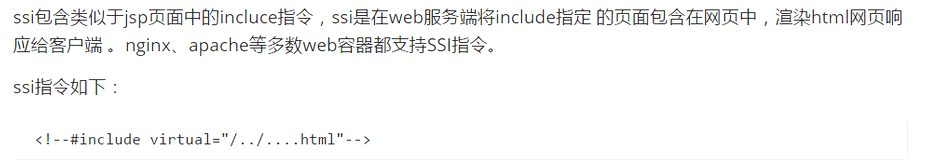
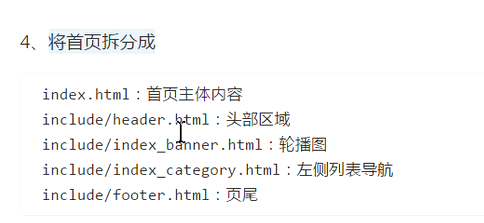
**CMS内容管理系统**

管理网站的静态页面，例如门户页面、课程详情页面。这些页面的内容会由于运营的需要而经常变动，通过该系统可以实现页面的快速开发、预览、发布等功能。

**SSI（server side include）服务端包含技术**

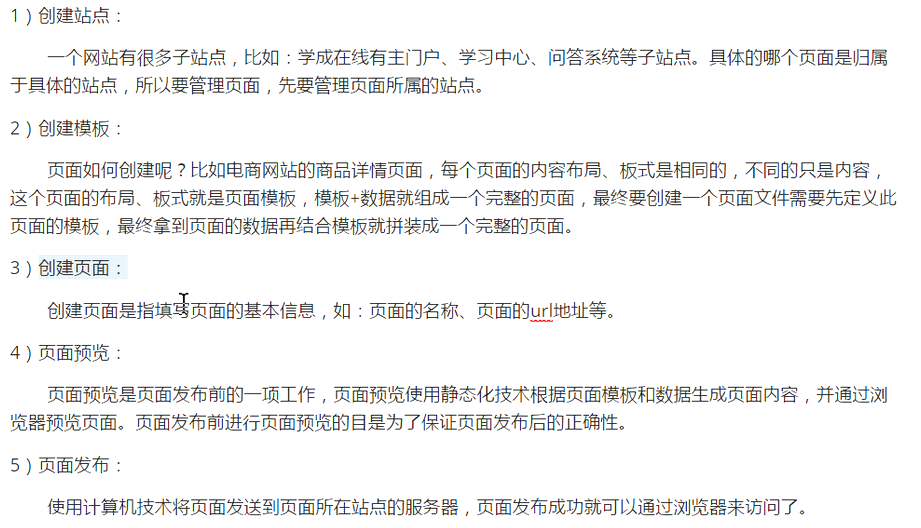


例如门户网站由一个一个的小页面组成



当需要更改部分内容时只需要更改其中的小页面内容即可。页面会在服务端进行拼装，然后返回给浏览器。

**CMS页面管理流程**



本项目实现的功能：页面的增删改查、页面预览、页面发布(例如发布到门户服务器)

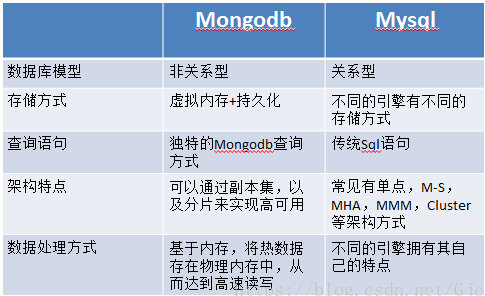
**MongoDB是什么，特点是什么**

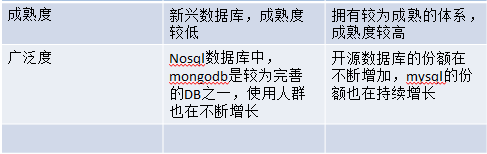
首先，MongoDB是文档型（Document store）的NoSQL数据库，数据以文档（对应关系型数据库的行记录）的形式在MongoDB中保存，文档实际上就是一个个JSON字符串(类似JSON的BSON格式，在JSON基础上开发，有更快的遍历速度，)。使用JSON的好处是非常直观，通过一系列的Key-Value键值对来表示数据，支持丰富的数据结构，Value可以是普通的整型、字符串，可以是数组，也可以是嵌套的子文档，使用嵌套的好处是在MongoDB中仅需一次简单的查询就能够获取到你所需的数据。这些独有属性可以直接以JSON子文档的方式嵌套在这个文档中，一次查询直接获取全部内容，不需要进行多表join；MongoDB文档的另一大特点是模式灵活：不同文档相同key的value类型可以是整形也可以是字符串等其他类型，不同文档可以有不同的key。

**MongoDB与MySQL的对比**

**关系型数据库**可以理解为依赖关系模型来创建的数据库。关系模型就是指二维表格模型,因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据组织。关系型数据库可以通过关系模型使多个表的数据关联起来，比如我们平时说的 一对一、一对多、多对一。由于是建立在数据模型的基础上，所以我们可以通过SQL语句很方便的在多个表之间做复杂的查询操作。关系型数据库相对安全，因为直接存储在硬盘中所以突然的宕机、停电等意外不会导致数据丢失。MySQL的存储方式是由自身的引擎决定的,常用的引擎有Innodb和MyISAM。他们主要的区别就是MyISAM 不支持事务，强调的是性能，执行速度比Innodb要快，Innodb提供支持事务等高级数据库功能。

**非关系型数据库**即我们常说的NoSQL数据库，部署起来都比较简单，没有关系型数据库那么复杂。Mongo的存储方式为虚拟内存+持久化存储，Mongo将数据写入内存中，再由虚拟内存管理器将其持久化到硬盘中，因此写操作会比关系型数据库快很多。NOSQL的存储格式是key-value形式，可以像关系型数据库那样存储基础数据类型的数据，也可以存储集合、对象等等。NoSQL虽然性能比较高，但是并不支持事物，也不能进行联表查询，一般用于较大规模数据的存储。





关系型数据库在使用过程中当数据量到达一定程度时，关系型数据库的效率会有明显的下降。一个复杂的查询操作，一系列的组合索引都会消耗非常多的内存空间，此时我们需要对数据库进行读写分离操作，或者将数据库结构进行拆分(水平拆分、垂直拆分)将请求压力分担在不同的库中。

MongoDB数据是存储在硬盘上的，只不过需要经常读取的数据会被加载到内存中，将数据存储在物理内存中，从而达到高速读写

优点：

快速！在适量级的内存的Mongodb的性能是非常迅速的，它将热数据存储在物理内存中，使得热数据的读写变得十分快。高扩展性，存储的数据格式是json格式！

缺点主要是不支持事务

**为什么使用MongoDB？**

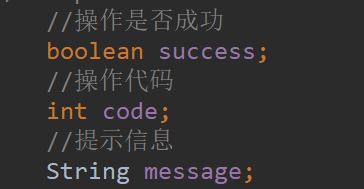
1数据存储灵活，例如我要存储页面数据，每个页面的数据格式都不一样，我可以把不一样的字段都放到一个Map里面进行存储，这样所有页面都可以放在一个集合里面存储。此外还可以很方便的存储数组类型的数据，长度不受限制。例如数据字典中存储的等级、状态类信息，这类数据在前端使用时往往需要返回为List类型。可以在一个集合里面存储项目所需的所有状态信息。

2.使用它的文件系统GridFS存储模板和页面数据。

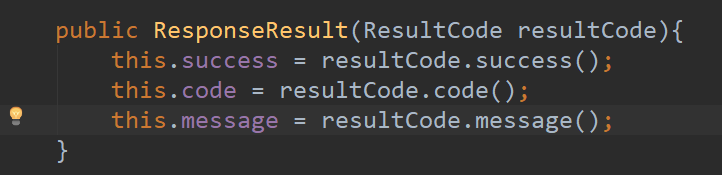
3.想学习一下使用新的数据库

**定义了统一的返回方式**

定义了公用的ResultCode接口，作为返回值中的状态信息部分。使用枚举类型去实现它。



ResponseResult使用ResultCode初始化。



相应结果为：是否操作成功、操作代码、提示信息及自定义数据。

**为什么在api工程中统一定义接口**

1.方便统一管理

2.以后调用微服务是基于接口的，所有微服务统一依赖该api接口工程即可，不用去依赖具体的微服务。

**前端跨域问题解决**

原因：浏览器的同源策略不允许跨域访问，所谓同源策略是指协议、域名、端口相同。

解决：采用proxyTable

