

Managing State in Single Page

with Ember.js

16 Feb



Ember.js

- This is only to pique your interest
- Browser based MVC framework
- Builds on JQuery
- Beyond event driven abstractions
- Auto updating of DOM

Users

3

Bob
Mark (me)

1

Tasks

Mow the lawn

Cut the hedge

Plant vegetables

2

Show tasks for a user

Users

3

Bob

Mark (me)

1

Tasks

Mow the lawn

Cut the hedge

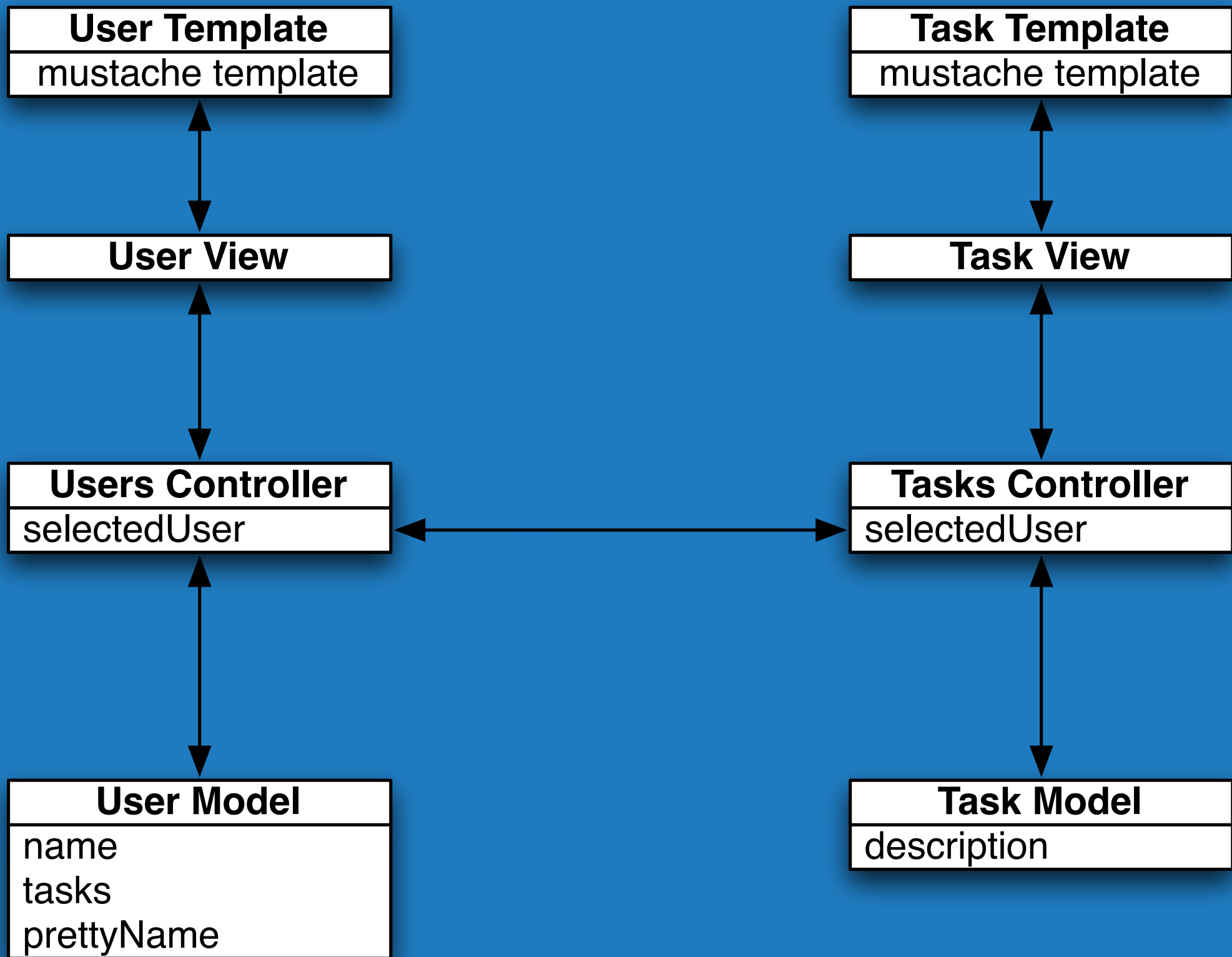
Plant vegetables

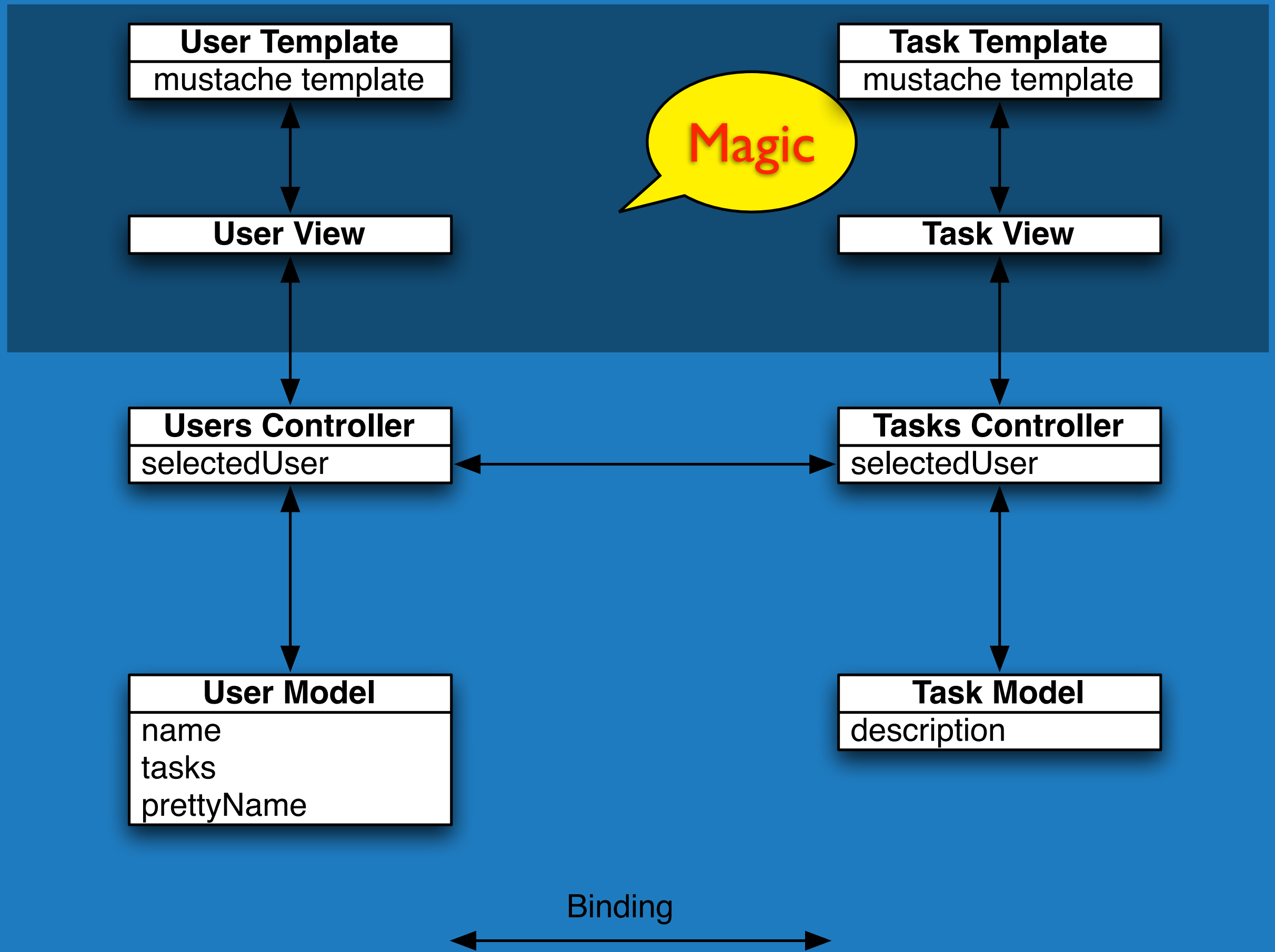
2

Show tasks for a user

Models

Controllers





Ember Models

```
User = Ember.Object.extend({  
  name: null,  
  tasks: []  
});
```

```
manager.name = 'John';
```

Ember Models

```
User = Ember.Object.extend({  
  name: null,  
  tasks: []  
});
```

```
/* create two users */  
bob = User.create({ name: "Bob" });  
mark = User.create({ name: "Mark" });
```

```
bob.get("name"); //=> "Bob"  
mark.get("name"); //=> "Mark"
```


Users Controller

// A standard Ember Object

```
MyApp.usersController = Ember.Object.create({  
  users: [],  
  selectedUser: null,  
});
```

```
MyApp.usersController.set("users", [bob, mark]);
```

Users Controller

// A standard Ember Object

```
MyApp.usersController = Ember.Object.create({  
  users: [],  
  selectedUser: null,  
});
```

```
MyApp.usersController.set("users", [bob, mark]);
```

// fake a <click> event

```
MyApp.usersController.set("selectedUser", bob);
```

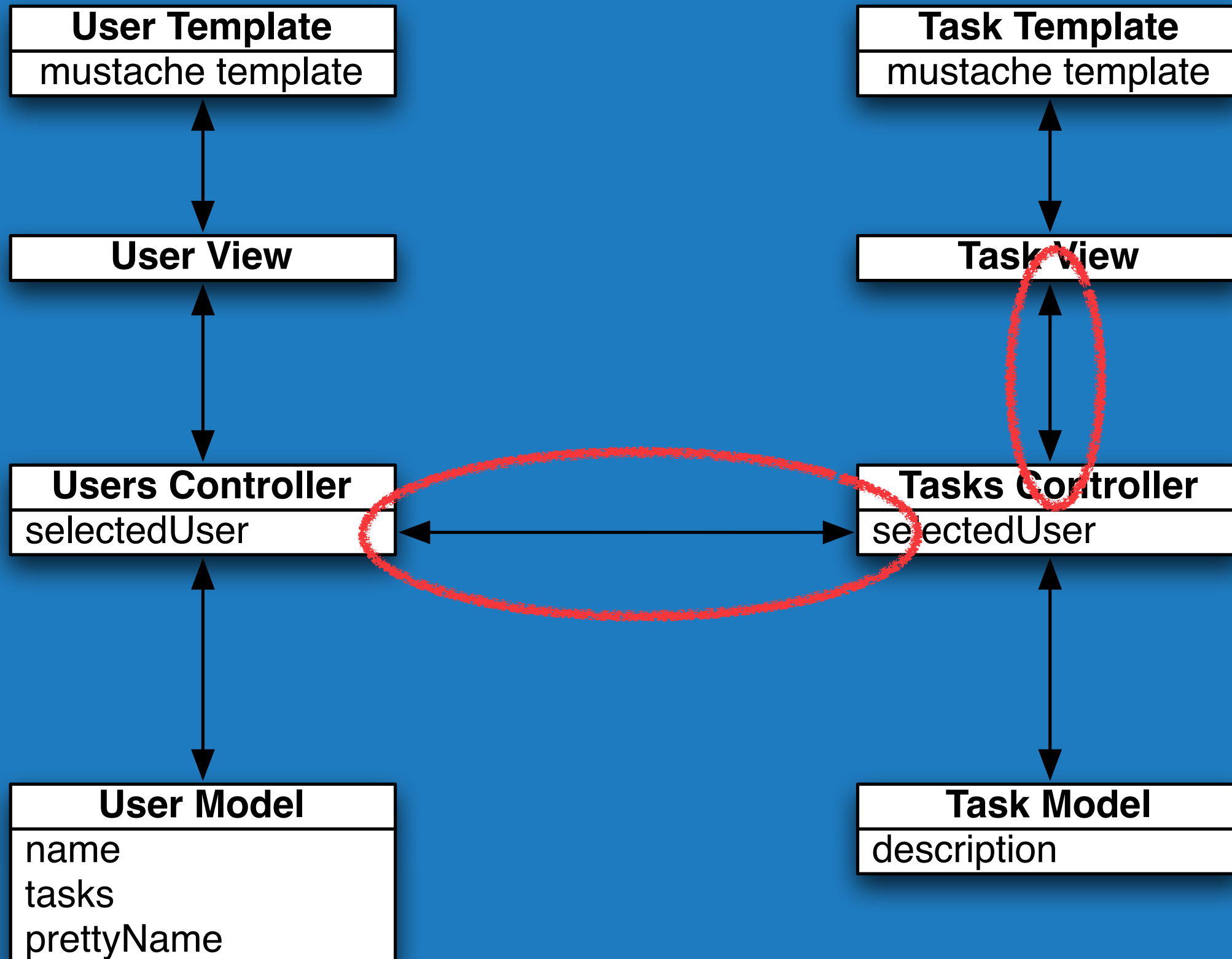


Magic

```
MyApp.usersController.getPath("selectedUser.name");  
//=> "Bob"
```

Encapsulation

- Users View should only care about the Users Controller
- Tasks View should only care about the Tasks Controller



Tasks Controllers

```
// A standard Ember Object  
MyApp.tasksController = Ember.Object.create({  
  selectedUserBinding: 'MyApp.usersController.selectedUser'  
});  
  
MyApp.tasksController.getPath("selectedUser.name");  
//=> "Mark" (or "Bob")
```

Tasks Controllers

```
// A standard Ember.Object.create({
  MyApp.tasksController: {
    selectedUserBinding: 'MyApp.usersController.selectedUser'
  }
});
```

Magic

```
MyApp.tasksController.getPath("selectedUser.name");
//=> "Mark" (or "Bob")
```

Observers

```
MyApp.tasksController = Ember.Object.create({  
  selectedUserBinding: 'MyApp.usersController.selectedUser',  
  filterResults: function() {  
    document.write("filter the results");  
    // triggers a re-render  
  }.observes('selectedUser')  
});
```

Observers

```
MyApp.tasksController = Ember.Object.create({  
  selectedUserBinding: 'MyApp.usersController.selectedUser',  
  filterResults: function() {  
    document.write("filter the results");  
    // triggers a re-render  
    .observes('selectedUser')  
  });
```


Observers

```
MyApp.tasksController = Ember.Object.create({  
  selectedUserBinding: 'MyApp.usersController.selectedUser',  
  filterResults: function() {  
    document.write("filter the results");  
    // triggers a re-render  
    .observes('selectedUser')  
  });
```

```
// somehow the selected user is changed!  
MyApp.usersController.set("selectedUser", mark);
```

```
MyApp.tasksController.getPath("selectedUser.name");  
//=> "Mark" (or "Bob")  
// side effect of "filter the results and re-render"
```

Users

3

Bob

Mark (me)

1

Tasks

Mow the lawn

Cut the hedge

Plant vegetables

2

Show tasks for a user

Users

3

Bob
Mark (me)

1

Tasks

Mow the lawn

Cut the hedge

Plant vegetables

2

Show tasks for a user

Computed Props

```
User = Ember.Object.extend({
  name: null,
  currentUserBinding: 'MyApp.currentUser',

  // computed property
  prettyName: function() {
    if(this == this.get("currentUser"))
      return this.get("name") + " (me)";

    return this.get("name");
  }.property("name")
});

bob.get("prettyName"); // => "Bob"
mark.get("prettyName"); // => "Mark (me)"
```

Computed Props

```
User = Ember.Object.extend({  
  name: null,  
  currentUserBinding: 'MyApp.currentUser',  
  
  // computed property  
  prettyName: function() {  
    if(this == this.get("currentUser"))  
      return this.get("name") + " (me)";  
  
    return this.get("name");  
  }.property("name")  
});
```

```
bob.get("prettyName"); // => "Bob"  
mark.get("prettyName"); // => "Mark (me)"
```

Computed Props

```
User = Ember.Object.extend({  
  name: null,  
  currentUserBinding: 'MyApp.currentUser',  
  
  // computed property  
  prettyName: function() {  
    if(this == this.get("currentUser"))  
      return this.get("name") + " (me)";  
  
    return this.get("name");  
  }.property("name")  
});
```

```
bob.get("prettyName"); // => "Bob"  
mark.get("prettyName"); // => "Mark (me)"
```

Computed Props

```
User = Ember.Object.extend({  
  name: null,  
  currentUserBinding: 'MyApp.currentUser',  
  
  // computed property  
  prettyName: function() {  
    if(this == this.get("currentUser"))  
      return this.get("name") + " (me)";  
  
    return this.get("name");  
  }.property("name")  
});
```

```
bob.get("prettyName"); // => "Bob"  
mark.get("prettyName"); // => "Mark (me)"
```

MOAR Magic

The Run Loop

- Pops a function on the queue
 - (sync, actions, destroy, timers)
- Triggered each Browser Event Loop
- JavaScript FAST – DOM slow

The Run Loop

**Magic just
got real!**

- Pong

- (sync,

- Triggered

- JavaScript

ent Loop

What we just saw

- Data propagated through multiple objects
 - computed properties
 - observers
 - bindings
- Data updated in batch

What do I know?

- Very steep learning curve
 - (~4 weeks to get past novice)
- Modern Browsers Only
- Very, very powerful

Agile Bench

Planning tool for agile teams

The screenshot displays the Agile Bench planning tool interface, which is divided into two main sections: a dashboard on the left and a Kanban board on the right.

Dashboard (Left):

- Summary Metrics:** Points: 137, Bus Value: 0, Stories: 35, Days: 106, Velocity: 13.
- 1 stories selected:** A section showing the current selection.
- All Users:** A list of user avatars.
- All Features:** A list of features with color-coded tags: API (orange), Blog (teal), Commercial (purple), Customer Support (red), Event (dark teal), Mobile (yellow), and Photos (light green).
- Iteration 5 (Current Iteration):** A table of tasks with columns for ID, Title, Status, and Assignee. The table shows tasks for Iteration 5 and Iteration 6.
- Iteration 6:** A section showing the next iteration's tasks.

Kanban Board (Right):

- Todo (6pts):** A column for tasks to be done. It contains two tasks: "48. Blocked: Fix broken feedback form" (Social, R1, M) and "21. As a user I want to publish contact information" (Blog, R1).
- In Progress (18pts):** A column for tasks currently being worked on. It contains three tasks: "22. As a site owner I want a pricing page" (Commercial, R1, S), "49. As a site owner I want a terms and conditions page" (Customer Support, R1, M), and "20. As a site owner I want to create a blog" (Blog, R1, M).