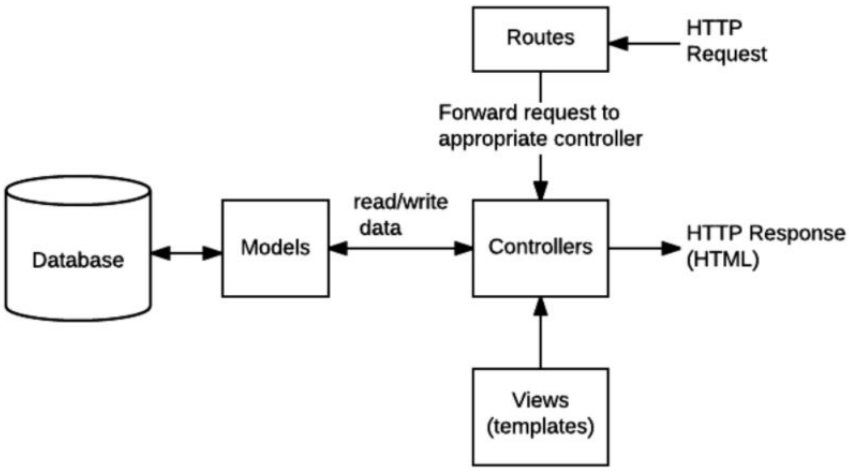
1: From the class:



1. Use JavaScript to implement the business layer of the application (**model**).
2. Use EJS pages to present the **view** to the browser.
3. Use routes to **control** the flow of the application.

2: From my perspective:

如何设计一个程序的结构，这是一门专门的学问，叫做"架构模式"（architectural pattern），属于编程的方法论。

MVC模式就是架构模式的一种，它对我的启发特别大。我觉得它不仅适用于开发软件，也适用于其他广泛的设计和组织工作。

下面是我对MVC模式的一些个人理解，不一定正确，主要用来整理思路

MVC是三个单词的首字母缩写，它们是Model（模型）、View（视图）和Controller（控制）。

这个模式认为，程序不论简单或复杂，从结构上看，都可以分成三层。

1）最上面的一层，是直接面向最终用户的"视图层"（View）。它是提供给用户的操作界面，是程序的外壳。

2）最底下的一层，是核心的"数据层"（Model），也就是程序需要操作的数据或信息。

3）中间的一层，就是"控制层"（Controller），它负责根据用户从"视图层"输入的指令，选取"数据层"中的数据，然后对其进行相应的操作，产生最终结果。

这三层是紧密联系在一起的，但又是互相独立的，每一层内部的变化不影响其他层。每一层都对外提供接口（Interface），供上面一层调用。这样一来，软件就可以实现模块化，修改外观或者变更数据都不用修改其他层，大大方便了维护和升级。

3: Example of the MVC

在我看来，不仅编写程序可以用MVC模式，家用电器也可以用。

以家用微波炉为例，可以将它也理解成三层结构。最简单的情况下，微波炉的操作用两个转盘实现，一个控制温度，另一个控制时间。这两个转盘就是"视图层"（view），而其内部的微波产生装置则是"数据层"（Model），这里的"数据"需要理解成"核心功能"。至于将用户通过转盘输入的信息，转换成对微波产生器的操作，则用"控制层"来实现。

如果每一层都是独立的，那么微波炉外部更换一个新潮的外壳，或者内部更换更大功率的微波产生器，完全可以在不更改其他层的情况下实现。这就是MVC模式的优势。