Budapest 2016

Ember.js Training

Joachim Haagen Skeie
Alex Speller

Thank you for your interest in Ember.js!

About us

Joachim Haagen Skeie

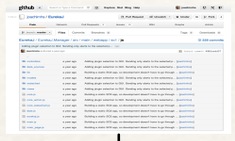
- Owner @ Haagen Software, and Kodegenet
- Author of Ember.js in Action
- Author of various programming books for children (Norwegian)
- Organizer of Ember Fest

Alex Speller

What is Ember.js?

New York Times

GitHub



Trello

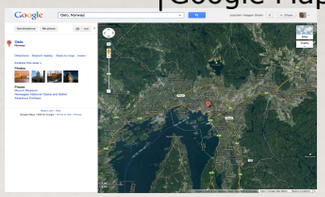
Traditional websites

Web Applications

Apple

Mint.com

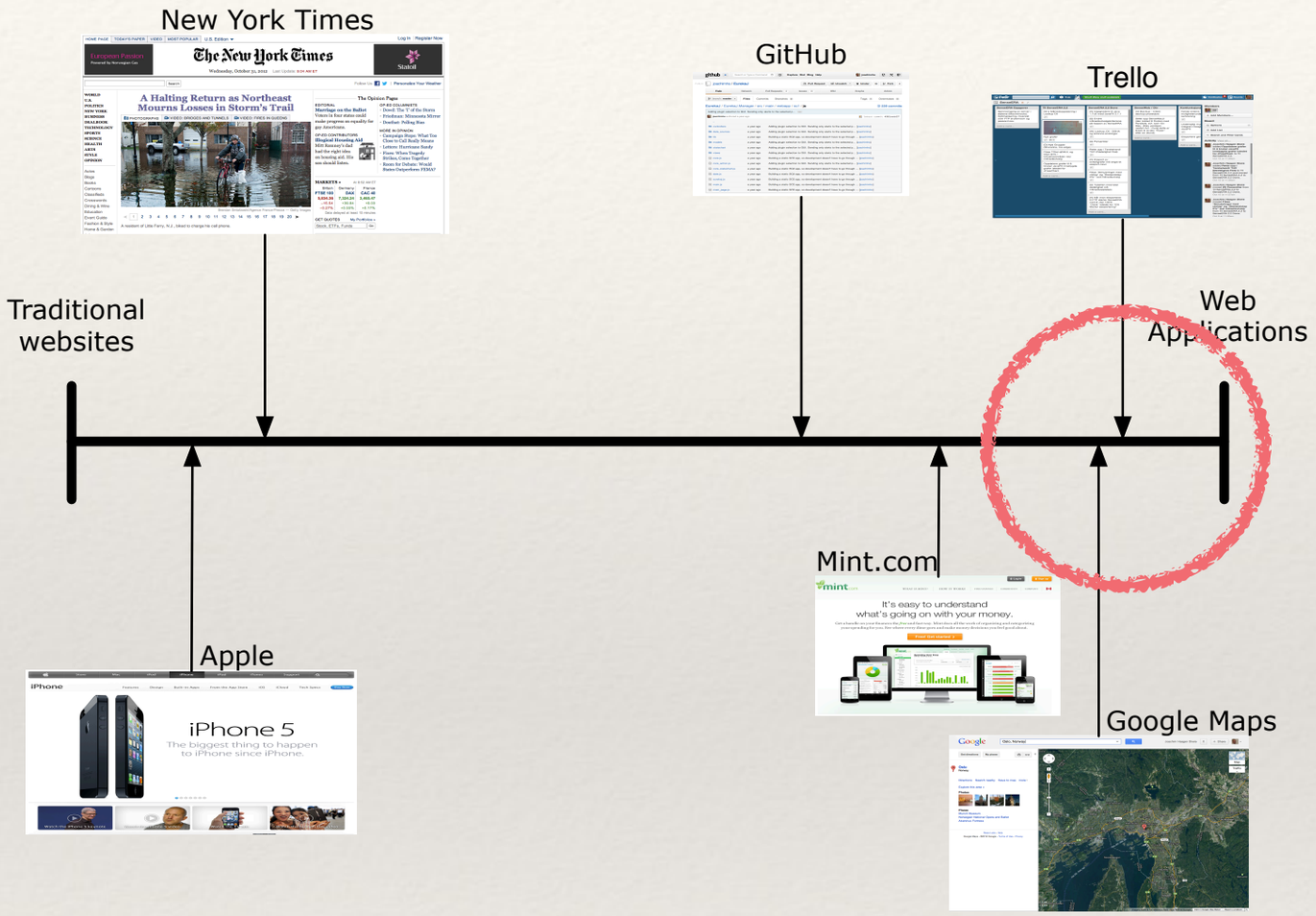
Google Maps



- HTML, CSS and JavaScript serialised on the server-side and passed to the server
- Full-page refresh when the user clicks on links
- Heavily reliant on the traditional HTTP Request-response lifecycle
- Easy to cache the website pages on the server

- Full JavaScript application sent on the first request
- Subsequent requests only transmit data
- Usually have a rich application-like user interface
- Faster navigation and user interaction
- Supports the rich feature set that users have come to expect from modern web based applications

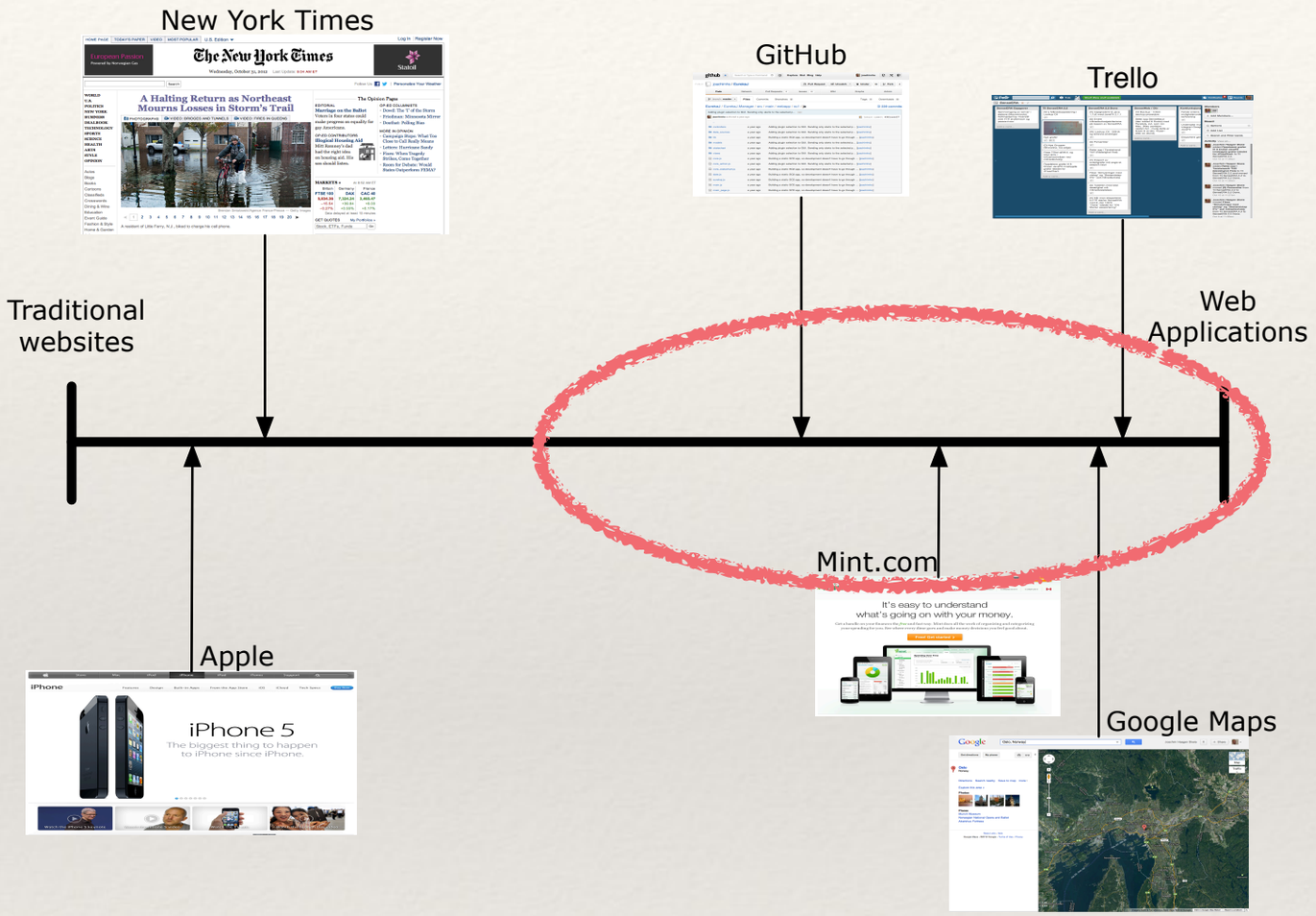
What is Ember.js?



- HTML, CSS and JavaScript serialised on the server-side and passed to the server
- Full-page refresh when the user clicks on links
- Heavily reliant on the traditional HTTP Request-response lifecycle
- Easy to cache the website pages on the server

- Full JavaScript application sent on the first request
- Subsequent requests only transmit data
- Usually have a rich application-like user interface
- Faster navigation and user interaction
- Supports the rich feature set that users have come to expect from modern web based applications

What is Ember.js?



- HTML, CSS and JavaScript serialised on the server-side and passed to the server
- Full-page refresh when the user clicks on links
- Heavily reliant on the traditional HTTP Request-response lifecycle
- Easy to cache the website pages on the server

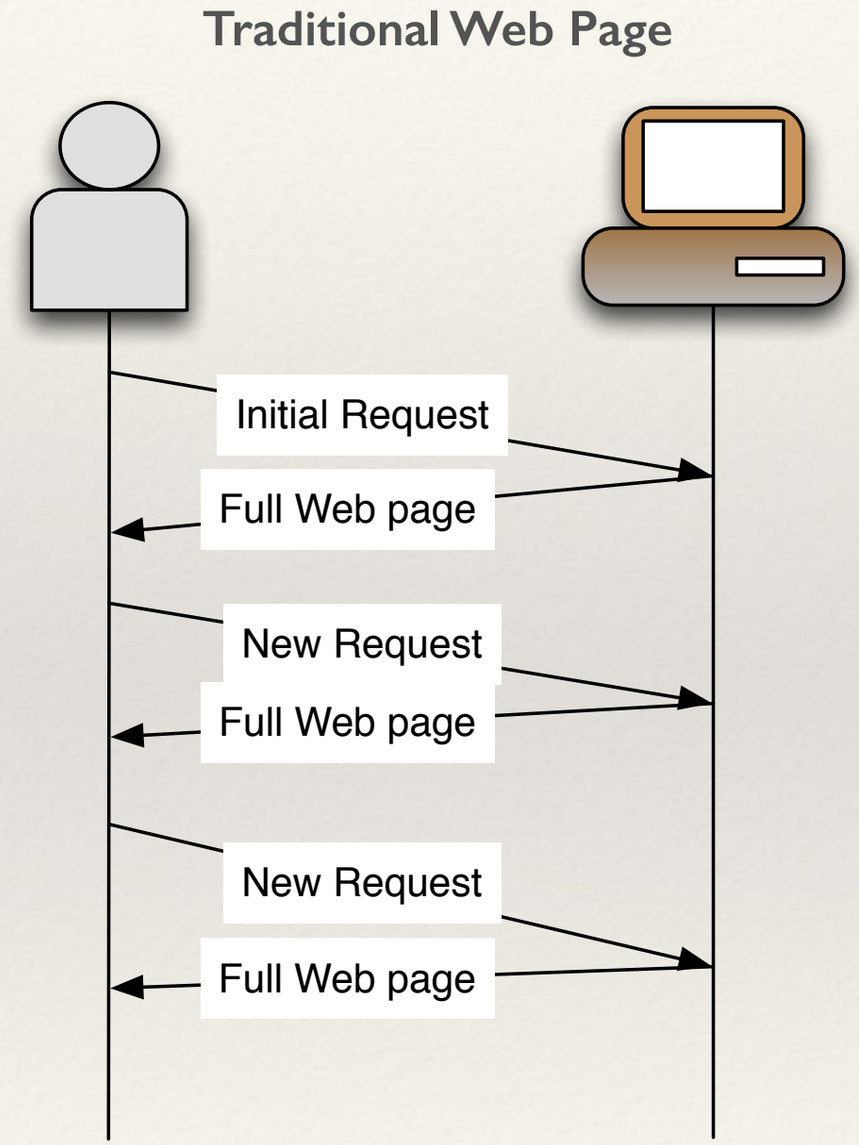
- Full JavaScript application sent on the first request
- Subsequent requests only transmit data
- Usually have a rich application-like user interface
- Faster navigation and user interaction
- Supports the rich feature set that users have come to expect from modern web based applications

What is Ember.js

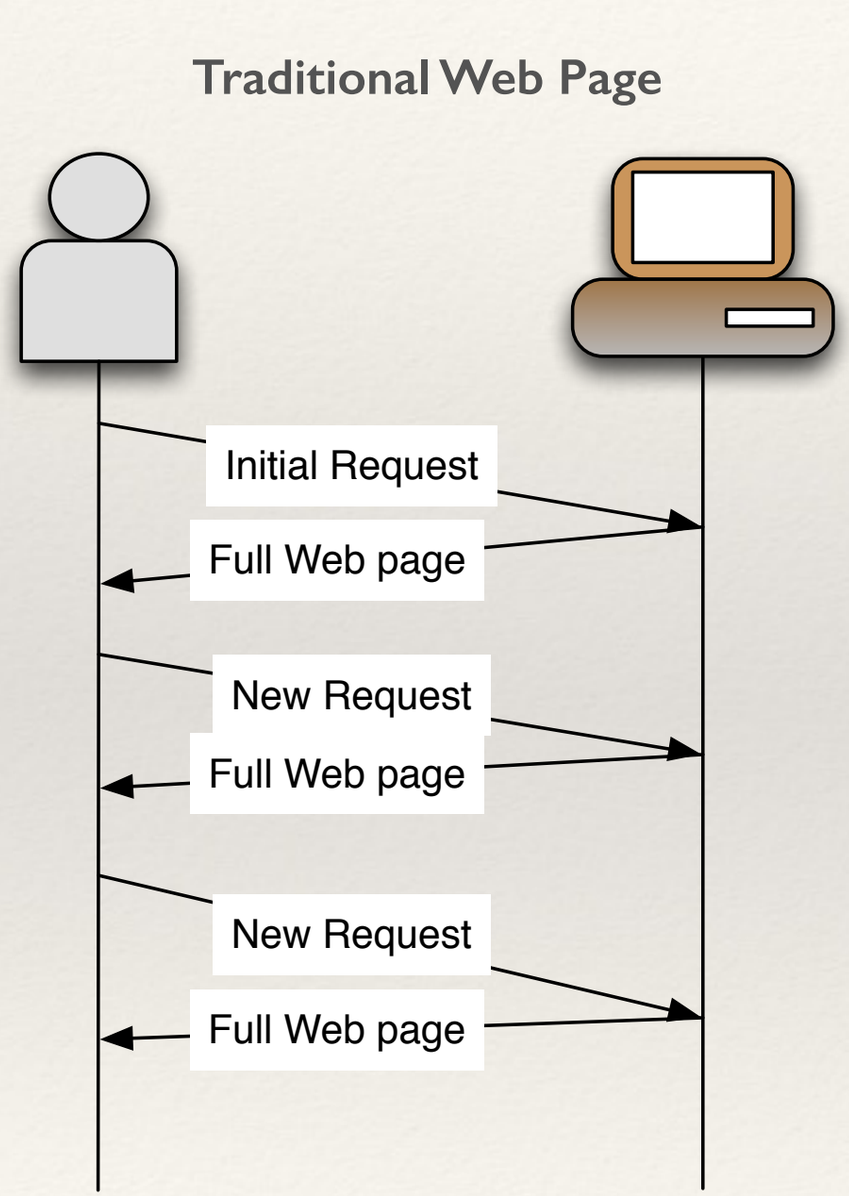
Websites built with Ember.js are generally

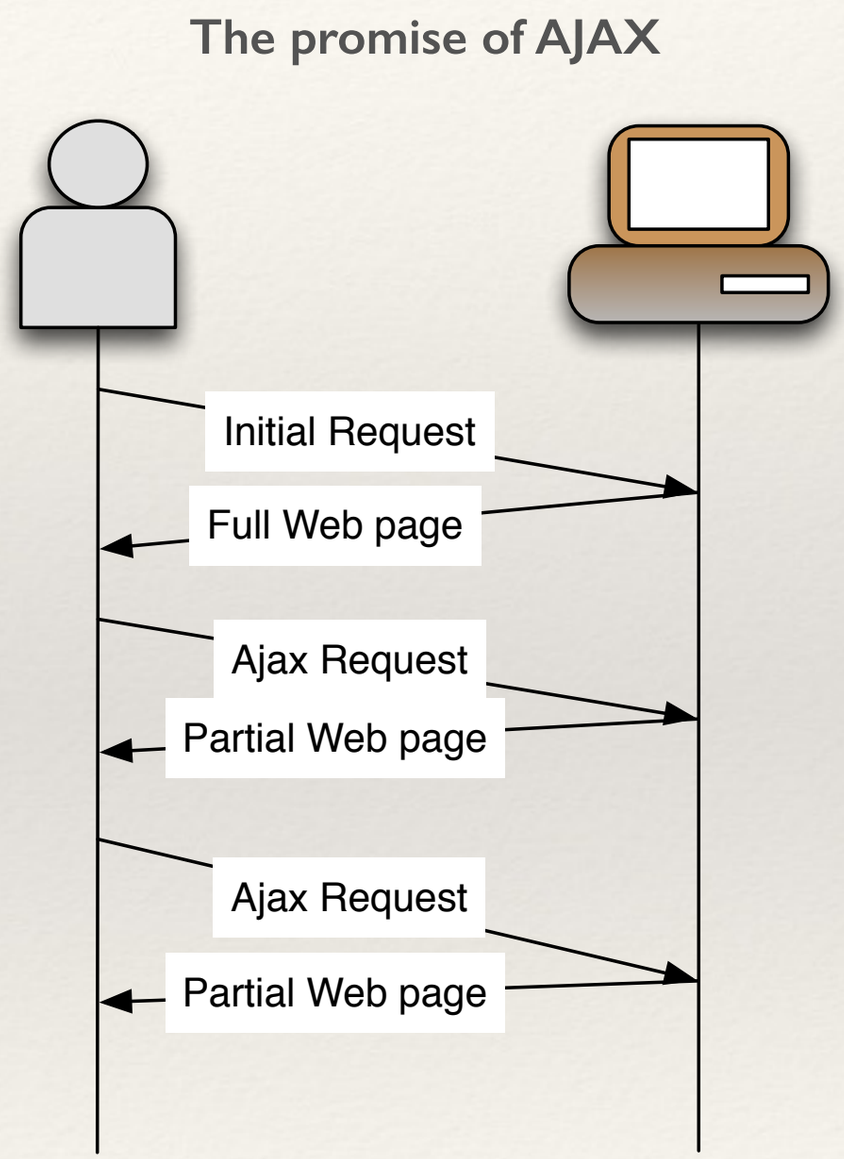
- Single Page Applications (SPAs)
- Highly interactive
- Often feels like a real application
- Snappy to use (after initial loading and bootstrapping)
- Have that Wow-factor your users are becoming expected to

Out with the old, in with the new

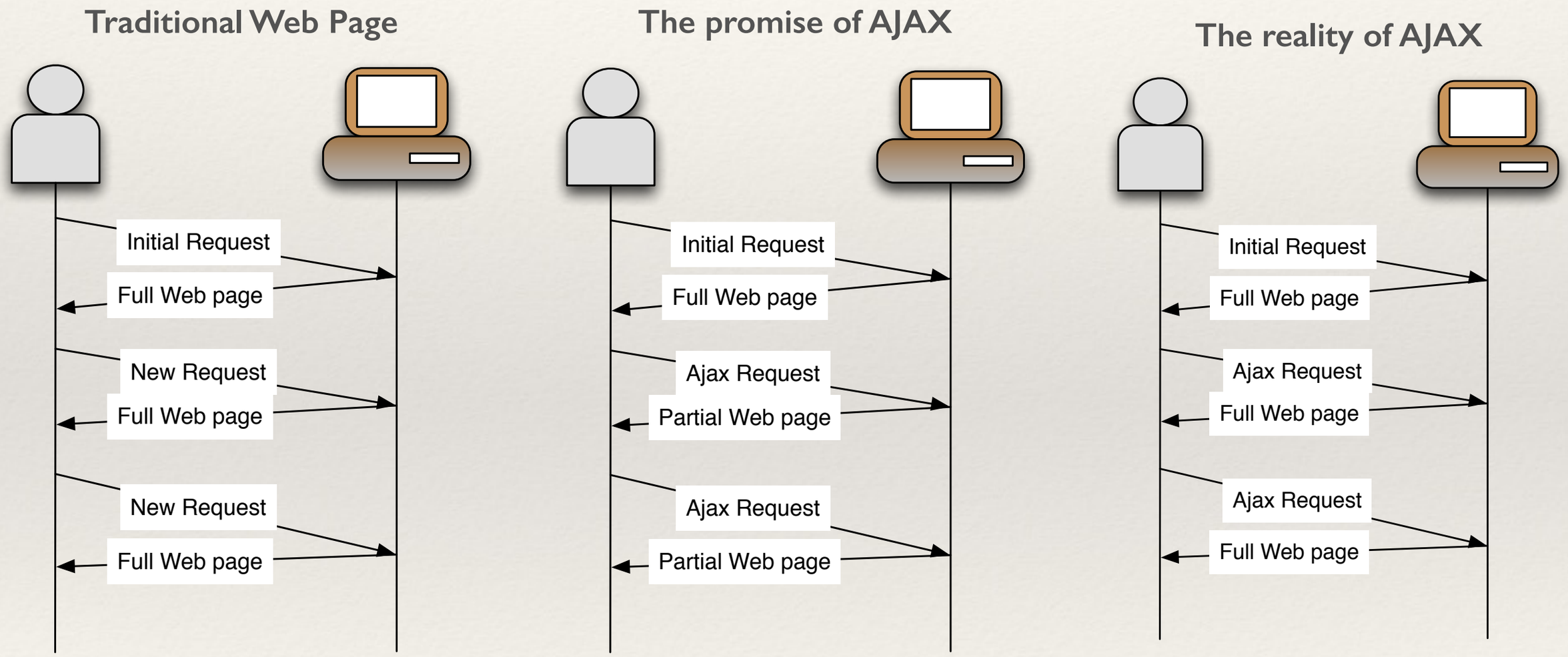


Out with the old, in with the new

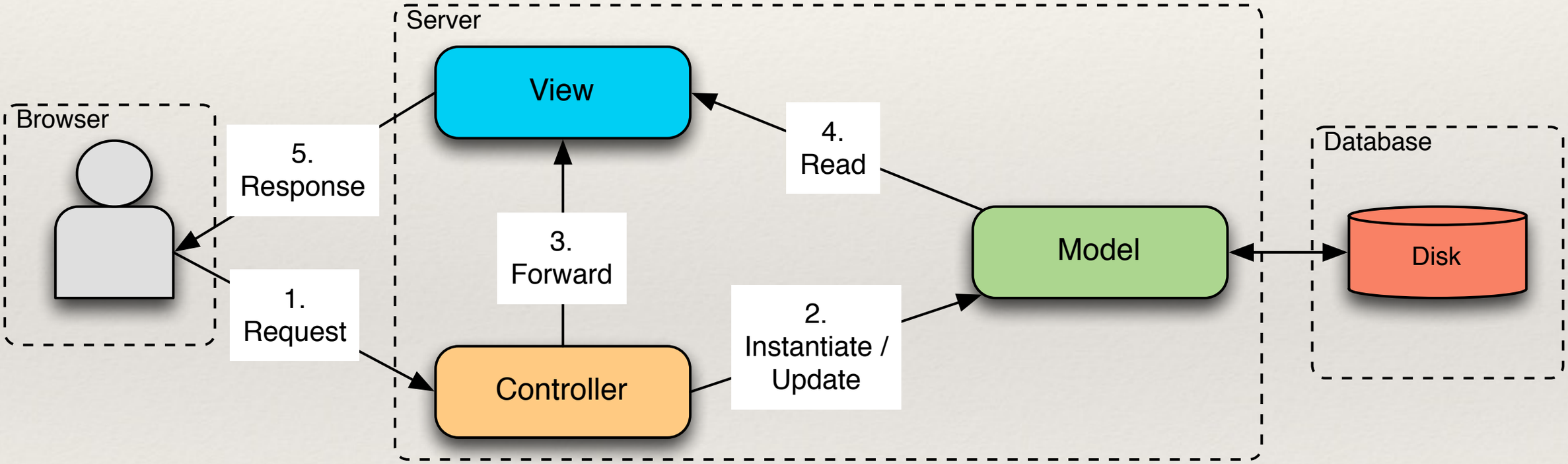


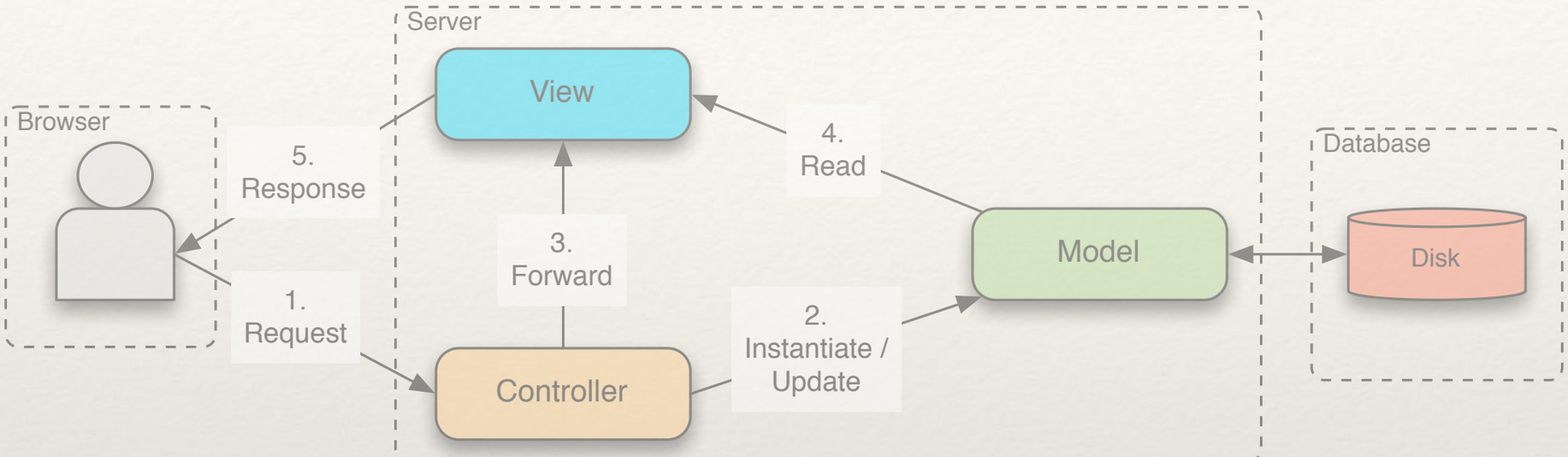


Out with the old, in with the new



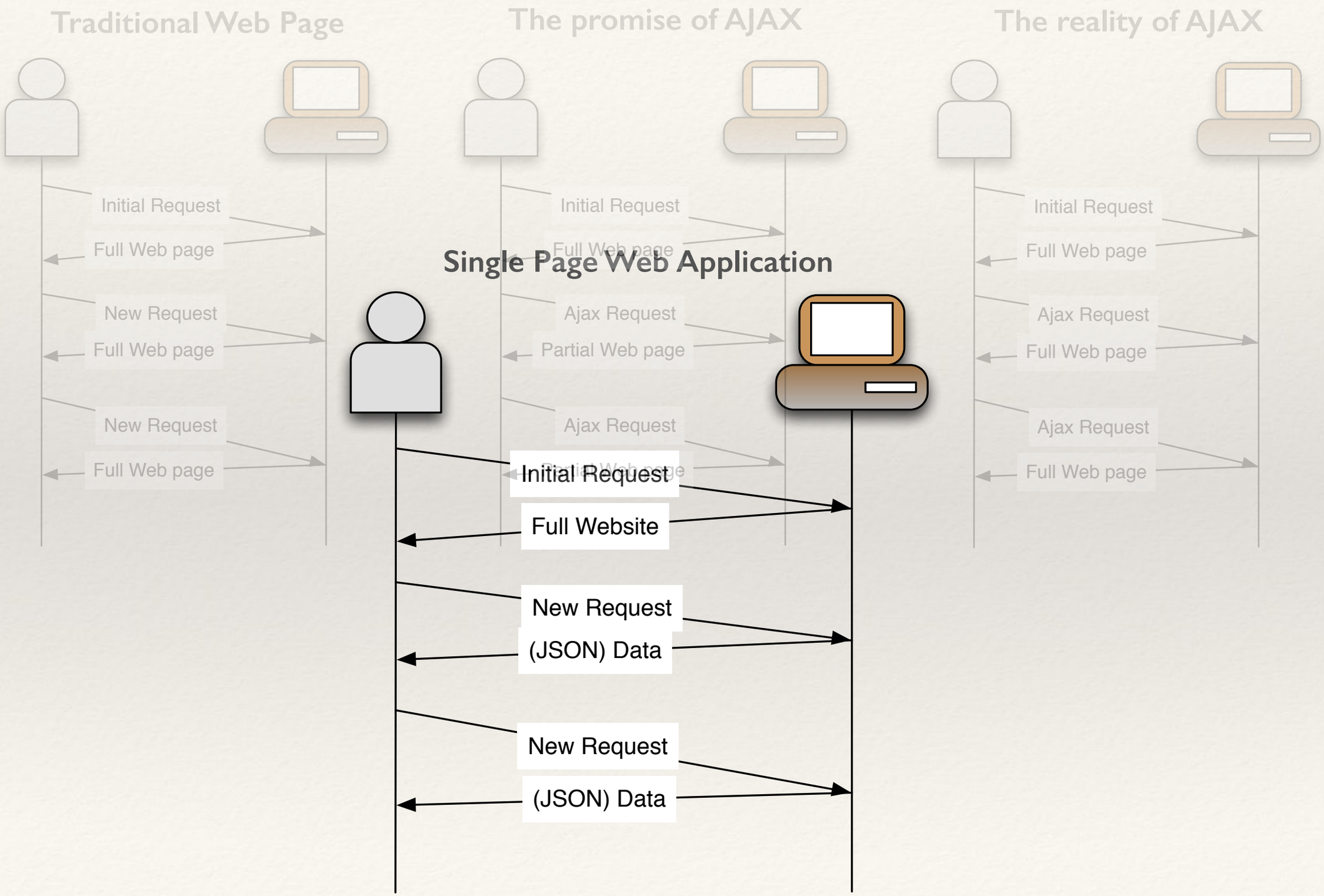
The AJAX MVC Pattern





- There is no little dynamic part in this model
- Relies upon the server generating JavaScript and HTML, which it passes to the client for each request
- Keeps both the application state and each user's state on the server
- Client-side state and server-side state needs to be in-sync
- If the user requests data to be updated, a complete new page is generated and transferred over the wire to the browser. The browser picks which part of the DOM to update
- Tough to scale horizontally

Out with the old, in with the new



The Ember.js MVC Pattern

Templates

- Responsible for rendering elements
- May use templates to define User Interface
- Responsible for reacting to user actions

The Ember.js MVC Pattern

Templates

- Responsible for rendering elements
- May use templates to define User Interface
- Responsible for reacting to user actions

Controller / Router

- Serves as a link between the view and the model
- Groups related models into defined data structures

The Ember.js MVC Pattern

Templates

- Responsible for rendering elements
- May use templates to define User Interface
- Responsible for reacting to user actions

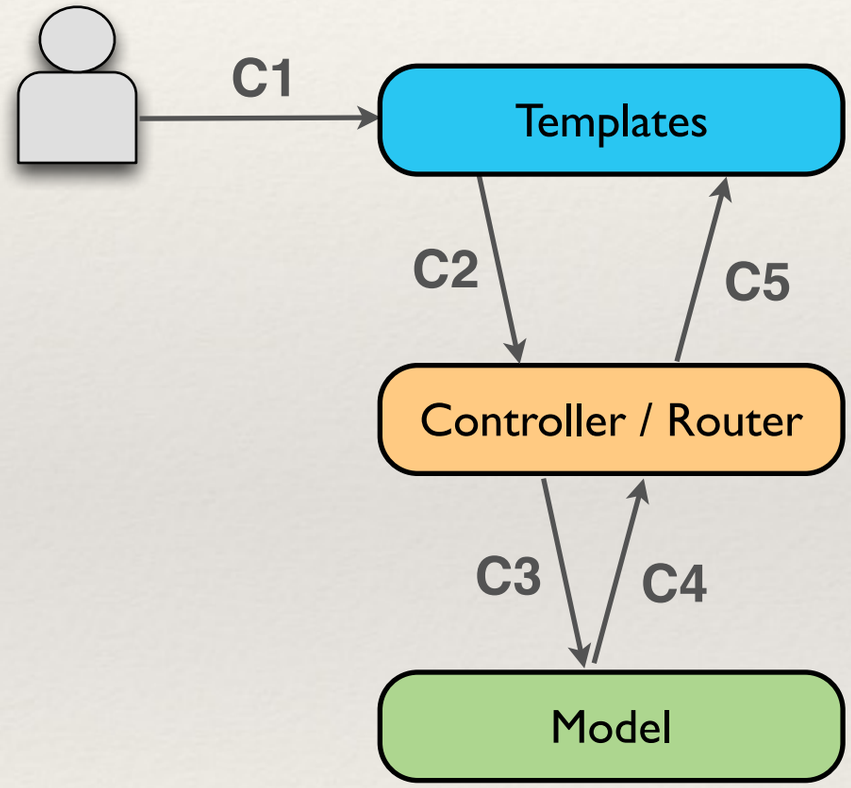
Controller / Router

- Serves as a link between the view and the model
- Groups related models into defined data structures

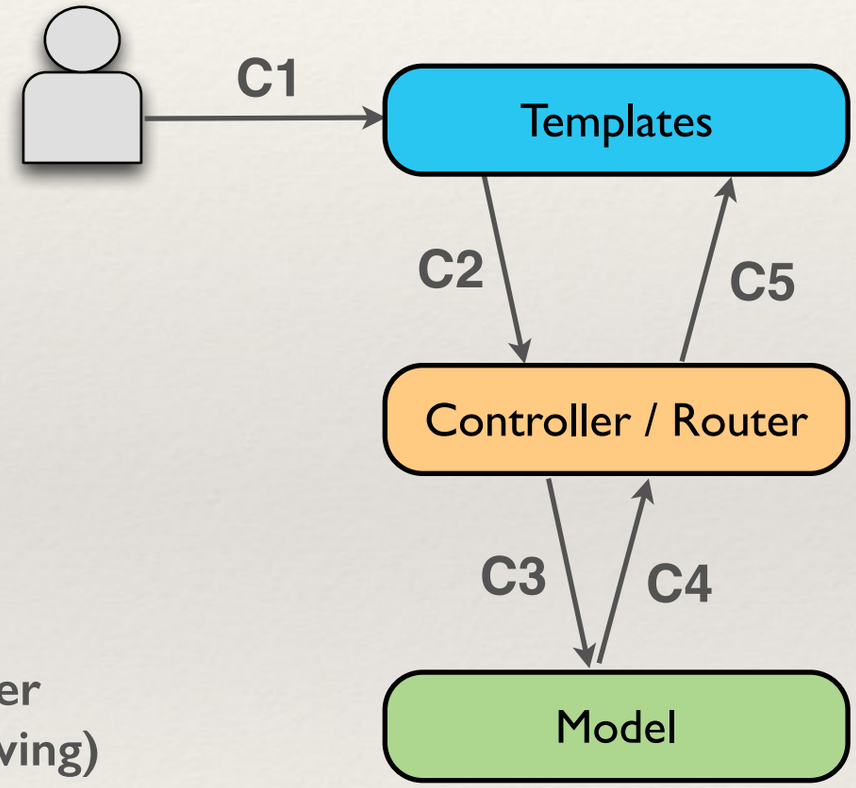
Model

- Holds the data for the application
- Responsible for most data formatting
- Responsible for client-server communication

The Ember.js MVC Pattern

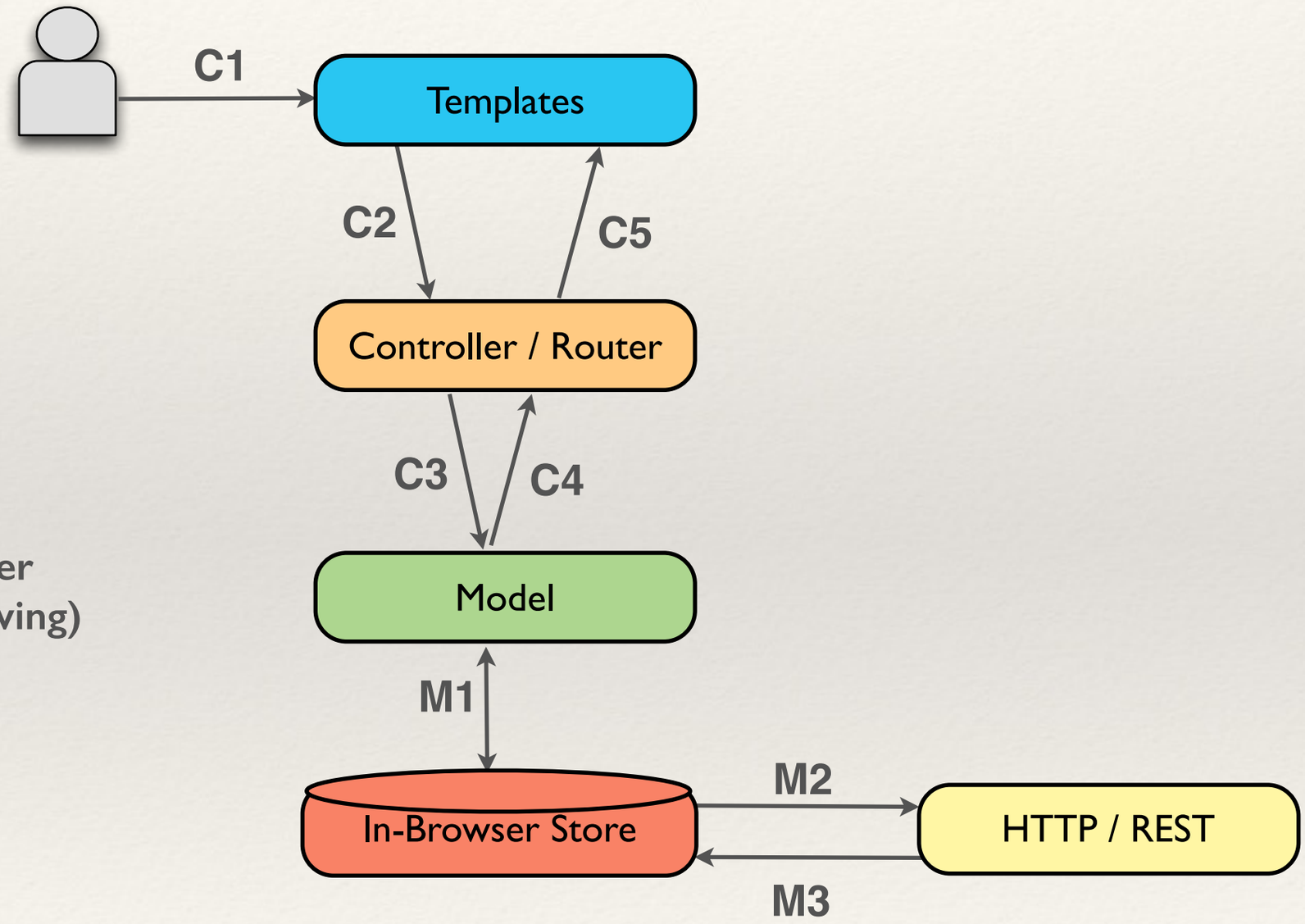


The Ember.js MVC Pattern



C1: User Event
C2: Controller Action
C3: Send Event to Controller
C4: Model Mapping (Observing)
C5: View Redraw

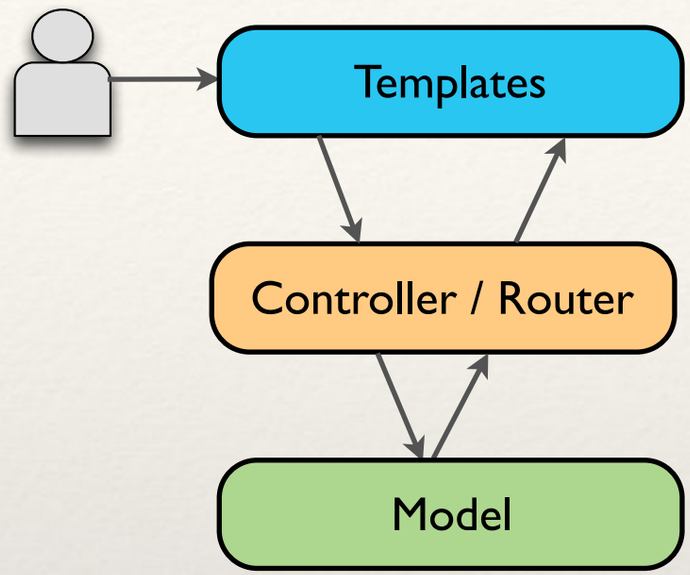
The Ember.js MVC Pattern



C1: User Event
C2: Controller Action
C3: Send Event to Controller
C4: Model Mapping (Observing)
C5: View Redraw

M1: Fetch from Cache
M2: Request Missing Data
M3: Update Cache

The Ember.js MVC Pattern

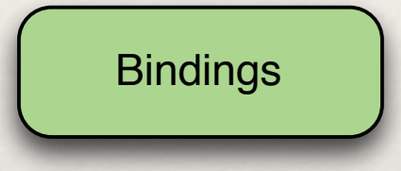


- The pattern is fully dynamic
- NO markup of scripts are transferred from the server-side
- Both application and user state is on the Client-side
- Frees server up to do what it does best - Serving data
- Any UI change is done on the Client-side via direct DOM manipulations
- Easier to scale horizontally
- Client and Server concerns are completely seperated

Demo Time

Bootstrapping the PhotoApp application

Bindings, Computed Properties and Observers



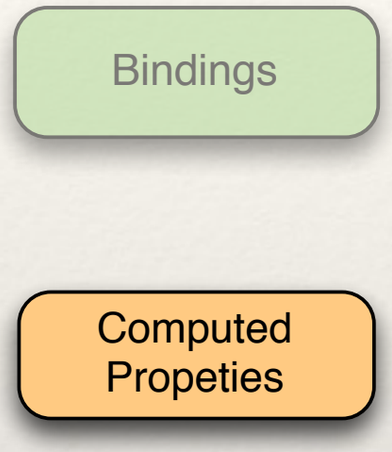
Ensures variables are updated between objects

Used to synchronize values across layers in your application

One way or Two way bindings

Mostly used in templates, {{model.id}}, and in Ember.computed.xxx functions

Bindings, Computed Properties and Observers



Used to create properties based on aggregations or calculations

Are kept up-to-date whenever the data it depends on change

Can be used as a regular property in your templates

```
isSelected: function() {
    return this.get('controller.selectedPhoto.id') === this.get('model.id');
}.property('controller.selectedPhoto.id', 'model.id'),
```

Bindings, Computed Properties and Observers

Bindings

Computed Propeties

Observers

Used to perform logic when data change

Normal functions

Used to fire data-bound events

Bindings, Computed Properties and Observers

Bindings

Computed Propeties

Observers

Used to inject Ember objects

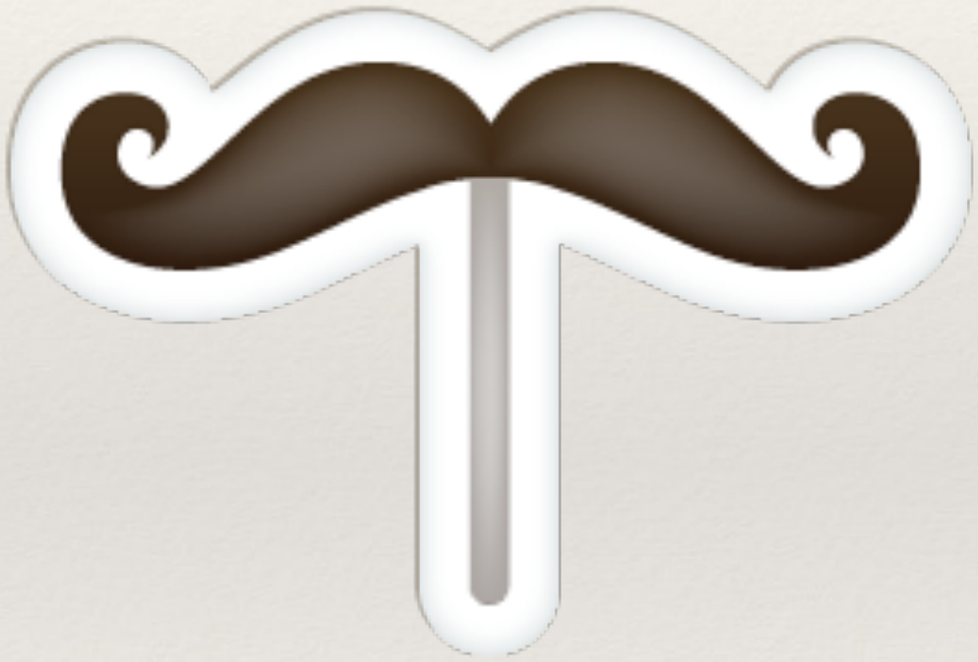
Injections

Access one controller from another

Access Ember services

What is Handlebars.js?

handlebars



What is Handlebars.js?

A JavaScript-based template engine that allows you to write complex templates in a familiar HTML-esque style

Ember.js uses Handlebars.js in order to enrich the view-layer with semantic templates

Contains both simple expressions and block expressions that allow you to express template-based logic

Logic expressions is also possible since Ember 2.0

Simple expressions

Simple expressions are identifiers that tell Handlebars.js to replace the contents of the expression at runtime

```
<h1>{{title}}</h1>
```

Expressions can traverse into your object-structure via dot-notation

```
<h1>{{book.title}}</h1>
```

Simple expressions can also render complex templates, and have properties and property-bindings

```
{{render "header"}}
```

```
{{my-component image=controller.imageUrl label="Hello"}}
```

Simple expressions

Handlebars.js ships with a range of simple expressions

{{view}}	Renders a view
{{bind-attr}}	Renders a DOM element attribute
{{action}}	Triggers an action on click
{{outlet}}	Renders a sub route
{{unbound}}	Renders an expression that won't be backed by a two-way binding
{{render}}	Render a template backed by a controller and a view
{{partial}}	Render a template with the current controllers' context
{{component}}	Render a component

Block Expressions

A block expression is an expression that, in addition to having a value, also has a body that can contain plain markup, simple expressions, other block expressions, as well as a combination of the above

```
{{#each book in book}}
    <h1>{{book.title}}</h1>
{{/each}}
```

The start of the block expression

The body of the block expression

The end of of the block expression

Handlebars.js ships with the most commonly used expressions

{{#if}} and {{#if}} ... {{else}} ... {{/if}}	Renders the contents based on a condition
{{#unless}} (which is !if)	Renders the contents if the expression is false
{{#with}}	Changes the scope of the content
{{#each}}	Renders the body for each item in a list
{{#link-to}}	Creates a link to another route in the application
{{! comment}}	Specifies a comment, which wont be rendered

Handlebars example

The application template

{{header-component}}

    {{outlet}}

{{footer-component}}

Creating your own expressions

Using **Handlebars.registerHelper** you can define your own expressions

```
Ember.Handlebars.registerBoundHelper('markdown', function(property) {
    var converter = new Showdown.converter();
    if (property !== null) {
        return new Handlebars.SafeString(converter.makeHtml(property));
    }
});
```

Then, you can use that helper as any other expression

{{markdown property}}

Demo Time

Creating albums and album routes

Ember Router

The Backbone of your application

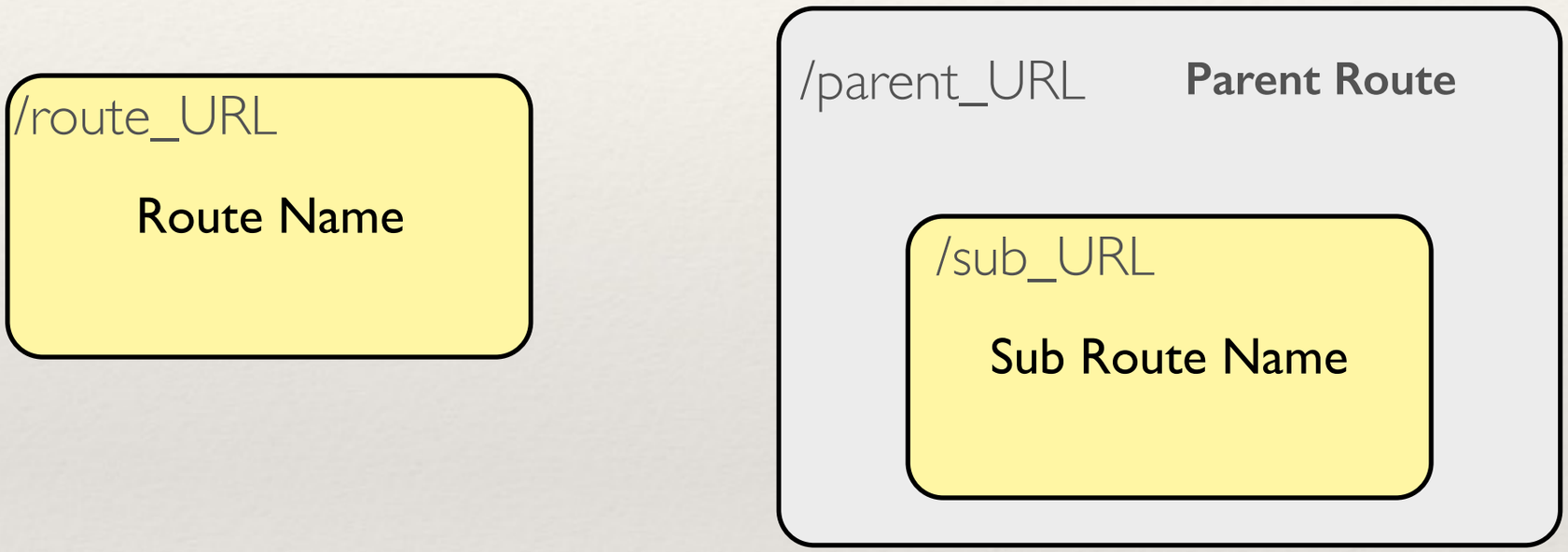
Defines the logical path that your users are able to navigate through your application

Cleanly defines your applications URL endpoints

A hierarchical structure, implemented as a Statechart

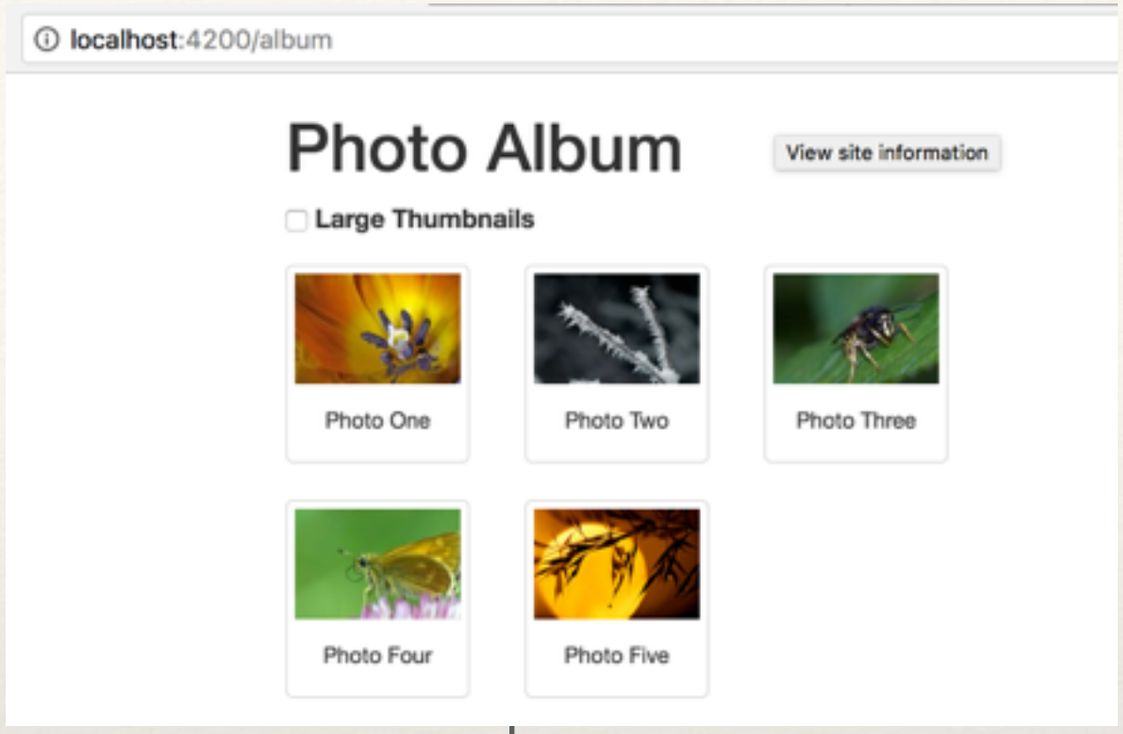
Consists of **routes**, which are the leaf-routes in the hierarchical **structure** and **resources**, which can contain other **resources** or **routes**.

Ember Router

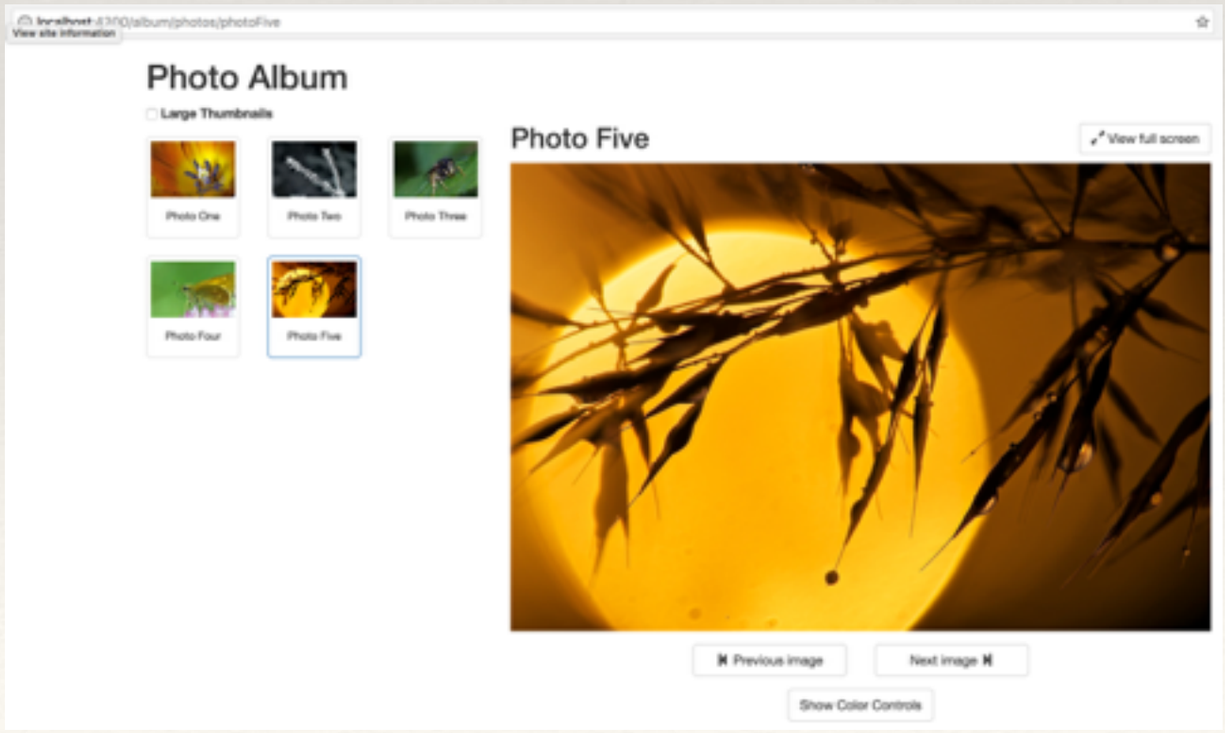


Ember Router

album



album.photo



Ember Router

app/router.js

```
Router.map(function() {
  this.route('album', function() {
    this.route('photo', {path: '/photos/:photo_id'});
  });

  this.route('lightbox', {path: '/lightbox/:photo_id'});
});
```

Ember Router

this.resource("**album**", {path: "/album"}, function() {})

URL: /album

/album

album

Template — **app/templates/album.hbs**

Route — **app/routes/album.js**

Controller — **app/controllers/album.js**

Model — **this.store.findAll('photo')**

ember generate route album
ember generate controller album
ember generate model photo

Ember Router

```
PhotoApp.Router.map(function() {
    this.resource('album', {path: "/album"}, function() {
        this.route('photo', {path: "/album/:photo_id"});
    });
});
```

URL: /album/photo/:photo_id

Template **app/templates/album/photo.hbs**

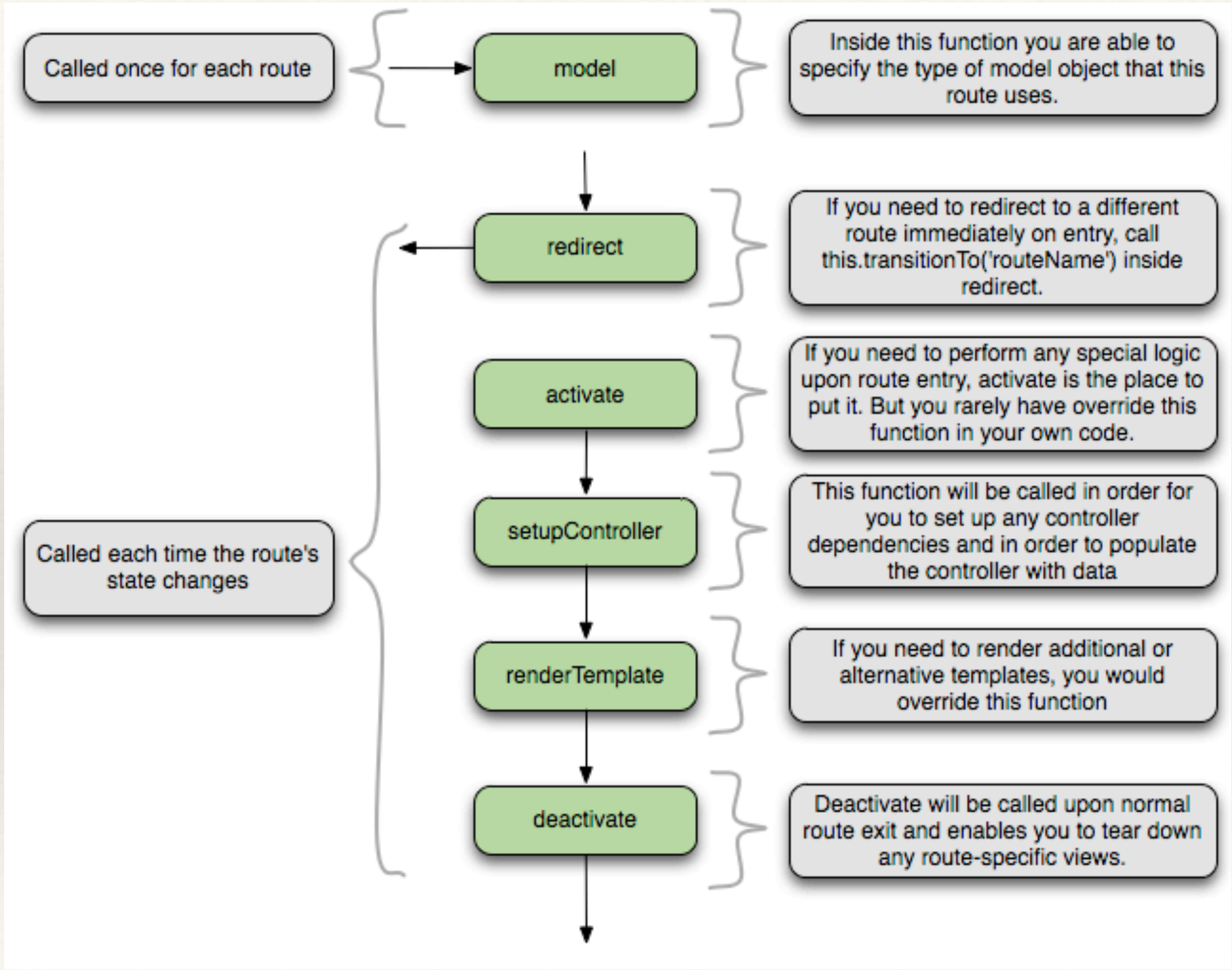
Route **app/routes/album/photo.js**

Controller **app/controllers/album/photo.js**

Model **this.store.find('photo', photo_id)**

ember generate route album/photo
ember generate controller album/photo

Ember Router



Demo Time

Mocking data for the albums and album routes

What is an Identity Map?

Summarized, an Identity Map is an in-app cache that maps a unique identifier to a single, object

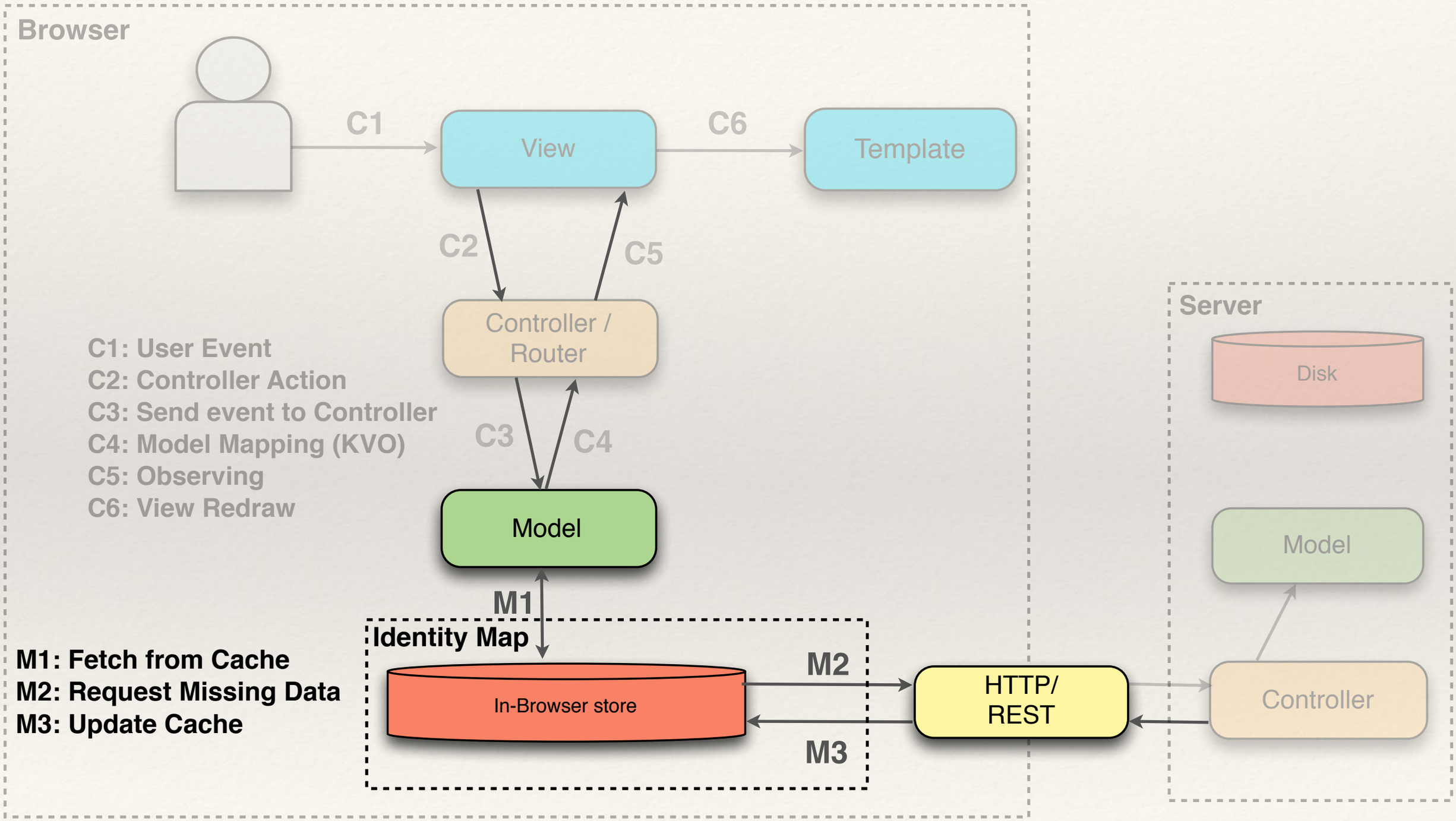
An Identity Map guarantees that there is only one loaded object in the cache for any unique identifier at any one time

```
var objOne = identityMap.get('abc');
var objTwo = identityMap.get('abc');
objOne == objTwo    //true
```

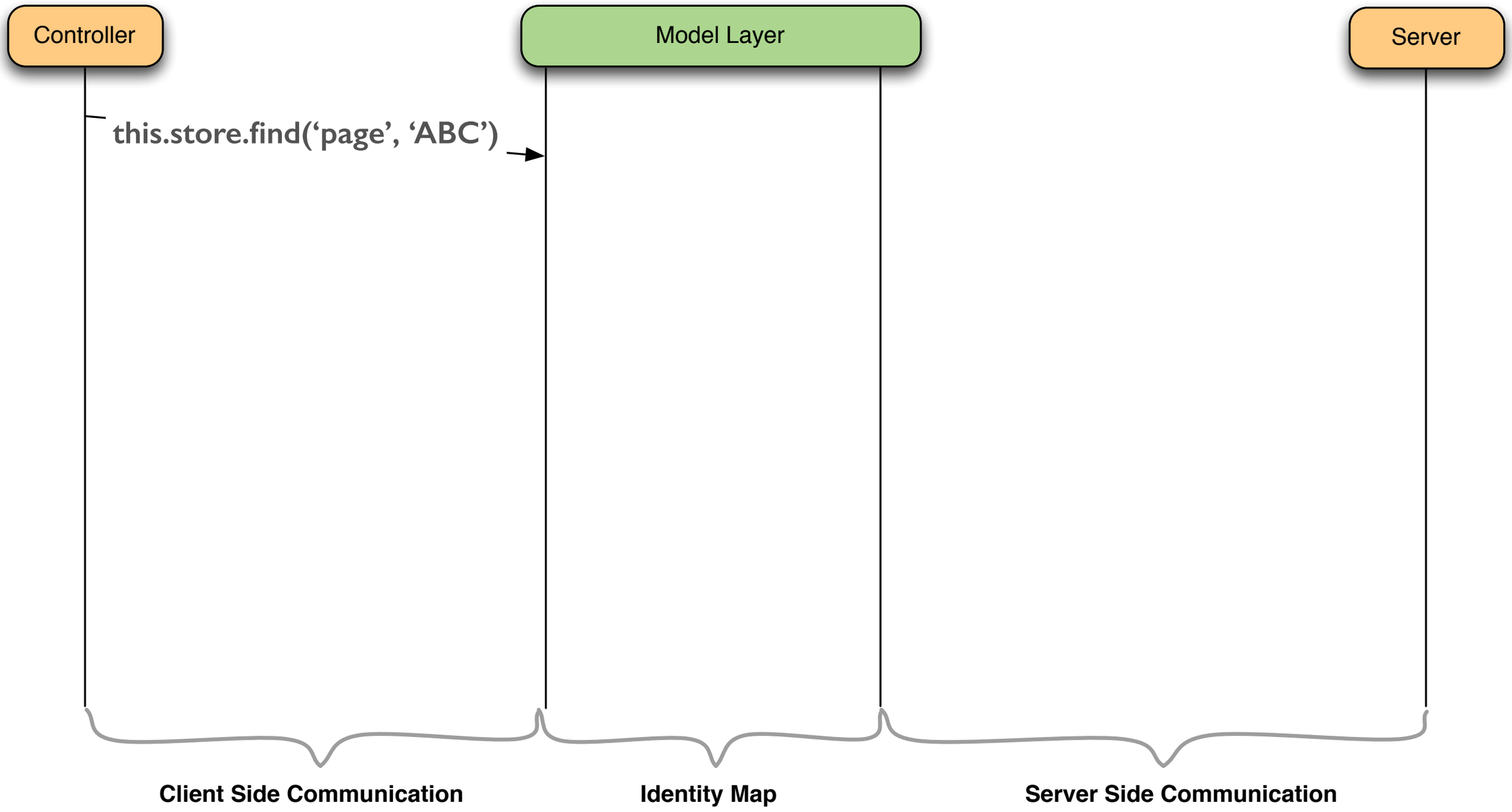
In asynchronous applications, the identity map guarantees the above, even between server-roundtrips

The Ember Router is optimized for an Identity-backed Client-Server implementation

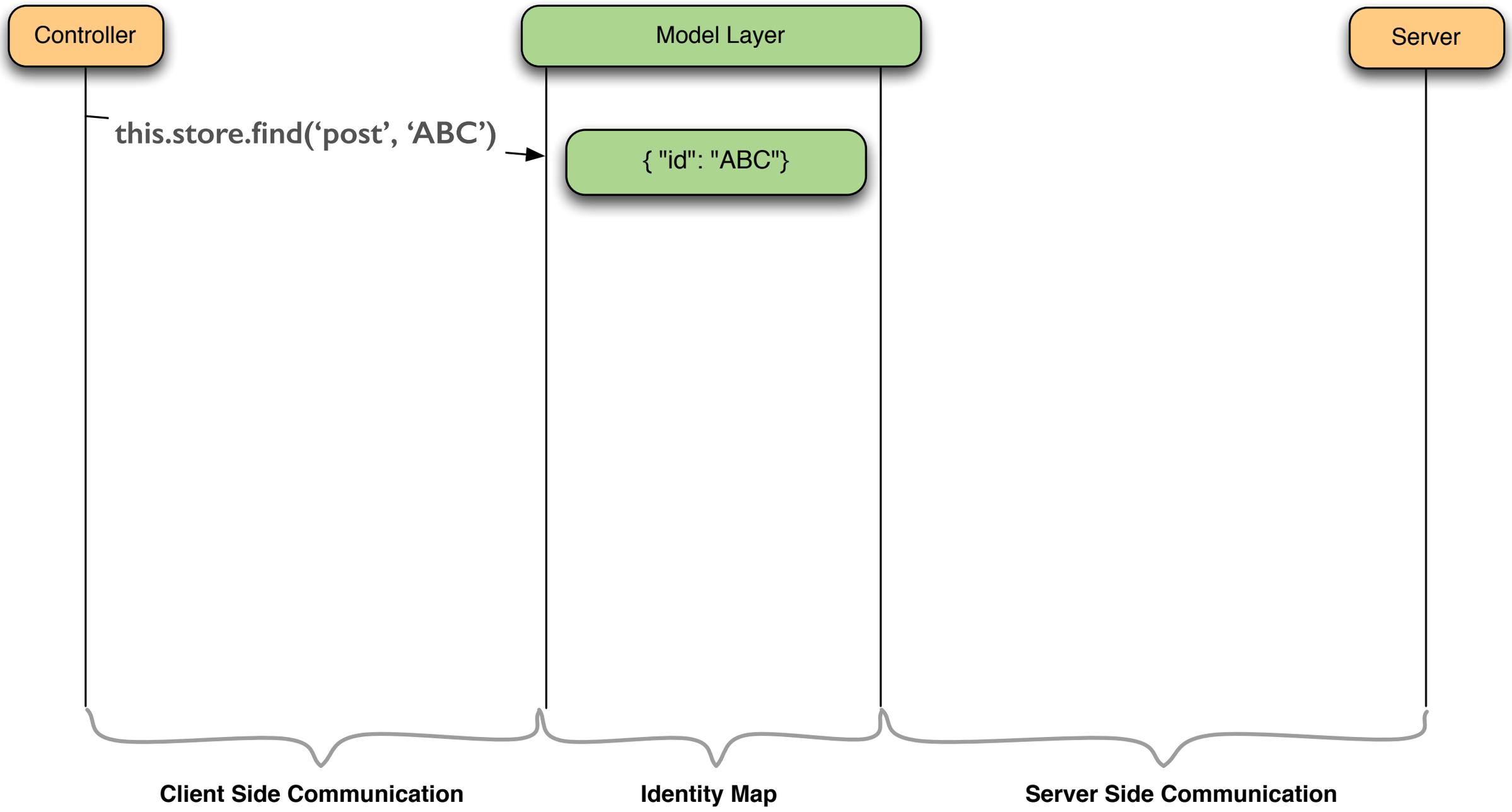
Identity Map



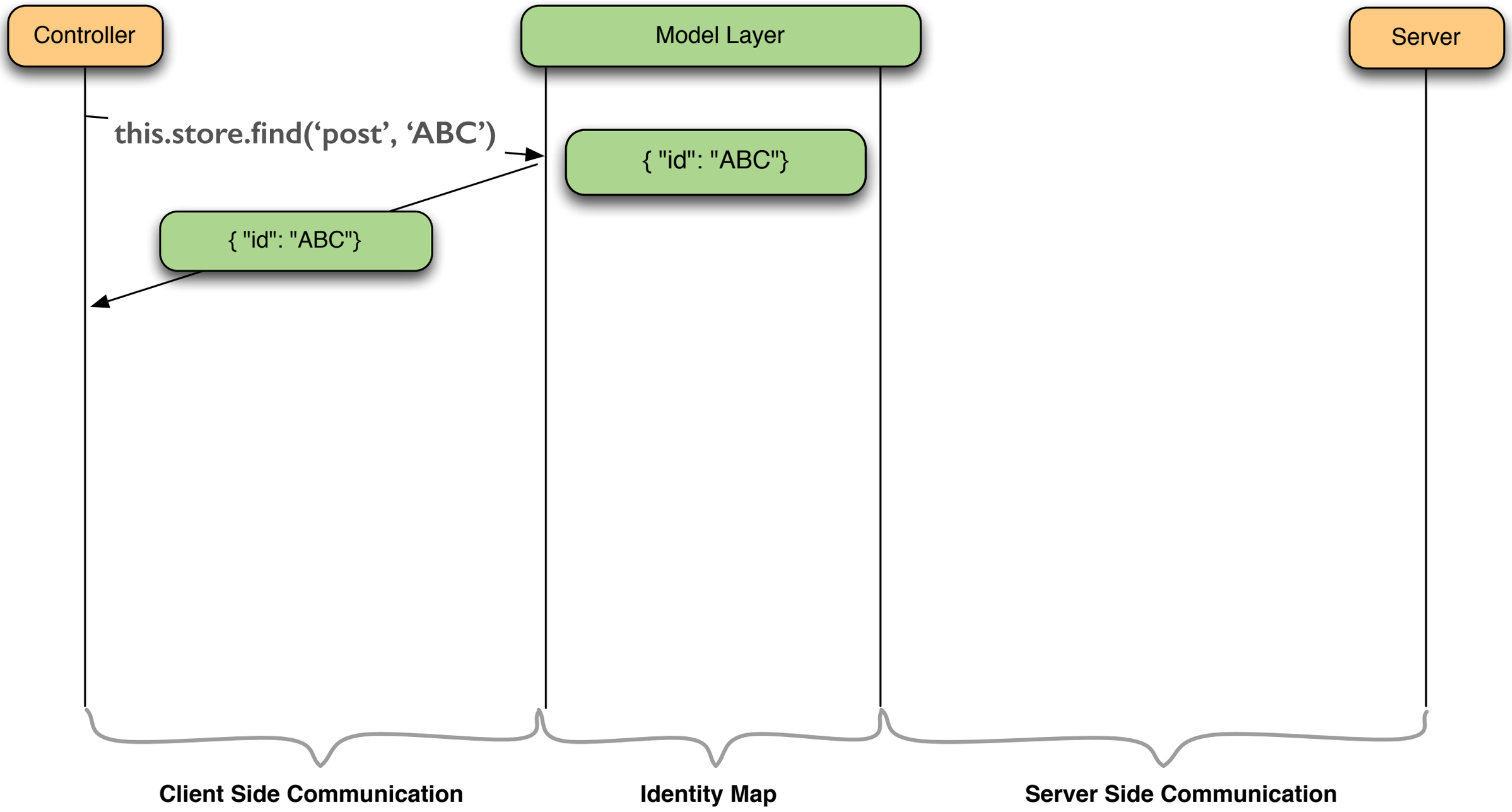
Identity Map



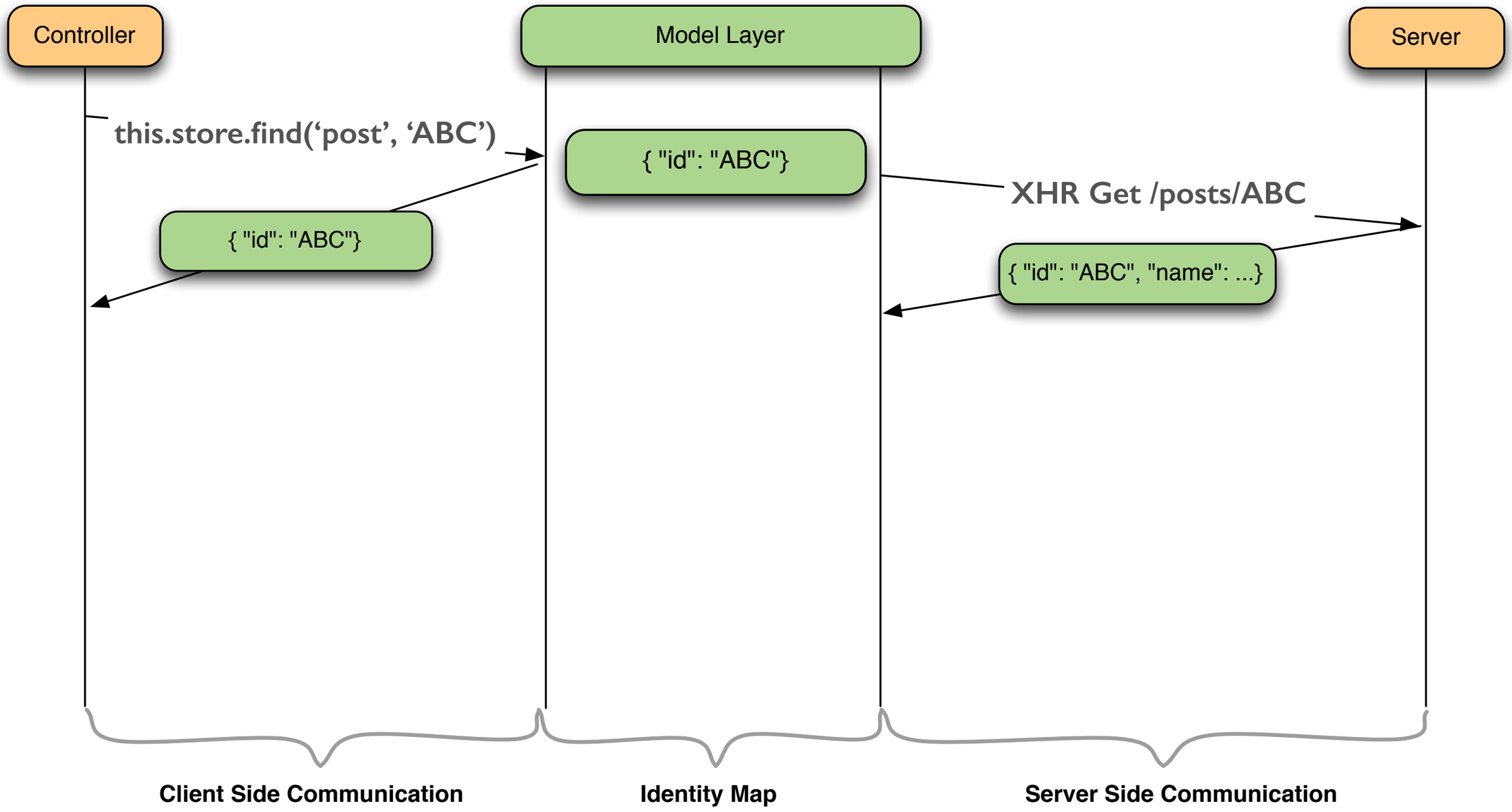
Identity Map



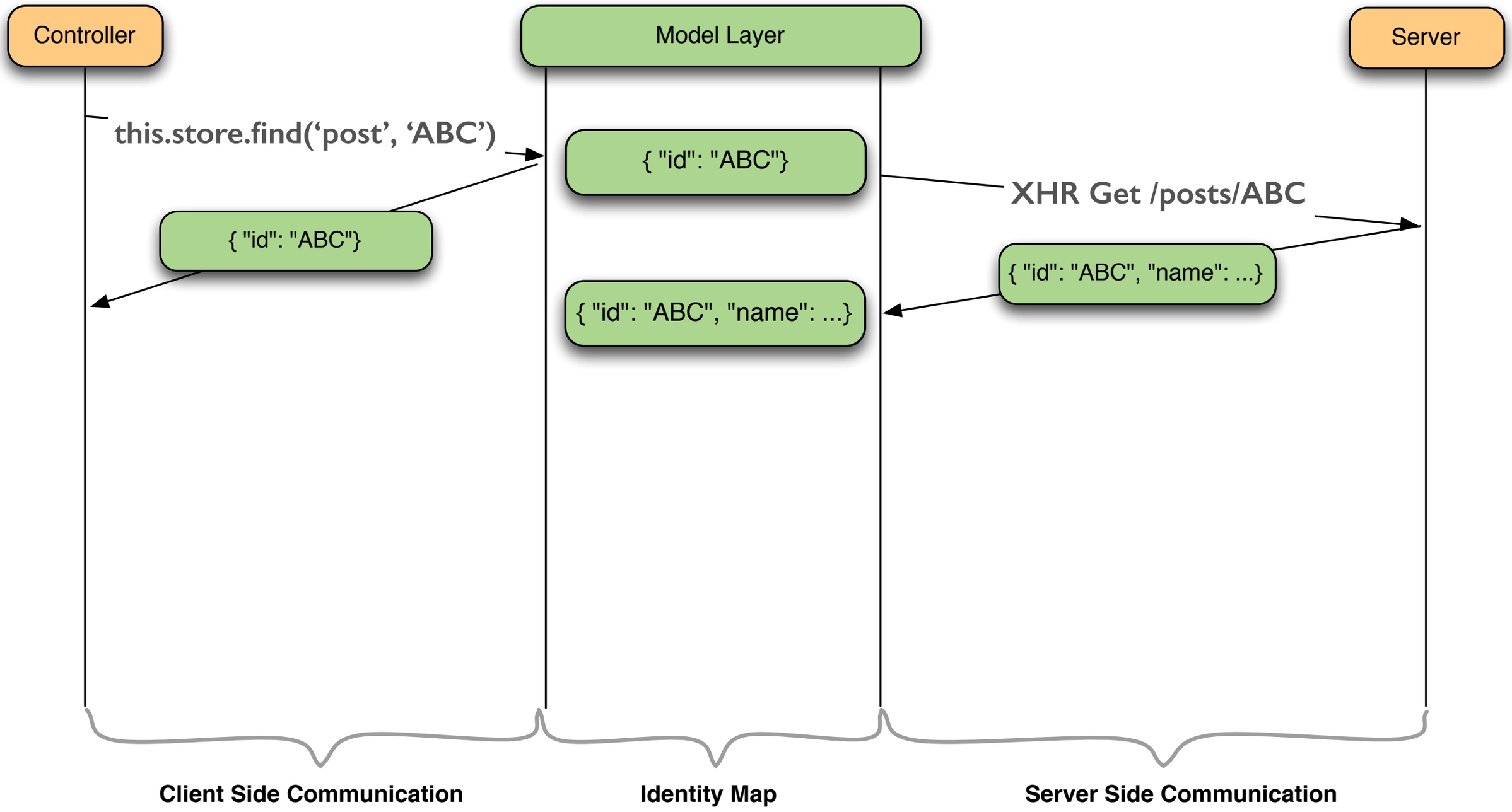
Identity Map



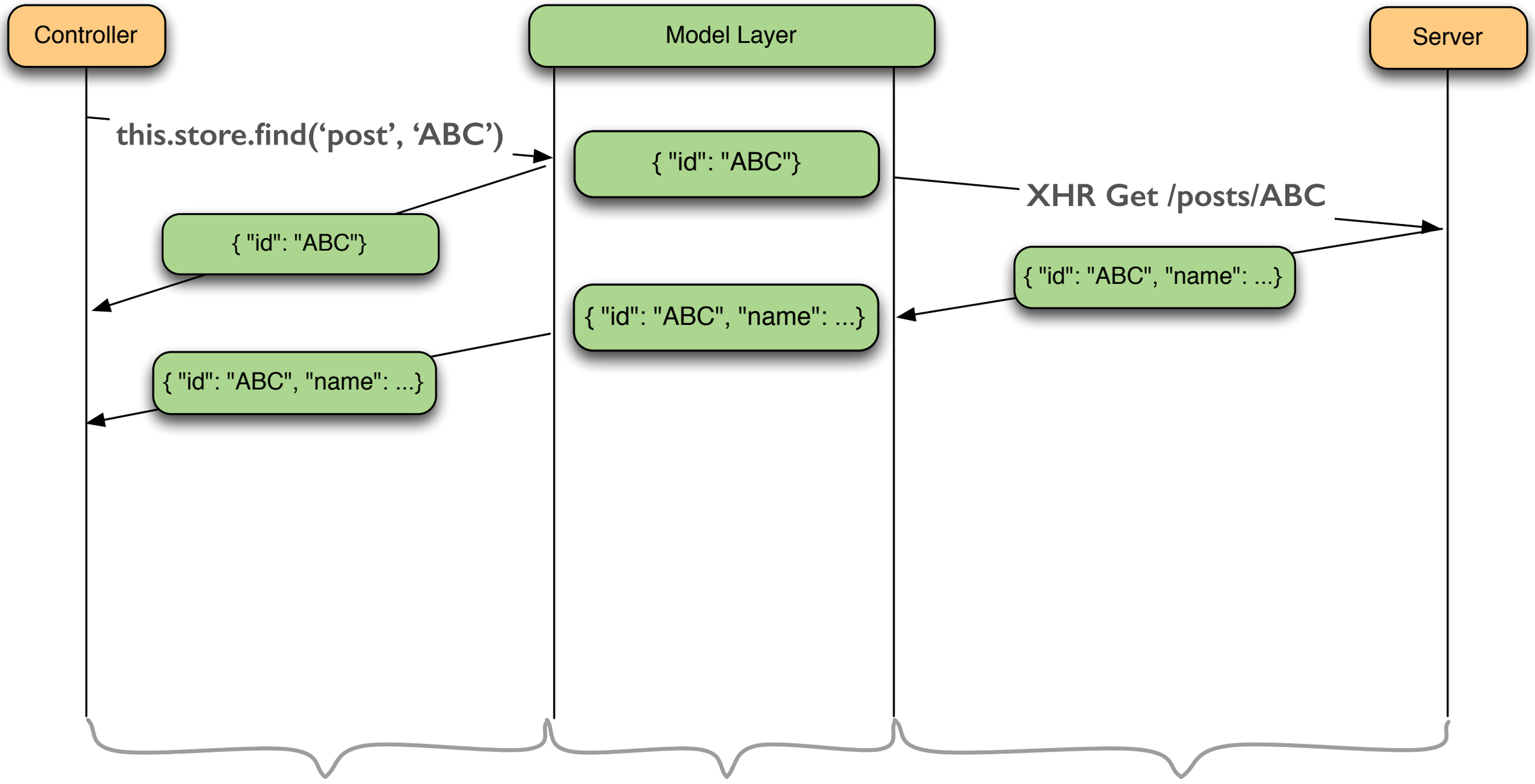
Identity Map



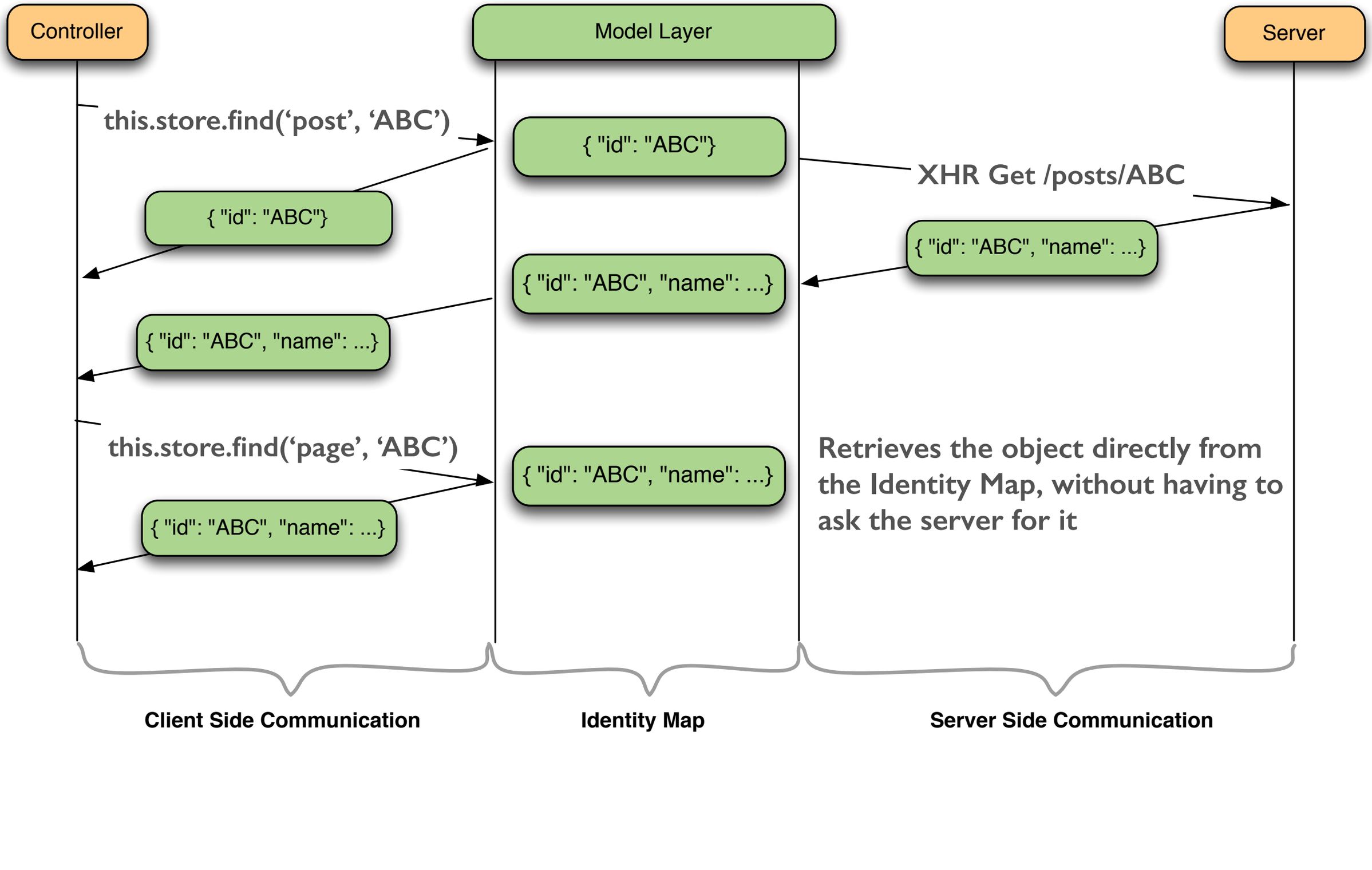
Identity Map



Identity Map



Identity Map



The REST Interface

curl "http://localhost:8080/json/data/albums"


```
{
    "albums": [
        {
            "id": "albumOne",
            "photos": [
                "photos_bird.jpg"
            ]
        },
        {
            "id": "albumTwo",
            "photos": [
                "photos_dragonfly.jpg",
                "photos_fly.jpg"
            ]
        }
    ]
}
```

The JSON Interface

curl "http://localhost:8080/json/albums/albumOne"


```
{
    "album": {
        "id": "albumOne",
        "photos": [
            "photos_bird.jpg"
        ]
    }
}
```

What is Ember Data

Ember Data is an Identity Map implementation

Handles relationships, transactions, and data serialization

Consists of

- A Store (identity map)
- An Adapter and a serializer (to serialize and deserialize data)
- An abstract Model implementation

Ember Data

Start by creating an Application Adapter

```
ember g adapter application
```

Choose either the JSONAPIAdapter, or the RESTAdapter. Add a namespace for the api if needed

```
import DS from 'ember-data';

export default DS.RESTAdapter.extend({
  namespace: '/api'
});
```

Ember Data

Create the Photo Model

ember generate model photo

Add in the properties, including property-types

```
export default DS.Model.extend({
  name: DS.attr('string'),
  squareUrl: DS.attr('string'),
  thumbUrl: DS.attr('string'),
  largeUrl: DS.attr('string'),
  xlUrl: DS.attr('string')
});
```

Ember Data

Built-in property types

string
number
date
one-to-one and one-to-many relationships

Relationships

One-to-One relationships

one to none

```
App.A {
   modelB: belongsTo('b')
}

App.B {
}
```

```
{
   "model_a": {
      "id": "a_id",
      "modelB": "b_id"
   }
}
```

One to one

```
App.A {
   modelB: belongsTo('b')
}

App.B {
   modelA: belongsTo('a')
}
```

```
{
   "model_a": {
      "id": "a",
      "modelB": "b_id"
   }
}

{
   "model_b": {
      "id": "b_id",
      "modelA": "a_id"
   }
}
```

Relationships

One-to-Many relationships

Many to none

```
App.A {
  modelB: hasMany('b')
}

App.B {
}
```

```
{
  "model_a": {
    "id": "a_id",
    "modelBs": [ "b_id", …] }
  }
}
```

Many to one

```
App.A {
  modelB: hasMany('b')
}

App.B {
  modelB: belongsTo('a')
}
```

```
{
  "model_a": {
    "id": "a",
    "modelBs": [ "b_id", …] }
  }
}

{
  "model_b": {
    "id": "b_id",
    "modelA": "a_id"
  }
}
```

Relationships

Many-to-Many relationships

Many to many

```
App.A {
   modelB: hasMany('b')
}

App.B {
   bodelA: hasMany('a')
}
```

```
{
   "model_a": {
      "id": "a",
      "modelBs": [ "b_id", …] }
   }
}

{
   "model_b": {
      "id": "b_id",
      "modelAs": [ "a_id", …] }
   }
}
```

Ember Data States

isNew	Specifies that this record is created locally but the adapter has not yet confirmed that the record is persisted to the backend
isLoaded	Specifies that the model has been successfully loaded from the server side; if the record is created locally, `isLoaded` is `true`
isDirty	Specifies that this model has been altered on the client side; Includes models that have been created locally but not yet persisted to the server side or deleted
isSaving	Specifies that the model has been sent to the server side, but the server side hasn't yet responded
isDeleted	Specifies that the model has been deleted on the client side but not yet sent to the server side
isError	Specifies that the model has either failed client side validation or that Ember Data was unable to persist the local changes to the server side

Ember Data Notifications

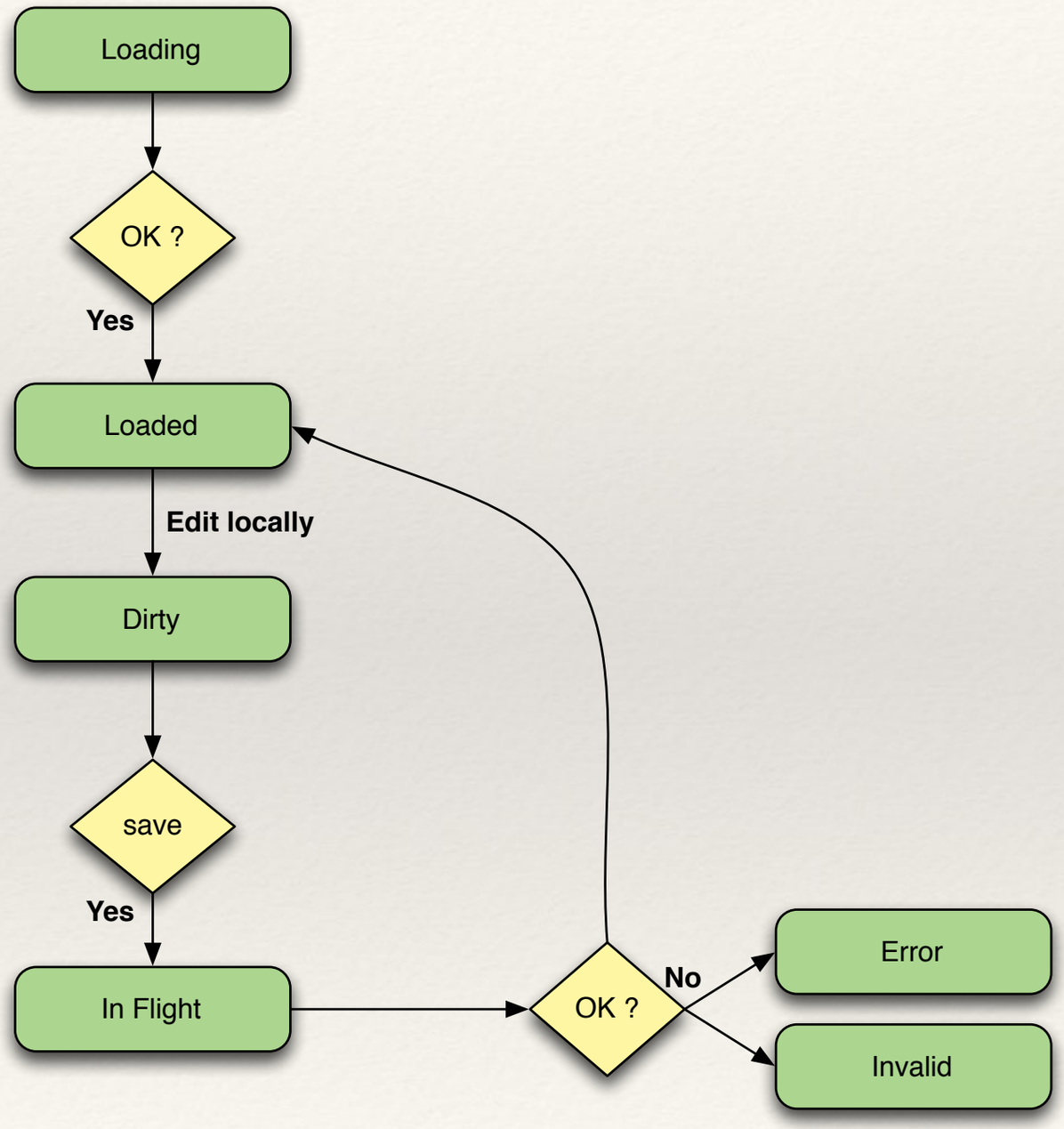
didLoad	Triggered when the model is loaded from the server
didCreate	Triggered when the model is created locally on the client side
didUpdate	Triggered when the model is updated locally on the client side
didDelete	Triggered when the model is deleted locally on the client side
becameError	Triggered when the backend reports an error while performing CRUD operations on the model
becameInvalid	Triggered when client side validations fail on the model

Ember Data Notifications

```
model.on('didLoad', function() {
  console.log("Loaded!");
});
```

Ember Data

Simplified Ember Data Statechart



Ember Data

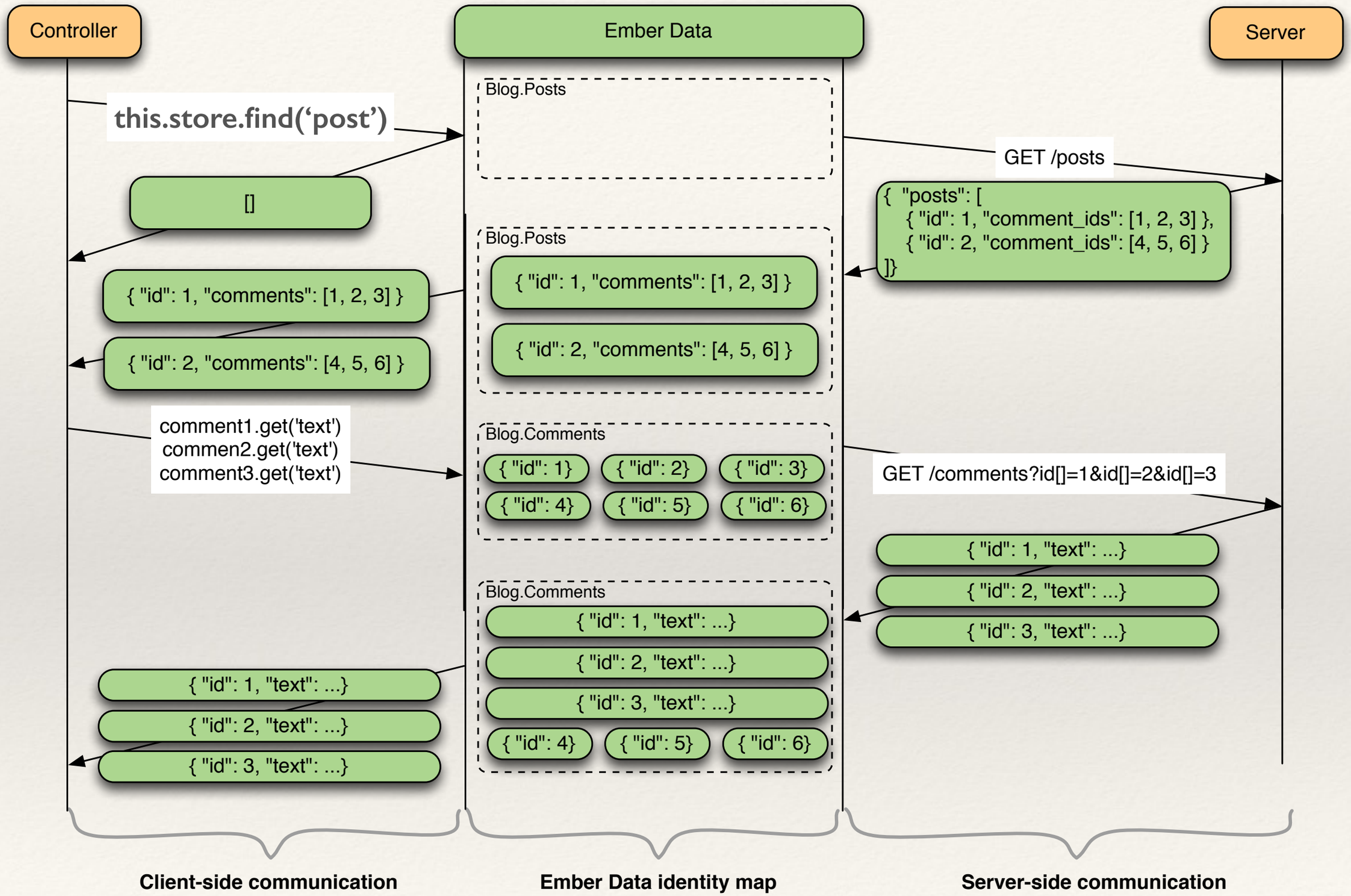
Album-model

```
export default DS.Model.extend({
    sortIndex: DS.attr('number'),
    photos: DS.hasMany('photo', {async: true}),
    cover: DS.belongsTo('photo', {async: true})
});
```

Photo-model

```
export default DS.Model.extend({
    href: DS.attr(‘string')
});
```

Ember Data - Sideloading



Ember Data

Sideloading data

```
{
  “posts: [
    {“id”: 1, “comment_ids”: [1, 2, 3]},
    {“id”: 2, “comment_ids”: [4, 5, 6]}
  ],
  “comments”; [
    {“id”: 1, “text”: “Comment 1”},
    {“id”: 2, “text”: “Comment 2”},
    {“id”: 3, “text”: “Comment 3”},
    {“id”: 4, “text”: “Comment 4”},
    {“id”: 5, “text”: “Comment 5”},
    {“id”: 6, “text”: “Comment 6”},
  ]
}
```

Ember Data

Embedding Data

```
{
  “posts: [
    {
        “id”: 1,
        “comments”: [
            {“id”: 1, “text”: “Comment 1”},
            {“id”: 2, “text”: “Comment 2”},
            {“id”: 3, “text”: “Comment 3”}
        ]
    },
    {
        “id”: 2,
        “comments”: [
            {“id”: 4, “text”: “Comment 4”},
            {“id”: 5, “text”: “Comment 5”},
            {“id”: 6, “text”: “Comment 6”}
        ]
    }
  ]
}
```

Ember Data

Specify special plurals

```
Ember.Inflector.inflector.irregular(
    'productCategory', 'productCategories');
```

Demo Time

Building out the functionality