BackboneJS

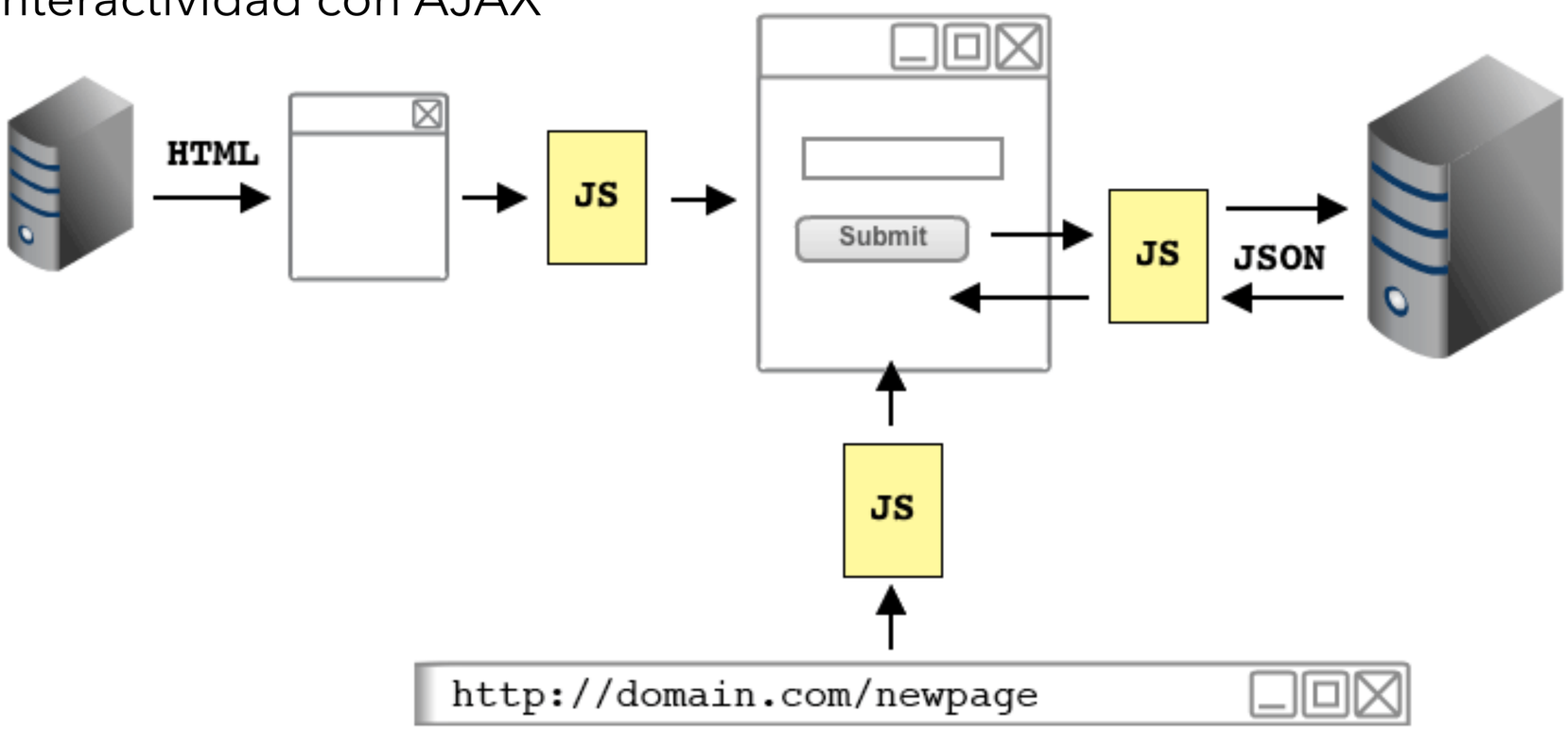
Sesión 1 - ¡Hola Backbone!

Índice

- **Single Page Applications**
- MVC en el cliente
- Backbone

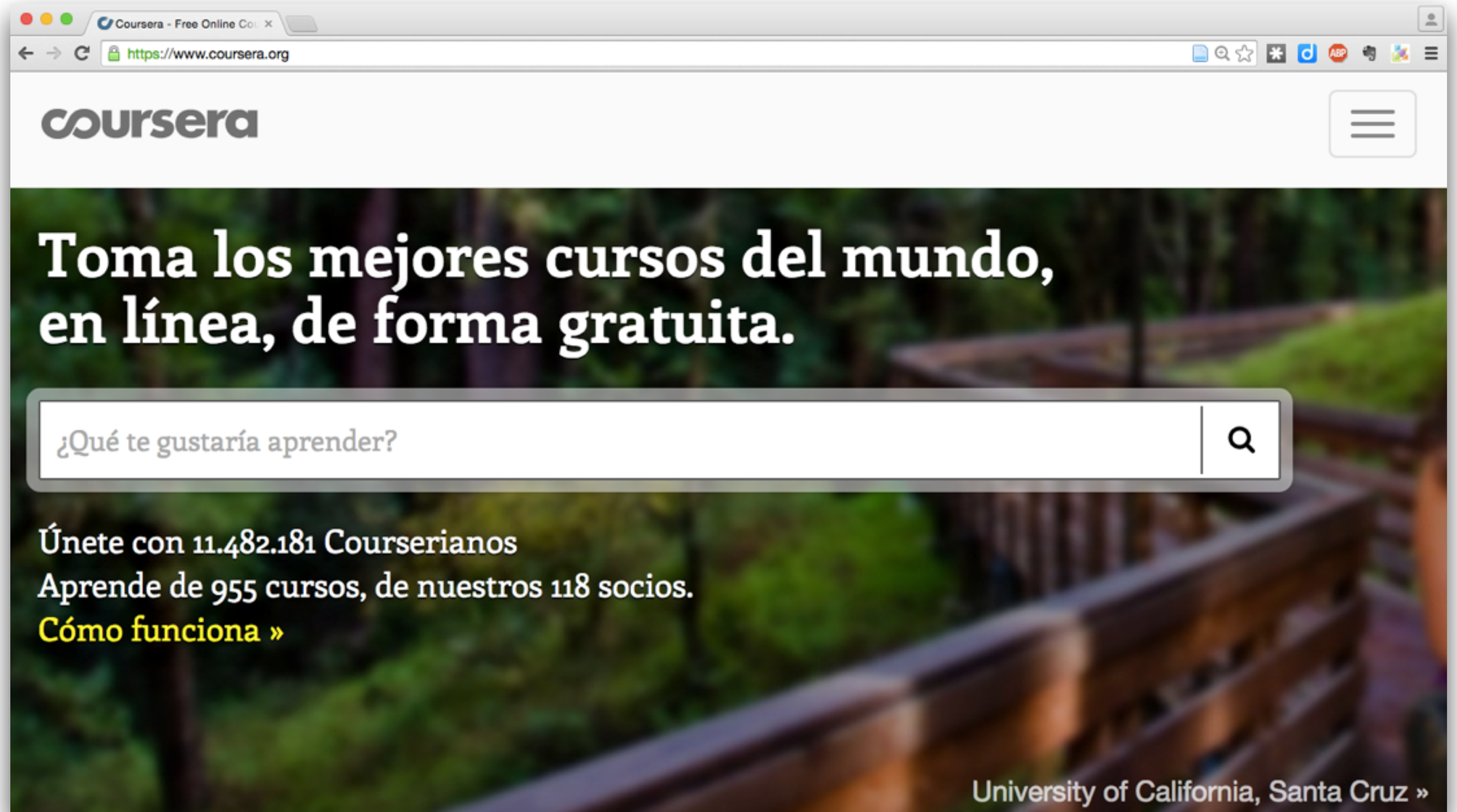
Single Page Application *(¡extrema!)*

- Solo se carga un HTML mínimo que contiene sobre todo JS
- El JS genera dinámicamente la interfaz
- Se añade interactividad con AJAX

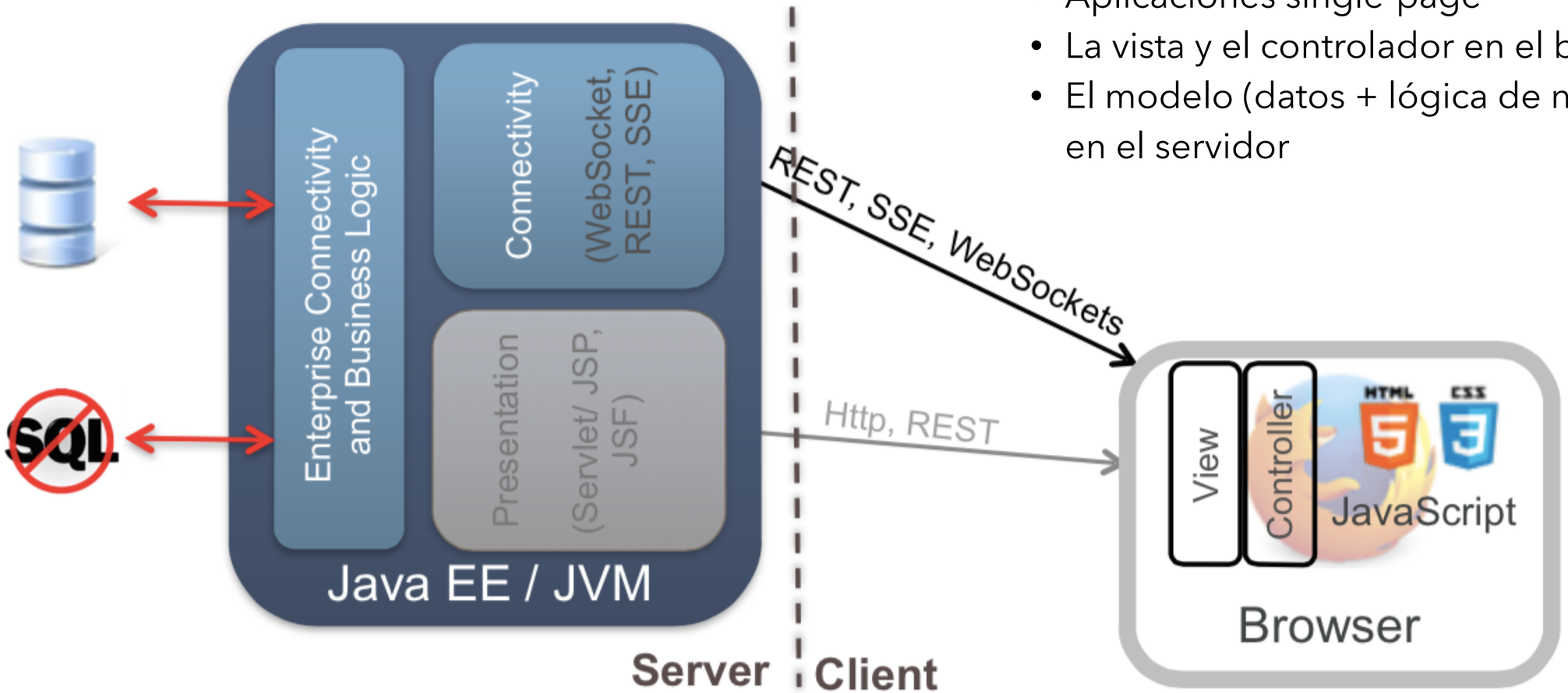


http://frontend-architectures.appspot.com/#/24

Ejemplo: Coursera



¿Cómo es una aplicación web con Java EE y JavaScript?



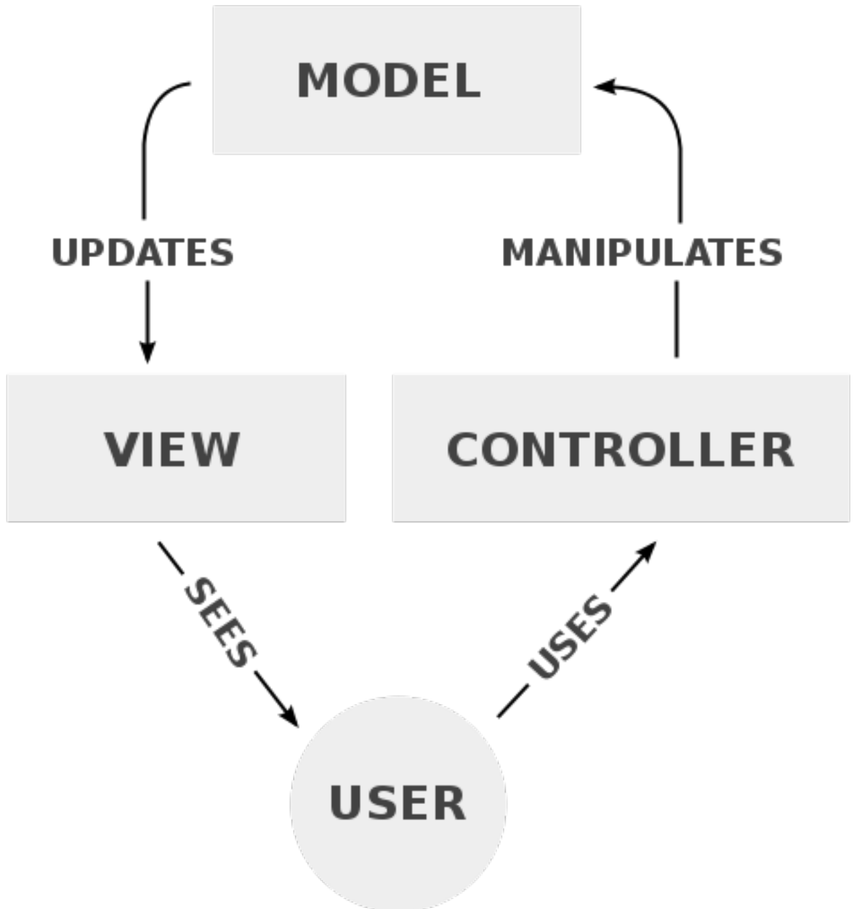
- Aplicaciones single-page
- La vista y el controlador en el browser
- El modelo (datos + lógica de negocio) en el servidor

Índice

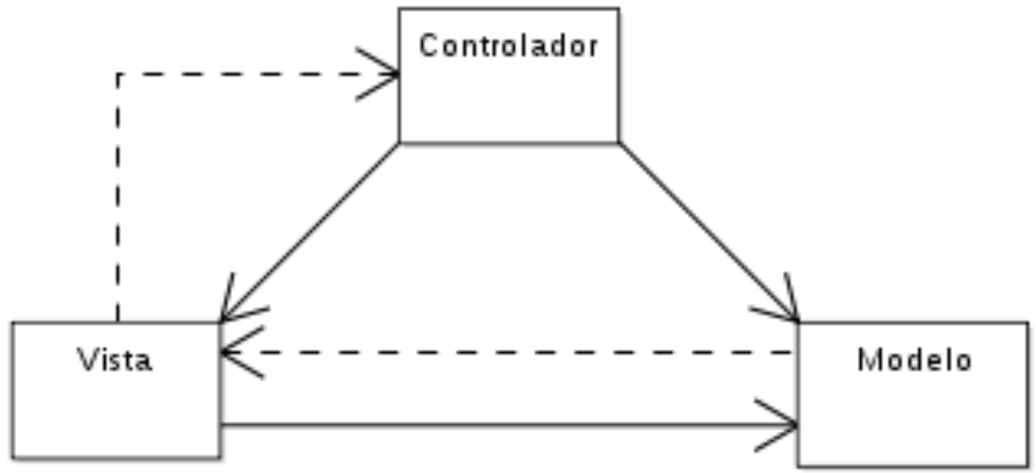
- Single Page Applications
- **MVC en el cliente**
- Backbone

MVC

- Patrón de diseño arquitectónico que **separa el modelo** del dominio **de su representación** en la interfaz

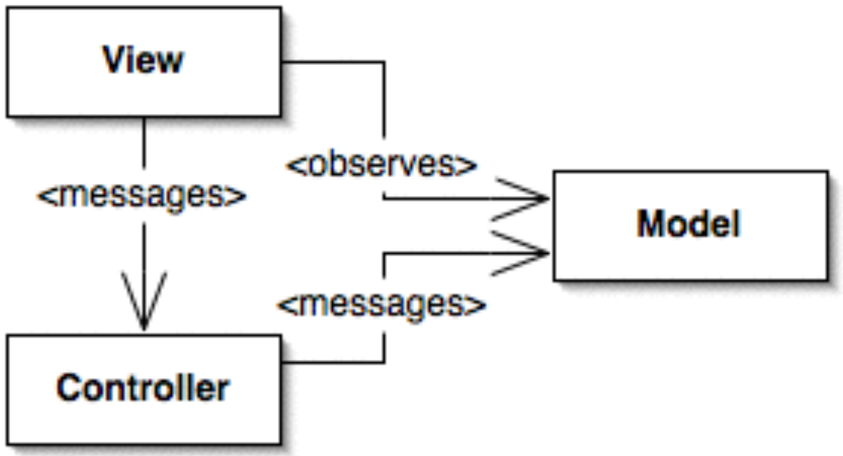


http://en.wikipedia.org/wiki/Model-view-controller

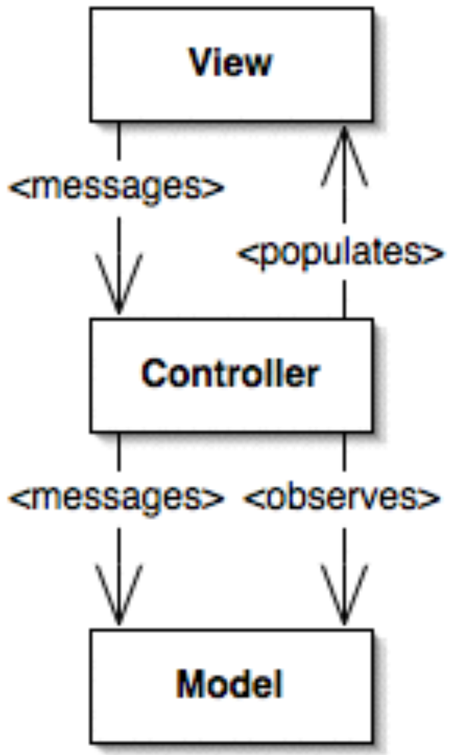


http://es.wikipedia.org/wiki/Modelo–vista–controlador

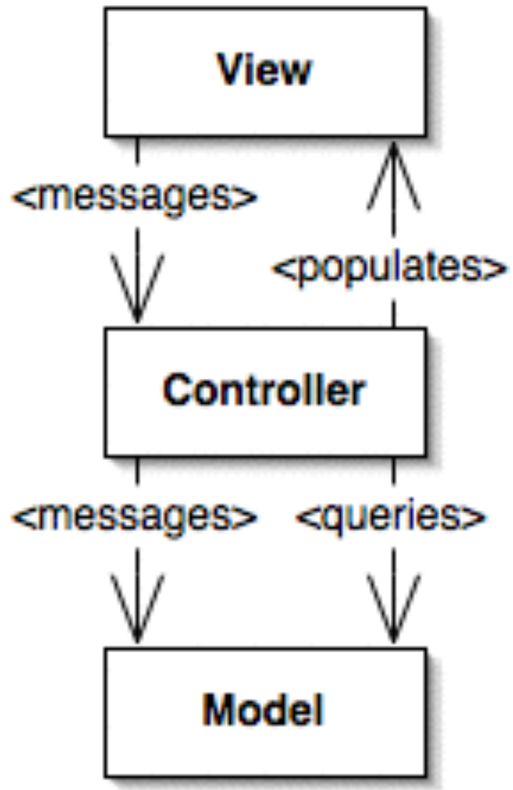
Algunas variantes



Smalltalk



NeXT (ahora OSX, iOS)



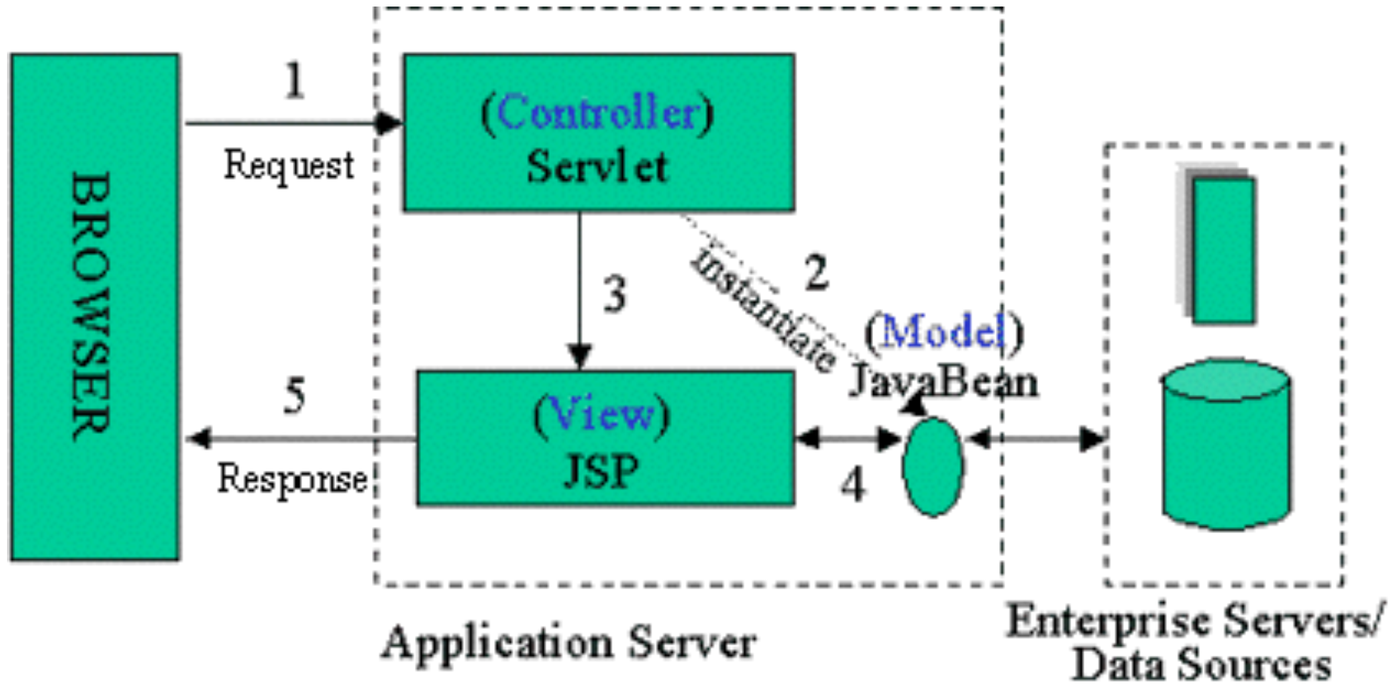
Model2 (web)

http://kasparov.skife.org/blog/src/java/mvc.html

Discusión de algunas variantes históricas de MVC

MVC en apps web (servidor)

- Adaptado a las peculiaridades de la web
  - Interacciones por HTTP
  - Sin estado
- Presente en todas las plataformas
  - JavaEE: Spring, Struts, JSF,…
  - .NET: ASP.NET MVC,…
  - PHP: CakePHP, Symphony, CodeIgniter,…
  - Ruby: Rails,…
  - Python: Django,…



Frameworks MVC en el cliente

- Nos pueden ofrecer
  - **Modelos, vistas y controladores** (¡lógicamente!)
  - **Data binding:** vinculación automática entre los datos del modelo y lo mostrado en la vista
  - **Sincronización con el servidor:** guardado/recuperación automática de los modelos si en el servidor hay un API REST
  - **Templating:** posibilidad de generar HTML con partes variables sin necesidad de andar concatenando cadenas
  - Control del **historial de navegación,** para que la URL pueda cambiar sin cambiar de página (Single Page App)

- **Pero sobre todo...**

Evitar el código spaguetti

```
xhr.open('POST', 'api/peticiones/' + idPeticion + "/firmas" , true)
xhr.onreadystatechange = function() {
   if (this.readyState==4)
      if (this.status==200)
          document.getElementById("resultado").innerHTML = "petición firmada"
}
xhr.setRequestHeader("Content-type", "application/json")
var firma = {};
firma.email = document.getElementById("email").value
firma.comentario = document.getElementById("comentario").value
...
firma.publica = document.getElementById("publica").checked;
xhr.send(JSON.stringify(firma))
```

Hay mucho donde elegir

Spine

ANGULARJS by Google

Maria

Knockout.

batman.js

cujoJS

canjs

AGILITY.JS

ember

Últimamente…

Spine

cujoJS

Maria

Knockout.

batman.js

ANGULARJS by Google

canjs

ember

AGILITY.JS

TodoMVC (http://todomvc.com)

- App de ejemplo: “lista de tareas pendientes”, implementada con muchos *frameworks* MVC distintos para que un desarrollador tenga elementos de juicio prácticos para compararlos o evaluarlos

Índice

- Single Page Applications
- MVC en el cliente
- **Backbone**

- http://backbonejs.org
- Uno de los pioneros de MVC en Javascript
- **Sencillo**: 1608 líneas de código (v 1.1.2), de las cuales la mitad son comentarios
- Amplia comunidad de desarrolladores y de plugins y extensiones

Hay más de una “forma correcta” de hacer las cosas

Y Hacker News new | comments | ask | jobs | submit login

▲ koko775 634 days ago | link | parent

I'm trying to learn Backbone, I really am. But there's so many tutorials suggesting everything that I don't have a clue how I should structure a project and what to think about as a codebase grows.

Granted, I'm fairly new to Javascript, but - no offense meant - I can't get excited about node.js. Backbone is the first thing about Javascript that really makes me want to write it. And I know enough to know what I lack is primarily practical experience, but every step I take with Backbone feels like a misstep, because it seems like there's no one right answer by design, so I'm paralyzed.

What to do?

https://news.ycombinator.com/item?id=3532542

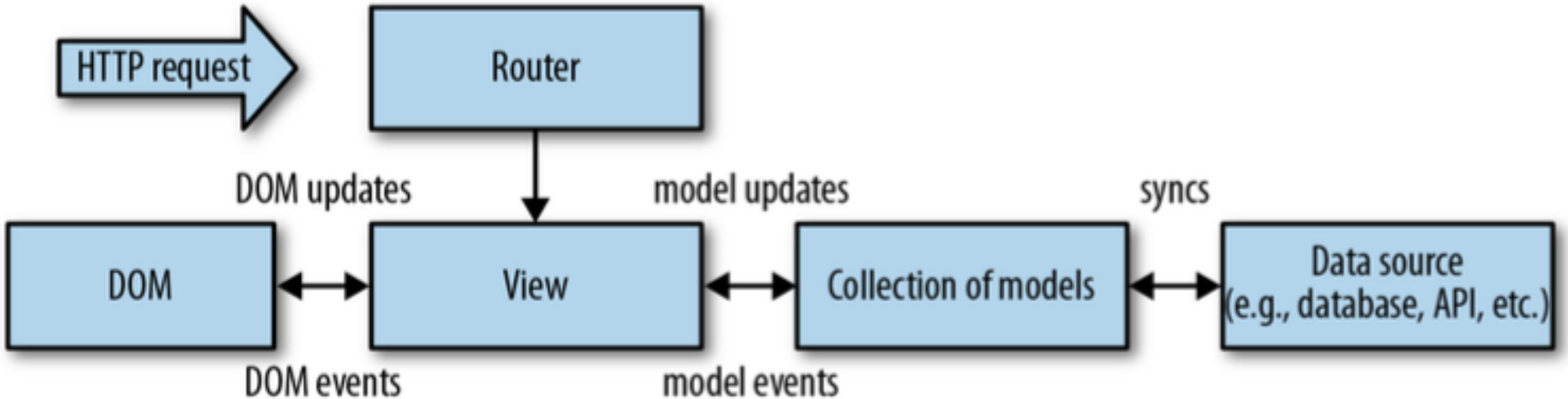
Componentes de Backbone

- **Modelos**
  - *getters* y *setters*
  - validación de datos
  - sincronización automática con el servidor, si este ofrece un API REST
- **Colecciones**
  - Conjunto de modelos del mismo tipo
- **Vistas**
- **Routers**
  - Asociar una URL a cierto código
- **Sistema de eventos**:
  - Sistema *pub/sub*, nos permite que un componente “avise” a otro sin acoplarlos demasiado

¿Quién hace de controlador?

- Recordemos que el papel tradicional del controlador en MVC es **actualizar el modelo en respuesta a los cambios de la vista**
- En backbone este papel se puede repartir entre las vistas y los routers, o bien implementar nuestras propias clases que hagan de controladores y sean accesibles a la vista
- Por eso realmente Backbone no es MVC, sino MV* (o MV-loquesea)

MV* en Backbone



Del libro "**Developing Backbone.js Applications**", *Addy Osmani*
http://addyosmani.github.io/backbone-fundamentals/

Modelos

```
var Libro = Backbone.Model.extend({
        toString: function() {
            var cad = this.get('titulo') + ' por ' + this.get('autor');
            if (this.get('prestado'))
                cad += " - prestado a " + this.get('poseedor')
            return cad;
        },
        prestar: function(nombre) {
            if (!this.get('prestado')) {
             this.set('prestado', true);
             this.set('poseedor', nombre);
            }
        }
   });

var libro1 = new Libro({titulo: "Juego de tronos",
                        autor: "George R.R.Martin"})
libro1.prestar("Pepe");
console.log(libro1.toString());
```

Eventos

- Sistema Publicar/Suscribir
- Los modelos automáticamente emiten eventos cuando sucede algo “interesante”
- Los objetos de Backbone pueden hacer de “oyentes” de forma nativa

```
var observador = Backbone.Model.extend({})
observador.listenTo(libro1, "change:prestado", function(modelo, nuevo_valor) {
        if (modelo.get("prestado")) //también valdría if (nuevo_valor)
            console.log("El libro ha sido prestado")
        else
            console.log("El libro ha sido devuelto")
})
```

Vistas

```
var LibroView = Backbone.View.extend({
    tagName: 'div',
    render: function() {
        this.el.innerHTML = '<span class="libro">' +
                            this.model.toString() + '</span>';
        return this
    }
});
var miLibroView = new LibroView({model: libro1});
$('body').html(miLibroView.render().$el);
```

Demo