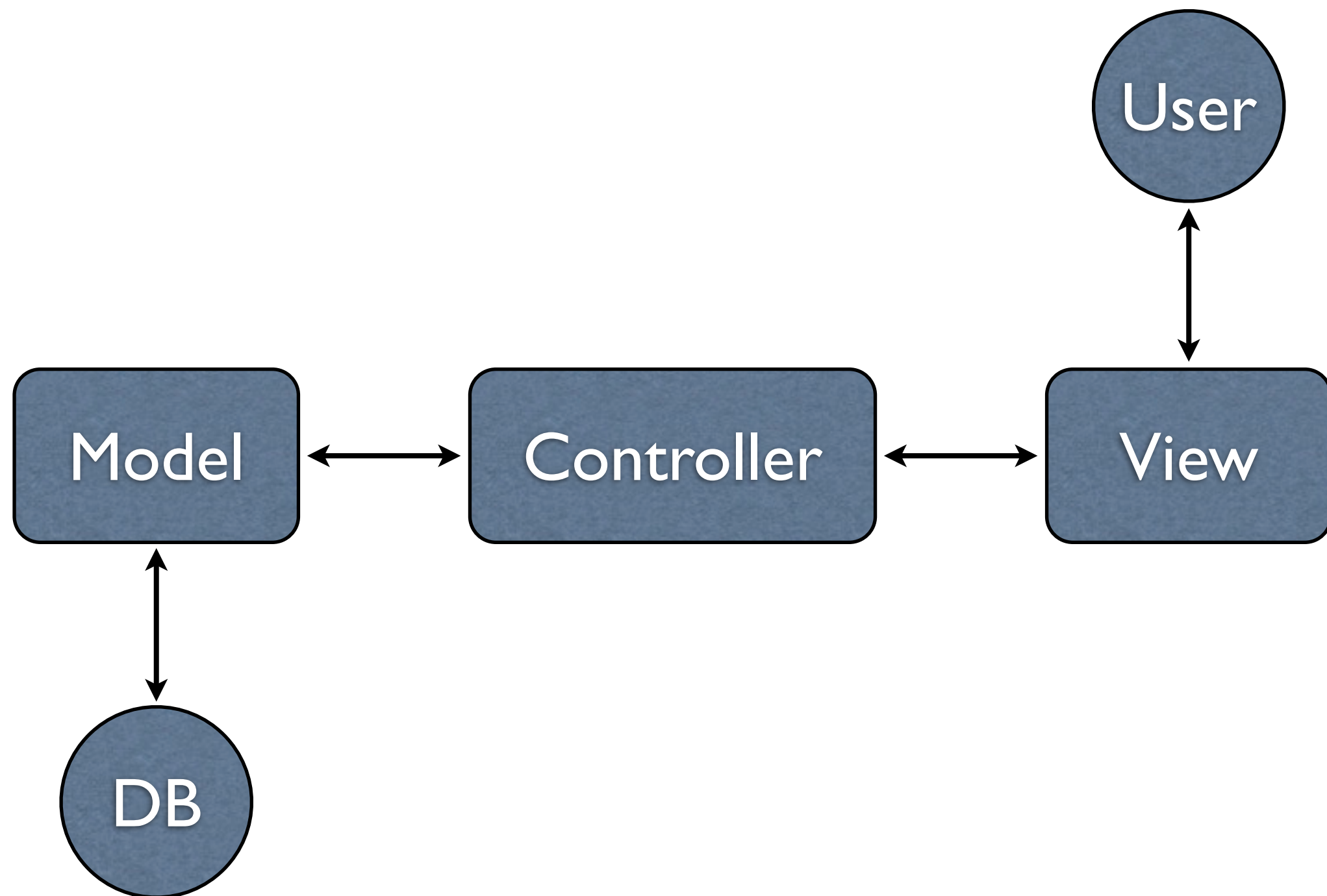


# Beginning Ruby on Rails

Session Three



# Controllers

- Controllers connect models and views
  1. Requests come in to controller
  2. Controller gets data from model
  3. Controller renders the appropriate view

# REST

Representational State Transfer

# REST

- Client-Server software architecture style introduced in 2000 by Dr. Roy Fielding
  - One of the authors of the HTTP spec
- Deals with the representation of *resources*
- Clients initiate requests to servers
- Servers process requests, return responses

# REST

- Remember CRUD?
  - Create
  - Read
  - Update
  - Delete

# REST

- Corresponding HTTP verbs:
  - Create => POST
  - Read => GET
  - Update => PUT
  - Delete => DELETE

# Routing

Connecting URLs to Code



# Resources

- Edit config/routes.rb
- Ignoring comments, it's only 3 lines:

```
Blog::Application.routes.draw do  
  resources :posts  
end
```

# Viewing Routes

- bundle exec rake routes

```
Tonys-Macbook:blog tony$ rake routes  
(in /Users/tony/Sites/rails-class/blog)
```

posts	GET	/posts(.:format)	{:action=>"index", :con
	POST	/posts(.:format)	{:action=>"create", :co
new_post	GET	/posts/new(.:format)	{:action=>"new", :contr
edit_post	GET	/posts/:id/edit(.:format)	{:action=>"edit", :cont
post	GET	/posts/:id(.:format)	{:action=>"show", :cont
	PUT	/posts/:id(.:format)	{:action=>"update", :co
	DELETE	/posts/:id(.:format)	{:action=>"destroy", :c

# Nested Resources

- Useful when one resource belongs to another resource.
- `config/routes.rb`:  

```
resources :posts do  
  resources :comments  
end
```

# Restricted Resources

- For now, we'll only worry about creating new comments.
- `config/routes.rb`:  

```
resources :posts do  
  resources :comments, :only => :create  
end
```

# Custom Routes

- Also called non-resourceful routes
- Useful for mapping old URLs to Rails or to simplify URLs for complex actions

match 'login' => 'user\_sessions#new'

match 'logout' => 'user\_sessions#destroy'

# The Root Route

- The Root Route sets the home page for your application
- Delete the file public/index.html
- Add this near the end of config/routes.rb  

```
root :to => "posts#index"
```

# Paths and URLs

- Adding a route also automatically create helpers for use in your views:

`posts_path`

`posts_url`

`new_post_path`

`new_post_url`

`edit_post_path(id)`

`edit_post_url(id)`

`post_path(id)`

`post_url(id)`

# Default Actions

Remember: “Convention Over Configuration”



# RESTful Methods

- Index
- Show
- New
- Edit
- Create
- Update
- Destroy

# RESTful Methods

- Index                List all records
- Show                Show one record
- New                Form to create a record
- Edit                Form to edit a record
- Create                Create a new record
- Update                Update an existing record
- Destroy                Delete a record

# RESTful Methods

● Index	List all records	GET
● Show	Show one record	GET
● New	Form to create a record	GET
● Edit	Form to edit a record	GET
● Create	Create a new record	POST
● Update	Update an existing record	PUT
● Destroy	Delete a record	DELETE

```
def index
  @posts = Post.all

  respond_to do |format|
    format.html # index.html.erb
    format.xml  { render :xml => @posts }
  end
end
```

```
def show
  @post = Post.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.xml  { render :xml => @post }
  end
end
```

```
def new
  @post = Post.new

  respond_to do |format|
    format.html # new.html.erb
    format.xml  { render :xml => @post }
  end
end
```

```
def edit
  @post = Post.find(params[:id])
end
```

```
def create
  @post = Post.new(params[:post])

  respond_to do |format|
    if @post.save
      format.html { redirect_to(@post,
        :notice => 'Post was successfully created.') }
      format.xml  { render :xml => @post,
        :status => :created, :location => @post }
    else
      format.html { render :action => "new" }
      format.xml  { render :xml => @post.errors,
        :status => :unprocessable_entity }
    end
  end
end
```



```
def update
  @post = Post.find(params[:id])

  respond_to do |format|
    if @post.update_attributes(params[:post])
      format.html { redirect_to(@post,
        :notice => 'Post was successfully updated.') }
      format.xml  { head :ok }
    else
      format.html { render :action => "edit" }
      format.xml  { render :xml => @post.errors,
        :status => :unprocessable_entity }
    end
  end
end
```

```
def destroy
  @post = Post.find(params[:id])
  @post.destroy

  respond_to do |format|
    format.html { redirect_to(posts_url) }
    format.xml  { head :ok }
  end
end
```

# More Controller Info

Stuff that didn't fit anywhere else

# Parameters

- Parameters are sent to the controller as a Hash named “params”
- These can be seen in the output from the rails server command

```
@post = Post.find(params[:id])
```

```
@post = Post.new(params[:post])
```

# Render / Redirect

- Every action must either render a view or redirect to another action
- Rails will look for a view with the same name as the action by default

`render :action => "edit"`

`redirect_to(posts_url)`

# Response Formats

- Rails can generate responses in other formats besides HTML
- Scaffold generated controllers include XML responses useful for creating APIs
- Other examples formats include Javascript and JSON.

# The Flash

- These are messages to the user that are only valid for a single request
- Usually styled differently to stand out
- These are usually set on a redirect:

```
redirect_to @post, :notice => 'Post was  
successfully updated.'
```

# Adding Comments

- We added a route for the create comment action earlier.
- Let's add a simple controller.
- We'll add the a form next time.



# CommentsController

- rails generate controller comments
- app/controllers/comments\_controller.rb

```
class CommentsController < ApplicationController
  def create
    @post = Post.find(params[:post_id])
    @comment = @post.comments.create(params[:comment])
    redirect_to post_path(@post)
  end
end
```