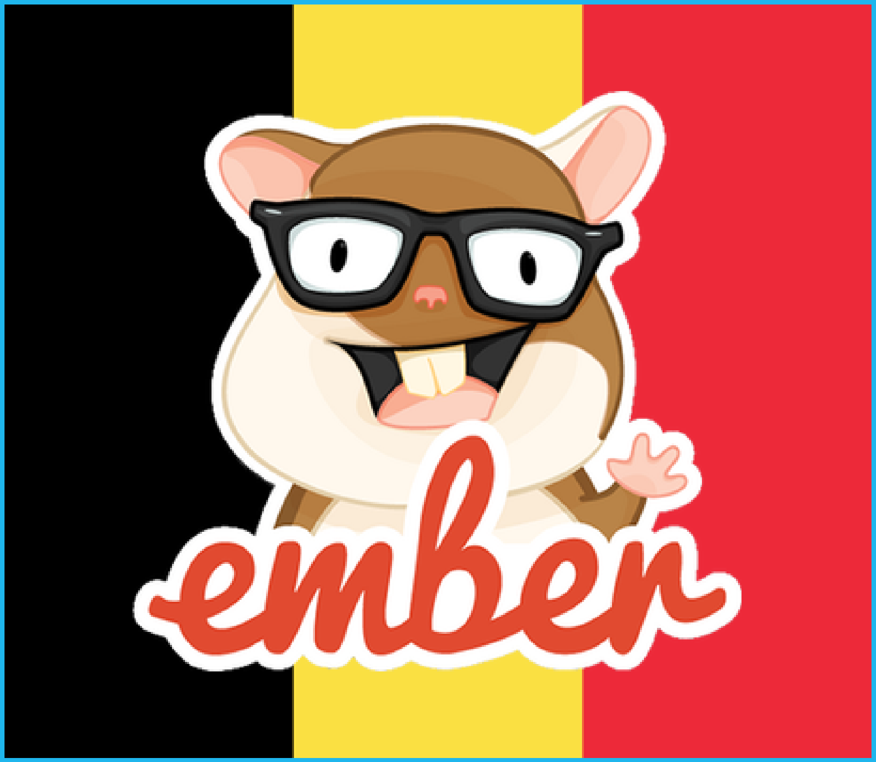
Welcome, Embeerists

to the **second** Ember.js meetup in **Brussels**

4th of December 2013



Agenda

- A Live Polling System in Ember.js (Yoran)
- Loading Animations for Asynchronous Data (Yoran)
- ?

Live polling system

For periodic updates in a single-page app

No real-time updates (need persistent connection like WebSockets)

Used in Hstry for live updates of comments and likes

How does Ember help?

Add models to the store and view updates itself

Example

We have `Post`'s and `Comment`'s

We want to periodically update the new comments to a post

Show me the code

Pollster object

Responsible for setting an interval that periodically calls an `onPoll` function

```
App.Pollster = Ember.Object.extend({
  start: function() {
    // POLL_INTERVAL is for instance 30 seconds
    this.timer = setInterval(this.onPoll.bind(this), POLL_INTERVAL);
  },
  stop: function() {
    clearInterval(this.timer);
  },
  onPoll: function() {
    // Issue JSON request and add data to the store
  }
});
```

Setup this pollster in the route

```
App.PostRoute = Ember.Route.extend({
  setupController: function(controller, model) {
    this.set('pollster', App.Polster.create({
      onPoll: function() {
        // Defined on the next slide
      }
    }));
    this.get('pollster').start();
  },
  // This is called upon exiting the Route
  deactivate: function() {
    this.get('pollster').stop();
  }
});
```

The `onPoll` function

```
// This code is executed in the 'setupController' hook of the Route
var route = this;

onPoll: function() {
  // AJAX request
  Ember.$.getJSON('/updates', 'GET').then(function(json_obj) {
    // The JSON structure is as follows:
    // {
    //   comments: [
    //     { ... },
    //     { ... },
    //   ]
    // }

    // Iterate through the comments
    Ember.$.each(json_obj.comments, function(index, comment) {
      // Make sure that the comment is not already in the store
      if (! route.get('store').recordIsLoaded(App.Comment, comment.id)) {
        route.get('store').push('comment', comment);
      }
    });
  });
}
```

Now we have added the data to the store

But the view is not necessarily updated

Need to use the `filter` API function

Filters, on the other hand, perform a live search of all of the records in the store’s cache.

Takes a type and filter function, and returns a live RecordArray that remains up to date as new records are loaded into the store or created locally.

Our template

```
<article class="post">
  <h2>{{title}}</h2>
  <p>{{content}}</p>
  <aside>{{render "comments" comments}}</aside>
</article>
```

Our controller for the template

```
App.PostController = Ember.ObjectController.extend({
  comments: function() {
    var postId = this.get('id');
    // Get all the comments that belong to this post
    return this.get('store').filter('comment', function(comment) {
      return comment.get('post.id') == postId;
    });
  }.property()
});
```

Questions?

Loading animations

People don't like waiting

But they mind less when they know something is happening

First solution in Ember

Use `beforeModel` and `afterModel` hooks

```
App.PostRoute = Ember.Route.extend({
  beforeModel: function() {
    // Assume the 'loading' class displays an overlay with a loading animation
    Ember.$('body').addClass('loading');
  },
  model: function(params) {
    return this.store.find('post', params.post_id);
  },
  afterModel: function() {
    Ember.$('body').removeClass('loading');
  }
});
```

Limitations

Only works if the model doesn't have any asynchronous associations

```
App.Post = DS.Model.extend({
  title: DS.attr('string'),
  comments: DS.hasMany('comment', { async: true })
});
```

Example JSON response for a Post

```
{
  post: {
    id: 1,
    title: "This is a post",
    links: {
      comments: "/post/1/comments"
    }
  }
}
```

```
<article class="post">
  <h2>{{title}}</h2>
  <p>{{content}}</p>
  <aside>{{render "comments" comments}}</aside>
</article>
```

The comments will only be loaded after the template has started rendering.

This is after the `afterModel` hook

Rest of the page will have rendered

Instead of comments the user will see "blank"

We want to wait until everything has loaded to remove the loading animation

Second solution

Use Promises

What does this return

```
post.get('comments');
```

`PromiseArray` if comments not yet loaded

`ManyArray` if comments already in the store

We want to have Promises everywhere

```
Ember.RSVP.makePromise = function(maybePromise) {
  // Test if it's a promise
  if (maybePromise.then) {
    // Then return it
    return maybePromise;
  } else {
    // Wrap it in a Promise that resolves directly
    return Ember.RSVP.resolve(maybePromise);
  }
};
```

The code

```
App.PostRoute = Ember.Route.extend({
  beforeModel: function() {
    // Assume the 'loading' class displays an overlay with a loading animation
    Ember.$('body').addClass('loading');
  },
  model: function(params) {
    return this.store.find('post', params.post_id);
  },
  setupController: function(post, controller) {
    // Pre-load the comments
    // The 'get' call will result in an AJAX call to get the comments and returns a promise
    var comments = Ember.RSVP.makePromise(post.get('comments'));

    // Wait until the promise has been resolved
    comments.then(function() {
      // Wait until all templates have finished rendering
      Ember.run.scheduleOnce('afterRender', this, function() {
        // Remove the loading animation
        Ember.$('body').removeClass('loading');
      });
    });
  }
});
```

Thank you

Oh, and Hstry is still hiring :-)