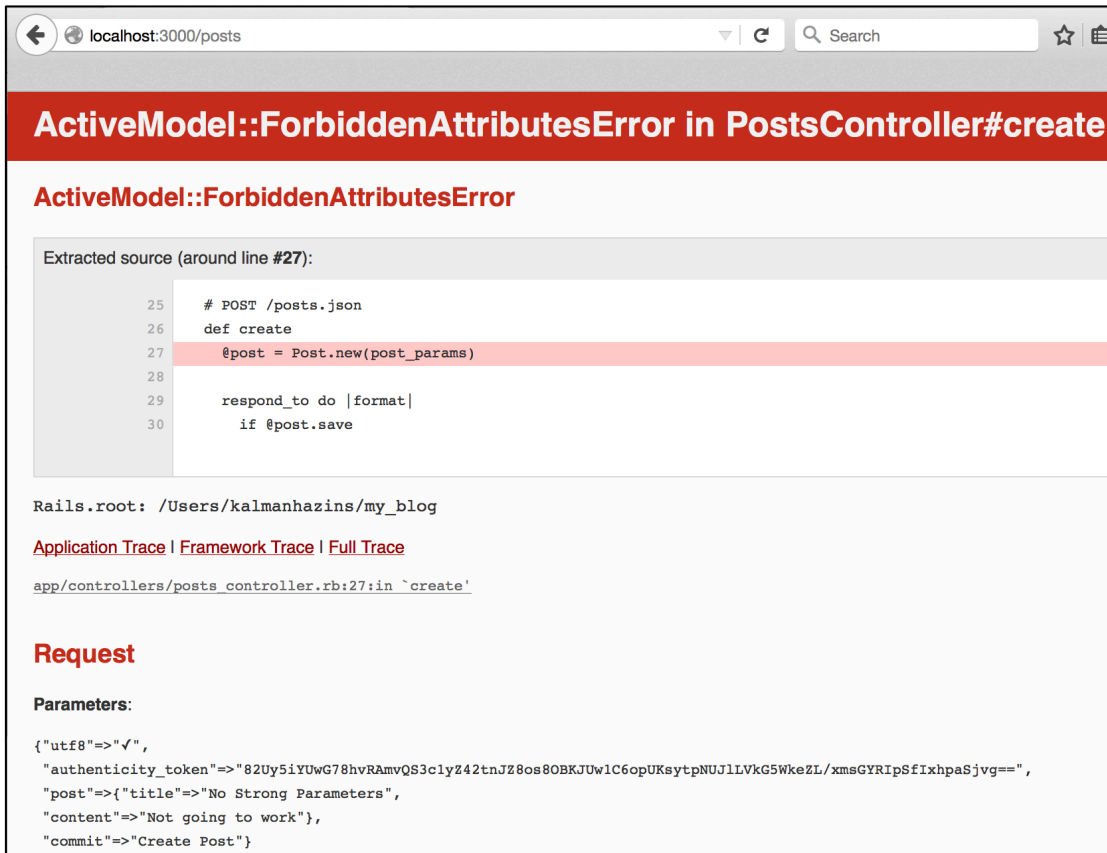


# Strong Parameters Not Implemented

```
# Never trust parameters from the scary internet, only allow the white list through.  
def post_params  
  # params.require(:post).permit(:title, :content)  
  params  
end
```



# Strong Parameters Not Implemented



A screenshot of a web browser window showing a Rails error page. The address bar indicates the URL is `localhost:3000/posts`. The error message is **ActiveModel::ForbiddenAttributesError in PostsController#create**. Below the error message, the source code around line 27 is displayed, with line 27 highlighted in red. The code shows a `POST /posts.json` request with a `create` action. Line 27 contains `@post = Post.new(post_params)`. The error occurs because the `post_params` hash contains attributes not permitted by the `Post` model's strong parameters. The browser also shows the Rails root path, application and framework traces, and the request parameters.

localhost:3000/posts

**ActiveModel::ForbiddenAttributesError in PostsController#create**

**ActiveModel::ForbiddenAttributesError**

Extracted source (around line #27):

```
25 # POST /posts.json
26 def create
27   @post = Post.new(post_params)
28
29   respond_to do |format|
30     if @post.save
```

Rails.root: /Users/kalmanhazins/my\_blog

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

`app/controllers/posts_controller.rb:27:in `create'`

**Request**

Parameters:

```
{ "utf8" => "✓",
  "authenticity_token" => "82Uy5iYUwG78hvRAMvQS3clyZ42tnJZ8os80BKJUw1C6opUKsytpNUJ1LVkG5WkeZL/xmsGYRIpSfIxhpaSjvg==",
  "post" => { "title" => "No Strong Parameters",
    "content" => "Not going to work",
    "commit" => "Create Post" }
```

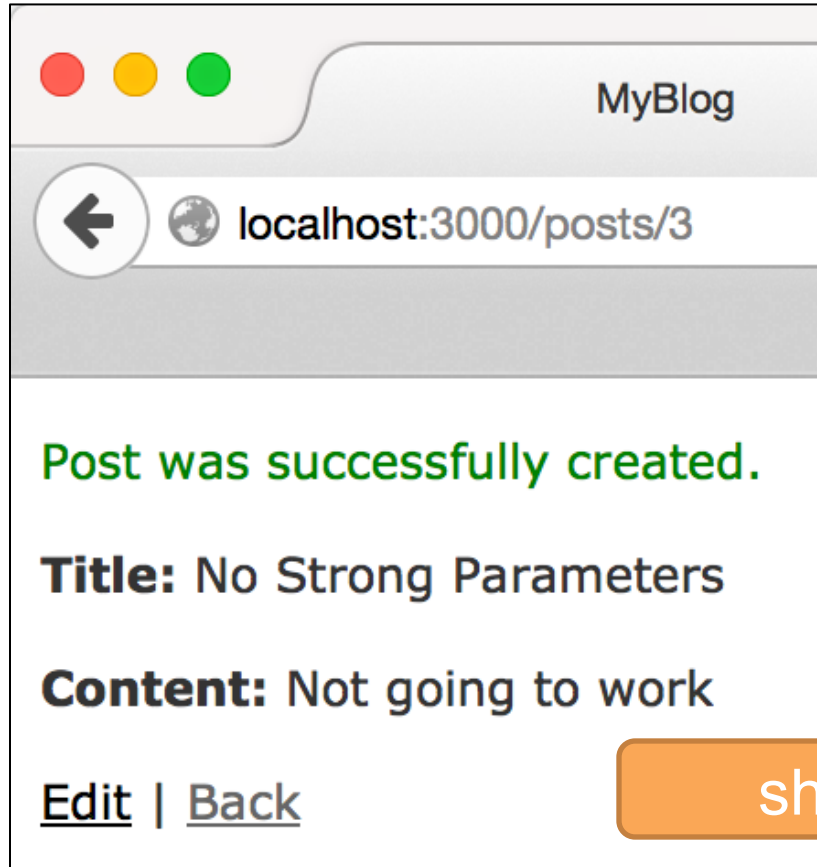
# Flash

- ✧ **Problem:** We want to *redirect* a user to a different page on our site, but at the same time *give* him some sort of a *message*? For example, “Post created!”
- ✧ **Solution:** flash – a *hash* where the data you put in *persists* for exactly *ONE request AFTER* the current request.

# Flash

- ✧ You can put your content into flash by doing `flash[:attribute] = value`
- ✧ Two very **common** attributes are `:notice` (good) and `:alert` (bad)
- ✧ These are so common in fact, that the `redirect_to` takes a `:notice` or `:alert` keys

# create action



show.html.erb (with a notice)



# Summary

- ✧ Strong parameters requires you to whitelist the parameters that you intend to create/update
- ✧ Flash persists for exactly one request after the current request/response cycle

## What's Next?

- ✧ edit and update actions