ember
ABOUT
GUIDES
API
COMMUNITY
BLOG
BUILDS
Search the guide
FORK US!
v1.10.0
GETTING STARTED
GETTING EMBER
CONCEPTS
THE OBJECT MODEL
APPLICATION
TEMPLATES
ROUTING
Introduction
Defining Your Routes
Generated Objects
Specifying a Route's Model
Setting Up a Controller
Rendering a Template
Redirecting
Specifying the URL Type
Query Parameters
Asynchronous Routing
Loading / Error Substates

INTRODUCTION

ROUTING

As users interact with your application, it moves through many different states. Ember.js gives you helpful tools for managing that state in a way that scales with your application.

To understand why this is important, imagine we are writing a web app for managing a blog. At any given time, we should be able to answer questions like: *Is the user currently logged in? Are they an admin user? What post are they looking at? Is the settings screen open? Are they editing the current post?*

In Ember.js, each of the possible states in your application is represented by a URL. Because all of the questions we asked above— *Are we logged in? What post are we looking at?* —are encapsulated by route handlers for the URLs, answering them is both simple and accurate.

At any given time, your application has one or more *active route handlers*. The active handlers can change for one of two reasons:

1. The user interacted with a view, which generated an event that caused the URL to change.
2. The user changed the URL manually (e.g., via the back button), or the page was loaded for the first time.

When the current URL changes, the newly active route handlers may do one or more of the following:

1. Conditionally redirect to a new URL.
2. Update a controller so that it represents a particular model.

ember
ABOUT
GUIDES
API
COMMUNITY
BLOG
BUILDS
Search the guide
FORK US!
v1.10.0

GETTING STARTED
GETTING EMBER
CONCEPTS
THE OBJECT MODEL
APPLICATION
TEMPLATES
ROUTING
Introduction
Defining Your Routes
Generated Objects
Specifying a Route's Model
Setting Up a Controller
Rendering a Template
Redirecting
Specifying the URL Type
Query Parameters
Asynchronous Routing
Loading / Error Substates

INTRODUCTION

ROUTING

As users interact with your application, it moves through many different states. Ember.js gives you helpful tools for managing that state in a way that scales with your application.

To understand why this is important, imagine we are writing a web app for managing a blog. At any given time, we should be able to answer questions like: *Is the user currently logged in? Are they an admin user? What post are they looking at? Is the settings screen open? Are they editing the current post?*

In Ember.js, each of the possible states in your application is represented by a URL. Because all of the questions we asked above— *Are we logged in? What post are we looking at?* —are encapsulated by route handlers for the URLs, answering them is both simple and accurate.

At any given time, your application has one or more *active route handlers*. The active handlers can change for one of two reasons:

1. The user interacted with a view, which generated an event that caused the URL to change.
2. The user changed the URL manually (e.g., via the back button), or the page was loaded for the first time.

When the current URL changes, the newly active route handlers may do one or more of the following:

1. Conditionally redirect to a new URL.
2. Update a controller so that it represents a particular model.

Information Testing

betta.io

WB