1. Web开发学习路线

1. HTTP
2. 前端开发
    1. HTML & CSS
    2. 脚本语言 & JavaScript & JS框架
    3. React框架 & Vue框架
3. 后端开发
    1. Servlet
    2. Java访问数据库-JDBC
    3. Java访问数据库-ORM
4. 前后端集成
    1. Ajax & JSON
5. 框架运用
    1. Spring 服务
    2. MVC 架构
6. 管理工具
    1. Maven
    2. Webpack
7. NoSQL-MongoDB
8. 微信小程序
9. Android & iOS

2. Web应用的发展历史

1. 什么是Web application?
    - 是一个**计算机程序**，允许用户使用**自己喜欢的浏览器**，通过**互联网**，从**数据库**中，**提交和访问数据**。
    - Static Website Browser从Web server(Apache)拿到静态的HTML页面
    - Web 1.0
        - 基于CGI(Common Gateway Interface公共网关接口)与文件系统的交互，使用ASP.NET/JSP开发，实现了动态的web应用
        - 不分前后端
    - Web 2.0
        - Ajax的出现和计算能力的增强，使得浏览器包含了更多的代码；
        - CDN(Content Distribute Network内容分发网络)；
        - MVC模式的应用提高了可维护性
            - 后端：Model-Beans，View-JSP，Controller-Servlet
        - MVC和Ajax联合后，组装页面的工作交给了前端，后端的View消失，前端出现了MVC,前后端使用JSON通信
            - 前端：Model-JSON Data，View-Templates，Controller
    - Full-stack JavaScript & REST API
        - Node.js
        - MongoDB-JSON

- HTML5+CSS3
- Apache HTTP Server
- PHP 是一种尤其适合网站开发的脚本语言
- Ruby on Rails
- Web Frameworks for Python
  - Django
  - TurboGears 2
  - Web2py
- node.js 轻量级框架，生态很好，新兴技术
- Windows .NET 重量级框架
- Erlang,Scala,Go

1. HTTP协议

- 是一种request-response的协议
- browser是一种UA(user agent用户代理)
- URL(Uniform Resource Locators)
  - URI = URL + URN
  - scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
  - hash(#)可以用于单页面内的路由和定位
- Request Methods

HTTP Method	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET	N	Y	Y	Y	Y
HEAD	N	N	Y	Y	Y
POST	Y	Y	N	N	Y
PUT	Y	Y	N	Y	N
DELETE	N	Y	N	Y	N
CONNECT	Y	Y	N	N	N
OPTIONS	Optional	Y	Y	Y	N
TRACE	N	Y	Y	Y	N
PATCH	Y	Y	N	N	Y

- HTTP session
  - TCP三次握手(SYN, SYN-ACK, ACK)
    - 第一次握手：客户端发送syn包(seq=x)到服务器，并进入SYN_SEND状态，等待服务器确认;
    - 第二次握手：服务器收到syn包，必须确认客户的SYN(ack=x+1)，同时自己也发送一个SYN包(seq=y)，即SYN+ACK包，此时服务器进入SYN_RECV状态;
    - 第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=y+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

    - HTTP是无状态的协议：Server端的HTTPSession存放了相关信息，然后把SessionID作为cookie传回给Client端，之后Client端带着cookie访问Server并识别谁是谁
- REST 特征
    - Client-Server隔离对待
    - 无状态
    - 可缓存
    - 分层系统
    - 统一接口
    - 支持按需代码

      Client request

      ```
      GET /index.html HTTP/1.1
      Host: www.example.com
      ```

      Server response

      ```
      HTTP/1.1 200 OK
      Date: Mon, 23 May 201822:38:34 GMT
      Content-Type: text/html; charset=UTF-8
      Content-Encoding: UTF-8
      Content-Length: 138
      Last-Modified: Wed, 08 Jan 201823:11:55 GMT
      Server: Apache/1.3.3.7  (Unix)  (Red-Hat/Linux)
      ETag: "3f80f-1b6-3e1cb03b"
      Accept-Ranges:  bytes
      Connection:     close

      <html>
          <head>
              <title>An Example Page</title>
          </head>
          <body> Hello World, this is a very simple HTML document.</body>
      </html>
      ```

2. HTML

- Hypertext Markup Language
- HTML tags + text = HTML Document => Web Browser => Webpage

3.CSS

- Cascading Style Sheet

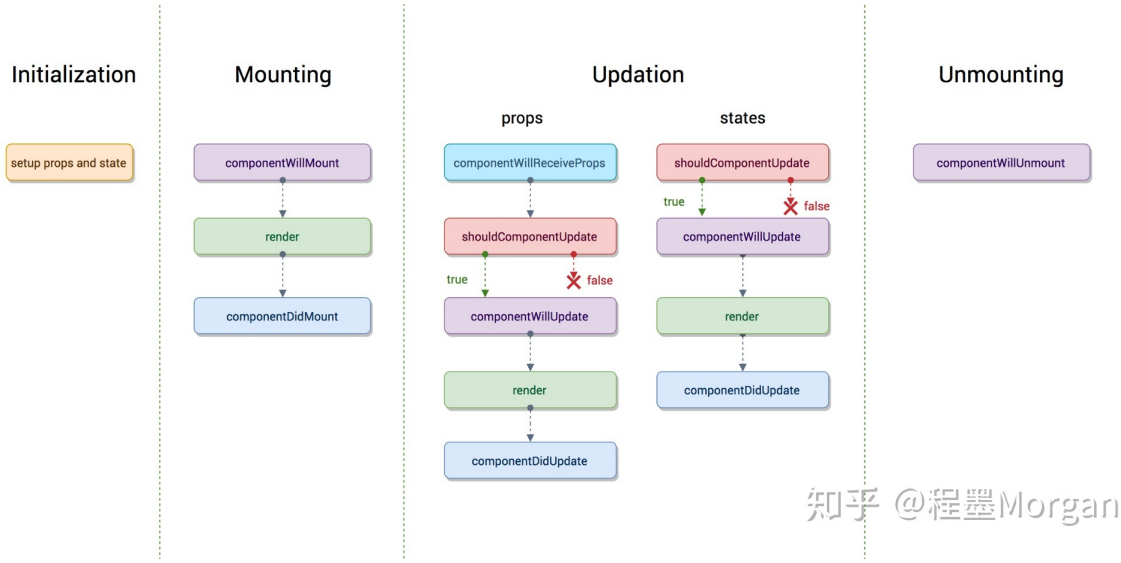
4.JavaScript

- script是运行前不需要预处理的程序代码
- 基于DOM树操纵html页面
- JavaScript是ECMAScript中最著名的实现
- 独立于平台的
- 需要考虑各种不同浏览器、不同语言版本的兼容问题。解决方案：
    - 最小公分母的方案
    - 测试特性是否存在
    - 写特定平台下的代码
    - 利用语言特性
    - 显式版本测试
    - 优雅报错
- Closure Compiler: 优化代码，去除冗余
- AngularJS
- jQuery
- BootStrap
- React

5. React

- 出现的背景：
    - 新型网页开发模型SPA(单页模型)，只需拉取必要的数据在brower端进行绘制，减少了和服务端的延迟
        - 缺陷：
            1. 如何保持数据和UI同步更新
            2. 如何提高DOM操作的效率
            3. 使用HTML开发UI界面异常复杂
    - 解决方案：
        1. 自动化的UI管理，数据和界面的同步，把数据变化转化为一系列事件，开发者只需根据事件转换界面的状态，实现了数据和UI的解耦
        2. 更高效的DOM操作，内存中的Virtual DOM进行缓存，定期对实际中DOM进行更新
        3. UI的组件化设计，大量可重用的组件
        4. 依赖JS开发UI界面，创新了JSX看起来像HTML，但本质上会解读为一系列DOM操作，简化了代码逻辑
- 本质是MVC中的View，将数据和UI结合
- props和state
    - function and class必须永不修改自己的props；所有的components必须像pure functions一样尊重它们的props
    - State与props相似，但它是私有的，并且完全由component自己控制；local state可以抽象为仅对class可见的特性
    - state的更新是异步的，为了提高性能可能会打包处理
- 生命周期

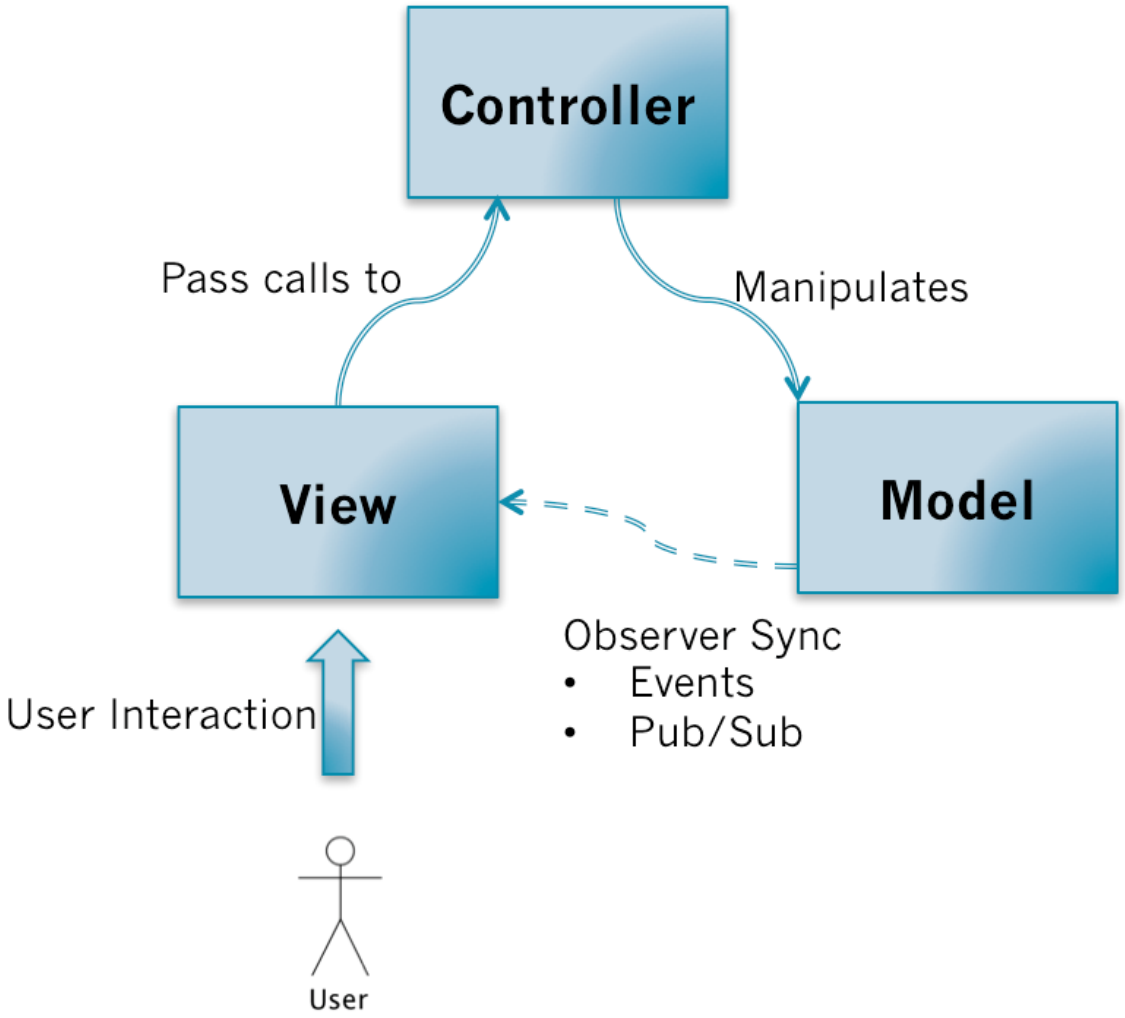
- WillMount
- DidMount
- WillUnmount
- WillUpdate
- DidUpdate
- ShouldUpdate
- WillReceiveProps



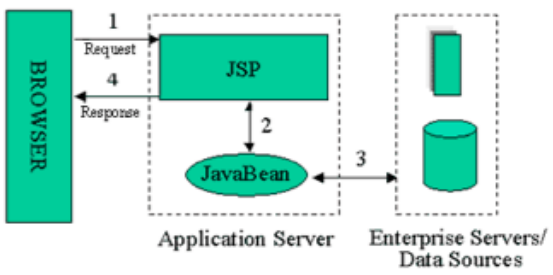
- 组件化的思想
    1. 代码重用
    2. 减少耦合
    3. 降低复杂度

6. Vue

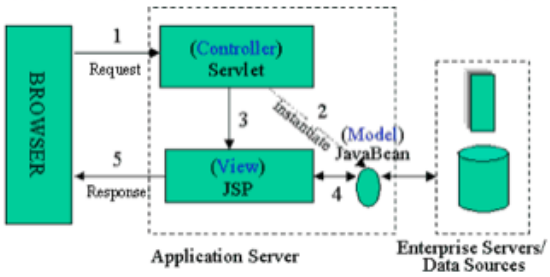
- 与React是类似的
- MVVM(Model-View-ViewModel)
    - MV*针对的是：GUI程序
    - MV*的基本模式是：Model异步通知View变更，View告知Model操作逻辑
    - MV*解决的问题是：View如何同步Model的变更
    - MVC中View把控制器交给Controller，Controller操纵Model，Model和View同步消息采用观察者模式



- Web服务端的MVC事实上是JSP Model 2，在这种情况下，观察者模式被迫打破，必须由View主动发出请求

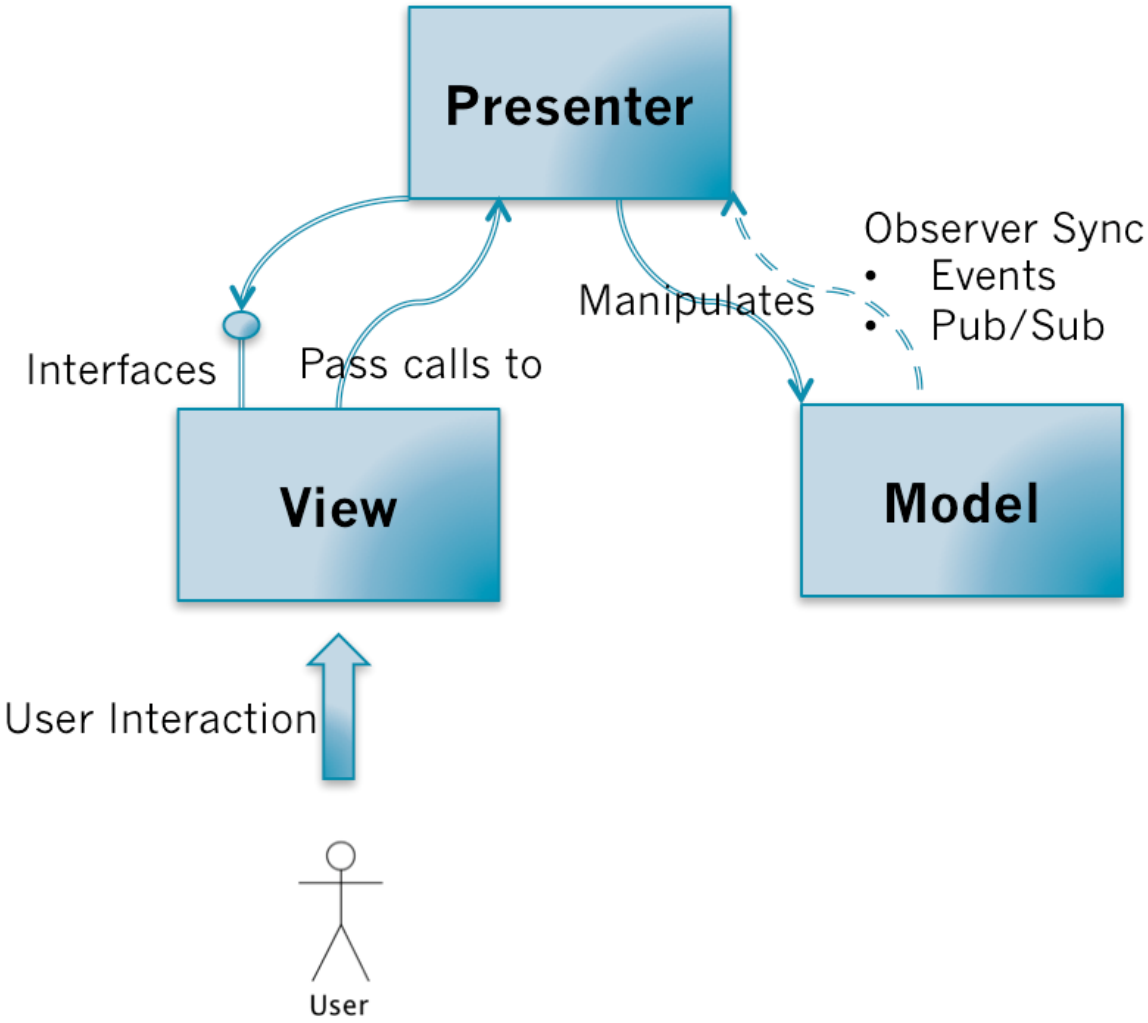


JSP Model 1

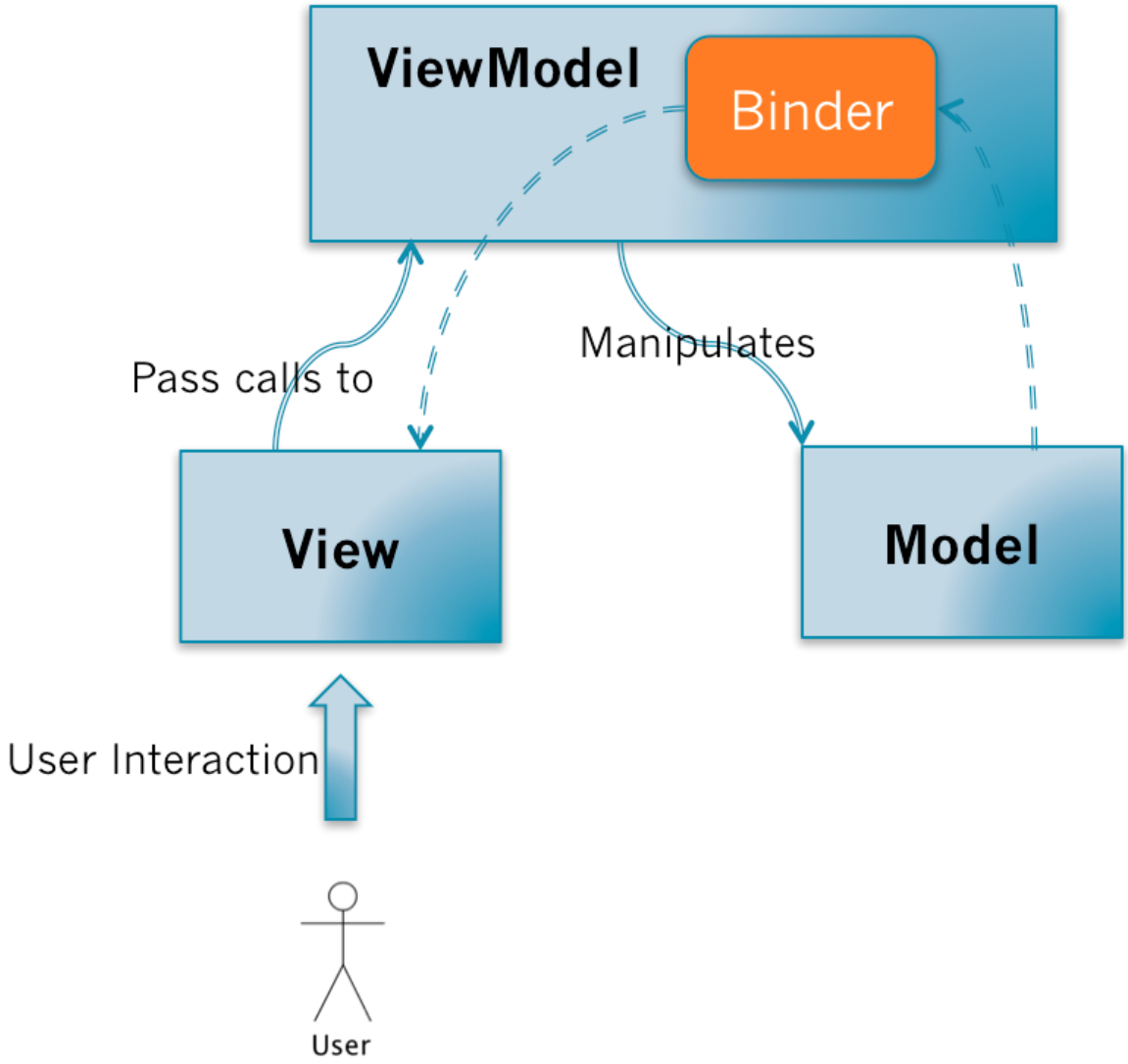


JSP Model 2

- MVP中把Controller换成了Presenter，打破View和Model之间的依赖关系，View只需提供接口，全权由Presenter负责同步和逻辑

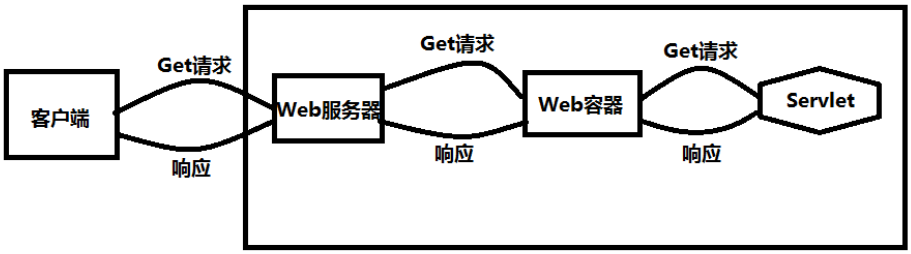


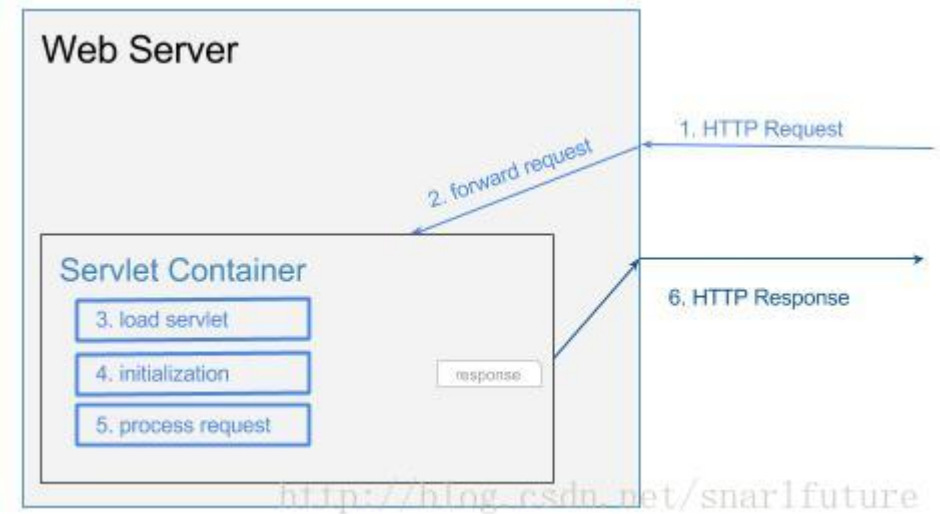
- MVVM是一种特殊的MVP，在ViewModel中新增了Binder(或称数据绑定引擎)，实现了View和Model双向数据绑定的自动化。



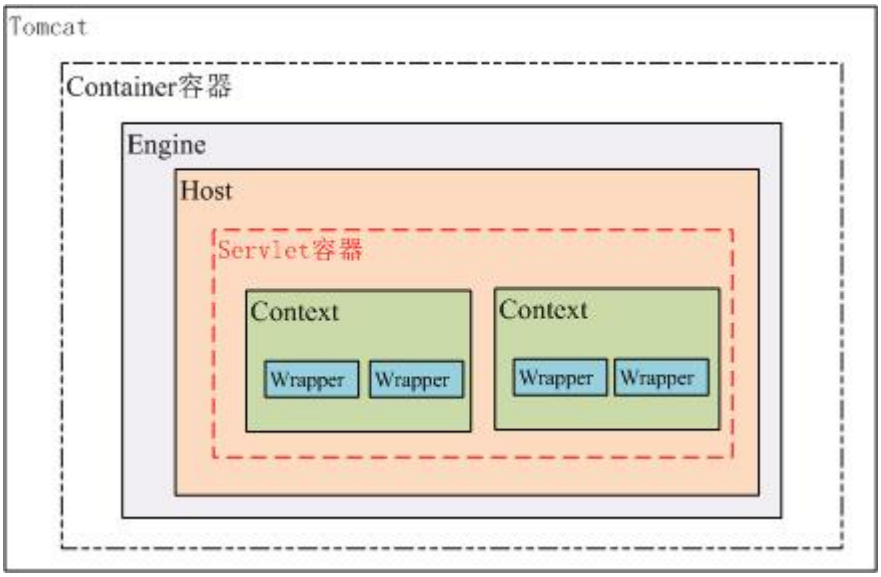
7. Servlet

- 本质是是一个Java类，受控于Container，用于基于request-response的server，HttpServlet是针对Web应用的特殊化。
- Web Server
  - 使用HTTP协议来传输数据
- Servlet Container
  - 本质是一个Java程序，是用来装Servlet的
  - 基本思想是在服务器端使用Java来动态生成网页
  - 控制Servlet对象的生命周期(类加载、实例化...)，调用相应的方法和传递request&response实体，如init(),service(),destroy()





- Tomcat
  - Tomcat 的容器等级中，Context 容器是直接管理 Servlet 在容器中的包装类 Wrapper，真正管理 Servlet 的容器是 Context 容器



- Filter：是一个对象，可以对指定url的requset&response处理其header&content
- HttpSession: 相同Web context下的任何组件都可以访问session内容，并处理同一会话的内容
- 上传文件：MIME type - multipart/form-data

8. Access to RDBMS

- JDBC：提供了用Java访问关系数据库的API
- 基本流程
    - 建立Connection
        - DriverManager
        - Datasource：使用JNDI(Java Naming and Directory Interface)
    - 执行SQL Statement并操纵Result
        - DatabaseMetadata
        - Statement(直接写SQL)，PreparedStatement(设置输入参数)，CallableStatement(调用Procedure获得返回值)
        - ResultSet，RowSet
- Context：就是容器，是一个对象，包含一系列Binding(name,object)的集合
- JDBC 的优点
    - 性能好，尤其适合访问大量数据
    - 利用了DBMS提供的各种function
    - 可以使用procedure完成负责逻辑
- JDBC 的缺点
    - 与DBMS紧耦合
    - 与数据结构紧耦合
    - 编程很复杂
- 为解决以上缺点提出了O/R Mapping
- ORM 的优点
    - 独立于DBMS
    - 独立于数据结构
    - OOP(面向对象编程)
- ORM 的缺点
    - 影响性能
    - 不能利用额外的function
    - O和R的映射可能很复杂
    - 不能调用procedure
- JDBC reading v.s. ORM： Requirement Driven

9. AJAX

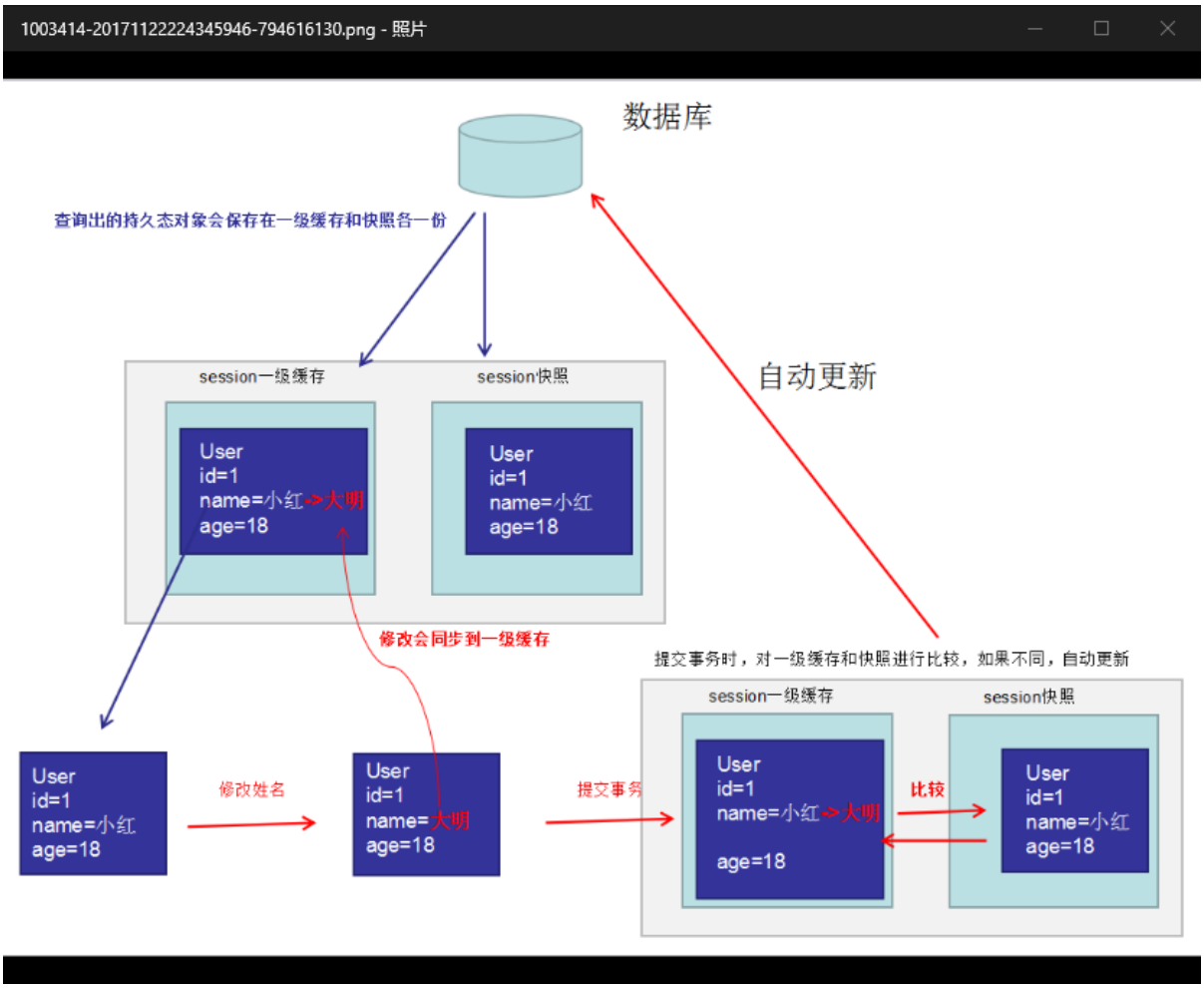
- AJAX = Asynchronous JavaScript and XML
- XMLHttpRequest是Ajax的基础，实现了数据传输
- Ajax的request由JavaScript代码触发，并通过触发回调函数接受response
- GET/POST的选择基于是否改变数据，GET可能被browser缓存，通常以query string的方式发送数据；POST通常不会被缓存，更倾向于以post data的方式发送数据
- jQuery 提供Ajax支持，抽离出浏览器的差异

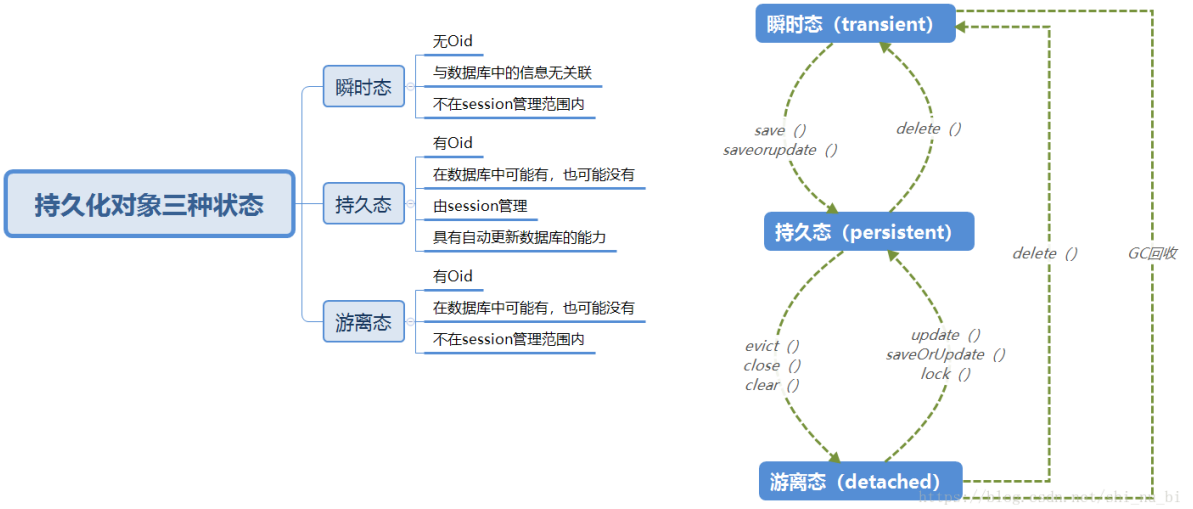
10. JSON

- 是一种基于文本的数据交换格式，由JavaScript而来，被用于web服务与其他应用连接
- 两种数据结构：object(键值对)和array(一组值)
- 六种数据类型：string,number,object,array,true&false and null
- JSON的序列化与反序列化
    - object model：在内存中创建一颗代表JSON数据的树
    - streaming model：使用一种基于事件的解析器，一次只读一个JSON数据

11. ORM

- entity是一个由Hibernate持有的常规Java对象(POJO-Plain Old Java Object)
- 继承策略
    1. table per subclass - 每个子类一张表，继承关系通过JOIN实现，需要同时操作子类和所有父类，通过外键标识继承关系
        - 关系模型完全标准化
        - 策略复杂，性能很低，需要大量JOIN
    2. table per concrete class - 每个具体类一张表，每张表包含所有父类字段
        - 有冗余
        - 完全抛弃继承关系，不支持多态
        - 对父类的查询，需要检索所有子类
        - 父类的修改会导致所有子类的修改
        - 除非必要，否则绝不推荐
    3. table per class hierarchy - 单表保存所有基类和子类的字段，用一个专门的列来标识类别，支持了多态
        - 最简单，性能最好
        - 子类的属性在数据库中必须被定义允许为空，失去对Not Null的约束
- Hibernate 对象状态
    - Transient：没有持久化标识OID，没有被Session管理
    - Persistent：有持久化标识OID，已经被Session管理
    - Detached：有持久化标识OID，没有被Session管理
    - 执行save()后的持久态对象只是放到Session中管理，同步到数据库要等到事务提交。提交前会比对一级缓存和snapshot，需要时才更新数据库。

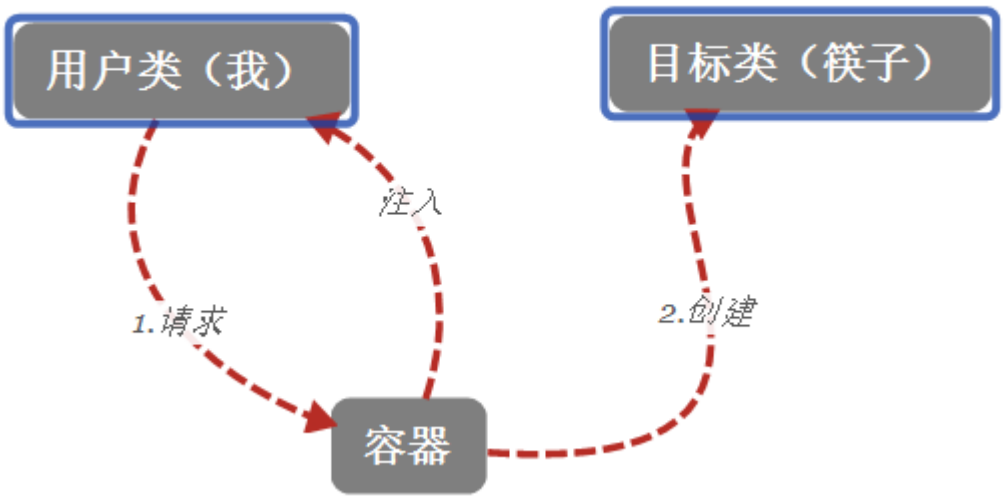




- flush Session默认发生在：
  - 一些query执行前
  - org.hibernate.Transaction.commit()
  - Session.flush()
- SQL Statement被提交的顺序
  1. entity insert按照Session.save()的顺序
  2. entity update
  3. collection deletions
  4. collection element deletion,update,insertion
  5. collecion insertions
  6. entity delete按照Session.delete的顺序
- 级联持久化

12. Spring JPA & IOC

- JPA(Java Persistence API)
  - 提供了对象持久化的规范
  - Hibernate框架是Java环境下ORM的解决方案，提供了JPA的参考实现
  - Spring Data JPA是一个数据访问的抽象，用于减少数据访问层的代码量
- Layered Architecture
  - 接口和实现分离
  - Entity - 从数据库中自动映射
  - Repository - 从现有的lib类中拓展
  - Dao - 自己的访问控制逻辑
  - Service - 业务逻辑
  - Controller - 分发请求
  - IoC/DI - 独立于实现
- Spring Framework: 是一个提供丰富基础设施的Java平台，可以从POJO构建应用程序，并把企业级服务非入侵地应用到其中。
- Spring platform 的优势
  - 无需处理事务API，就可使Java方法在数据库事务中执行
  - 无需处理远程API，就可使本地Java方法成为远端过程
  - 无需处理JMX API，就可使本地Java成为管理操作
  - 无需处理JMS API，就可使本地Java成为消息处理器
- 核心技术 - IoC container
  - 一个Spring IoC container管理一个或多个beans
  - bean的创建使用了应用到这个容器中的配置文件数据
  - IoC/DI是同一思想下不同维度的表现
    - IoC强调的是我不需要new，只需告诉容器我需要new，我的对象依赖容器获得
    - 容器把对象注入给我的过程，就是DI
- Dependency injection 基于Constructor还是基于Setter
  - 对于强制参数依赖使用Constructor
  - 对于可选依赖使用Setter



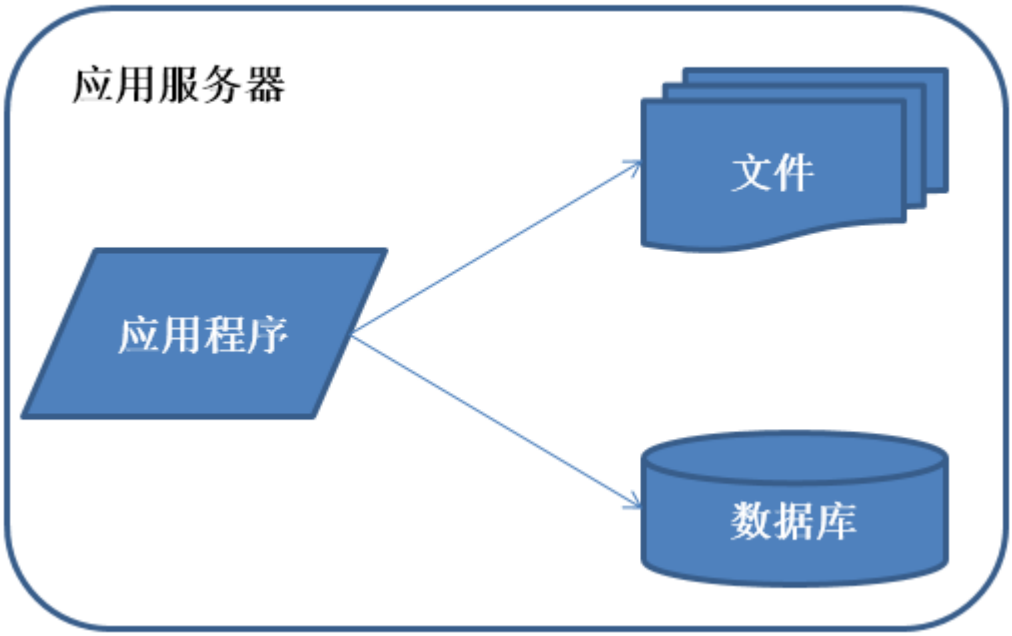
13. NoSQL & MongoDB

- 背景
    - 数据量越来越大，访问速度却有限，如果能能同时访问多个磁盘，就可以提高访问速度
    - 需要解决硬件的故障问题，因为硬件越多，发生故障概率越大，为避免数据丢失的通用方案是复制
    - 第二个需要解决的问题如何结合不同磁盘上的数据，做到这一点是出了名的难，MapReduce是一种解决方案
    - RDBMS的情况
        - 单表：当一条语句执行时，为保证数据完整性表会被锁掉，性能很差
        - 水平分表：基于rows,规则如Range,Hash,Key,List and Composite
        - 垂直分表：基于columns，需要手动实现，但可以显著改善性能
        - 分区的原因
            - 磁盘seek速度与tranfer速度相比慢得多
            - 传统B-Tree的数据模式难以胜任太大的数据频繁更新
        - 但对于相互关联的表，RDBMS很难分表
        - 此外，对于半结构化和非结构化的数据，更难以处理
- NoSQL DBMS
    - Bigtable：按行按列和时间戳键值索引
    - Dynamo
    - Cassandra
    - MemcacheDB
    - Apache CouchDB
    - MongoDB：面向文档的数据库，能自动分发文档，实现数据和负载均衡
- Sharding
    - 是MongoDB实现拓展的方法，可以增加更多的机器来处理大量数据
    - MongoDB支持Autosharding，省去了人工的管理
    - 通常不需要shard，需要的时候再转换为shard
    - 以下情况需要shard
        - 当前机器空间用尽
        - 想要更快的写速度
        - 想要再内存中持有更大量的数据以提升性能
    - Shard Key
        - 设置切片时需要选择一个key作为依据切分数据
        - 当增加和取消shard时，MongoDB会重新平衡数据切分
    - shard切分的结果叫Chunk

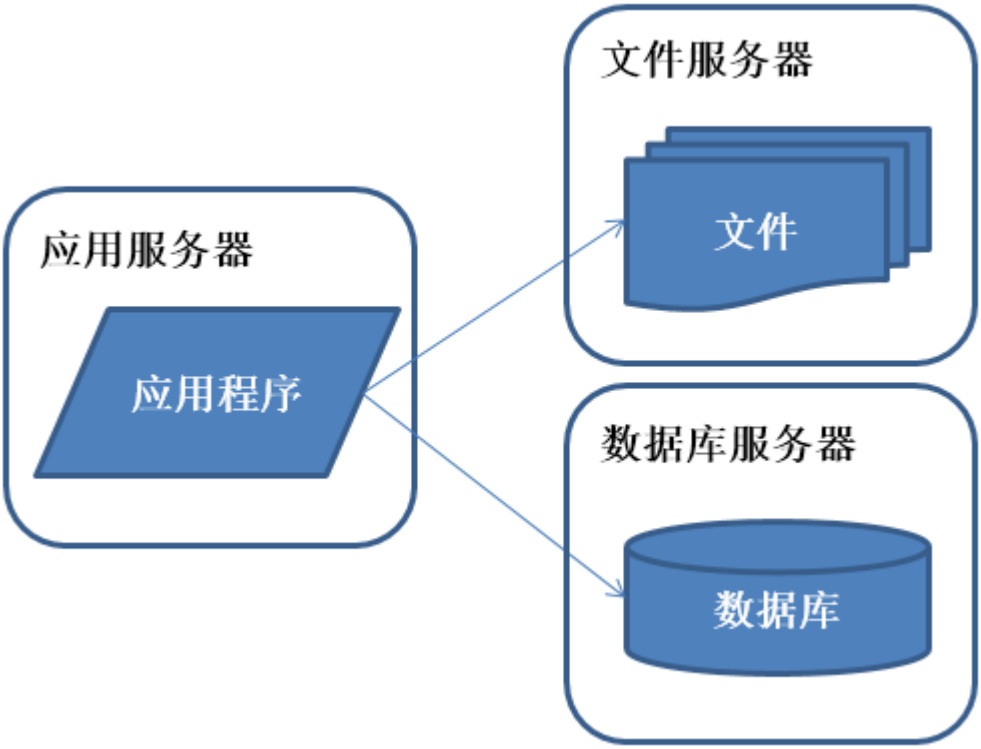
14. Web 应用架构

1. 不同系统不同语言之间的交互 Web service
2. 不同系统相同语言之间的交互 RPC(远程过程调用)/RMI(远程方法调用)
3. 单个产品的架构演进

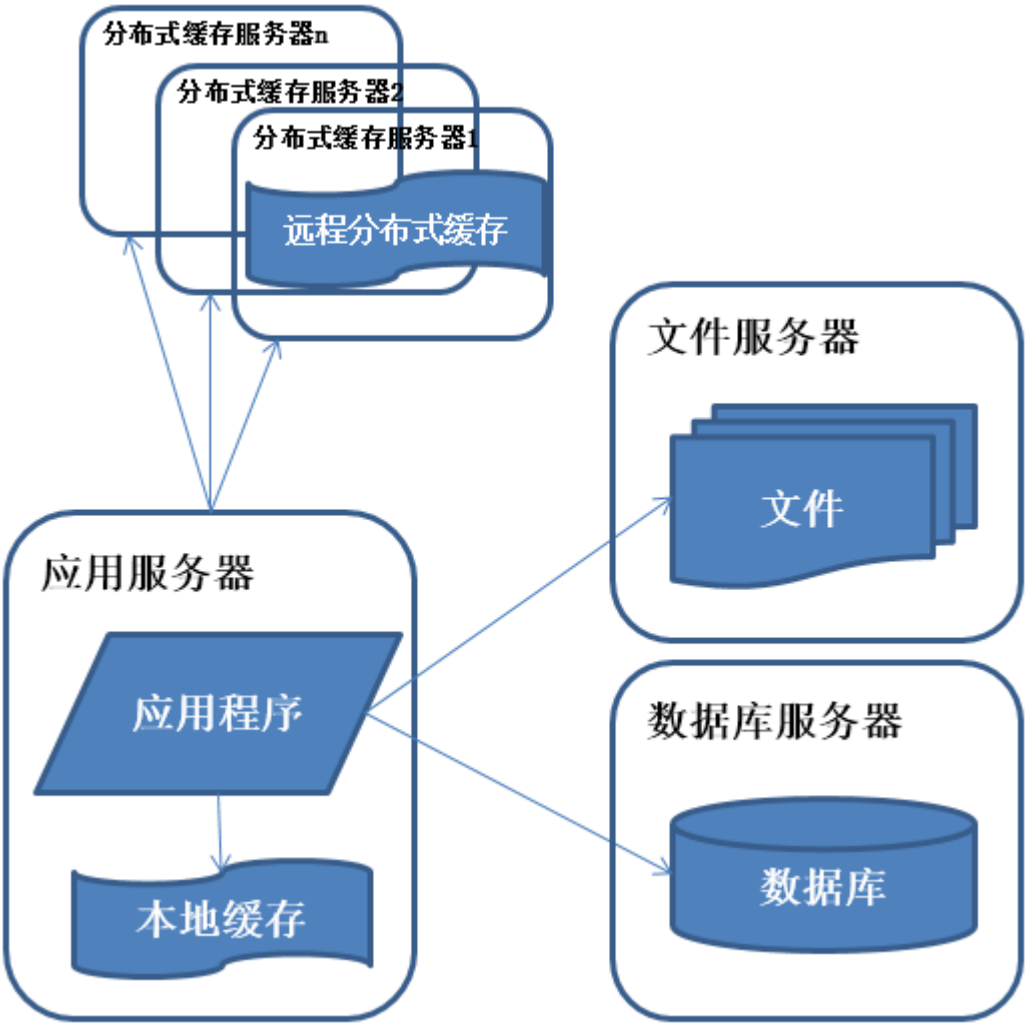
1. 分布式架构的演进系统架构演化历程-初始阶段架构



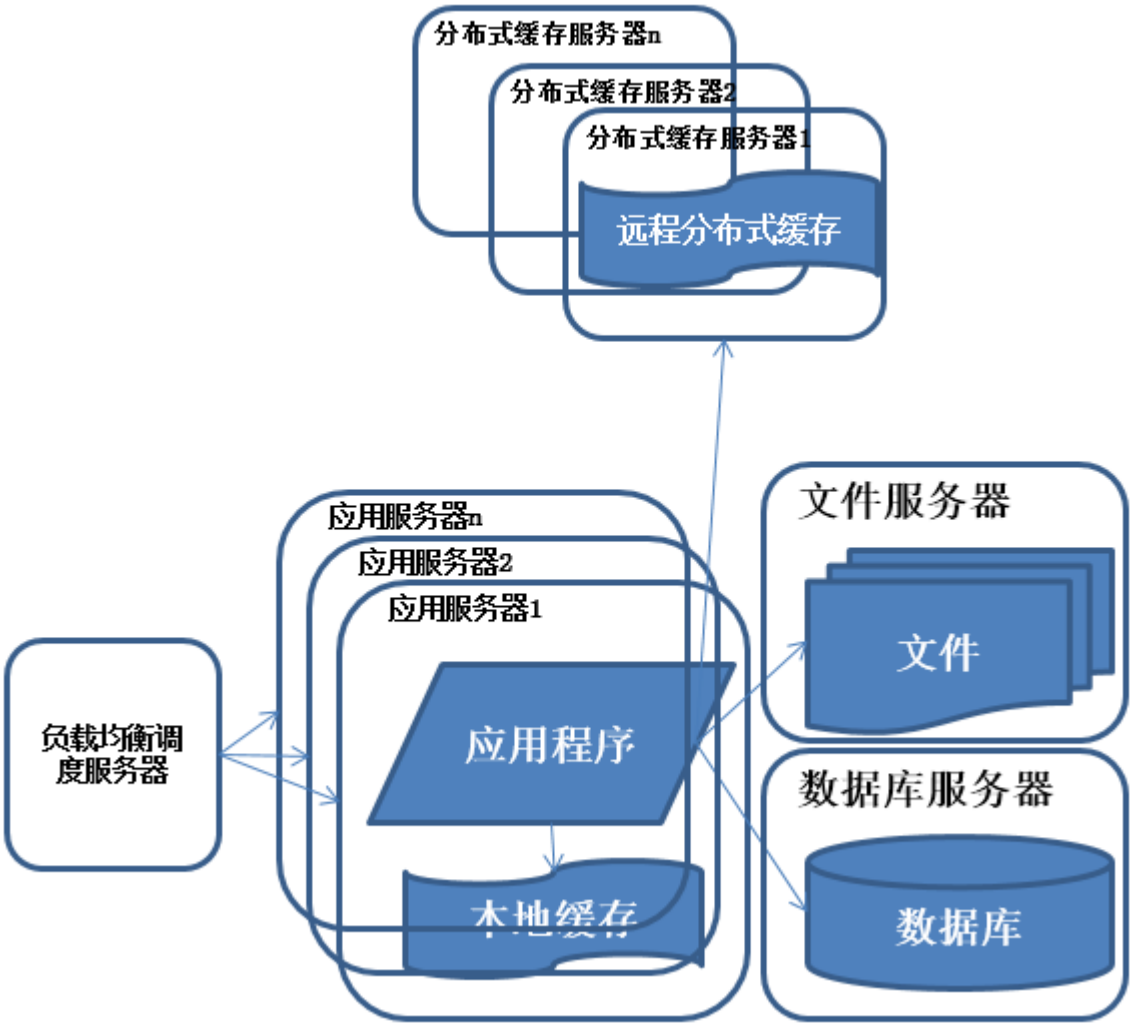
2. 系统架构演化历程-应用服务和数据服务分离



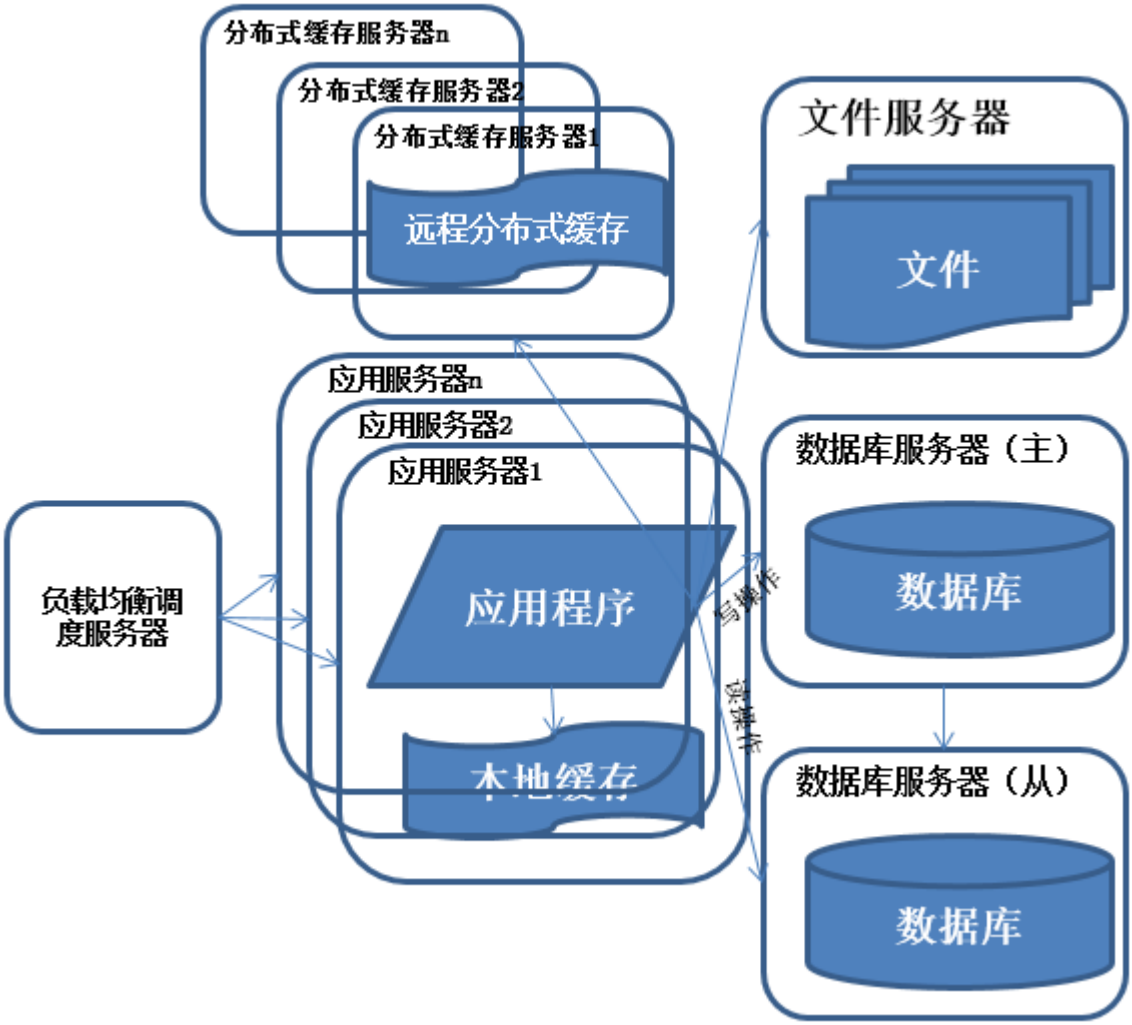
3. 系统架构演化历程-使用缓存改善性能



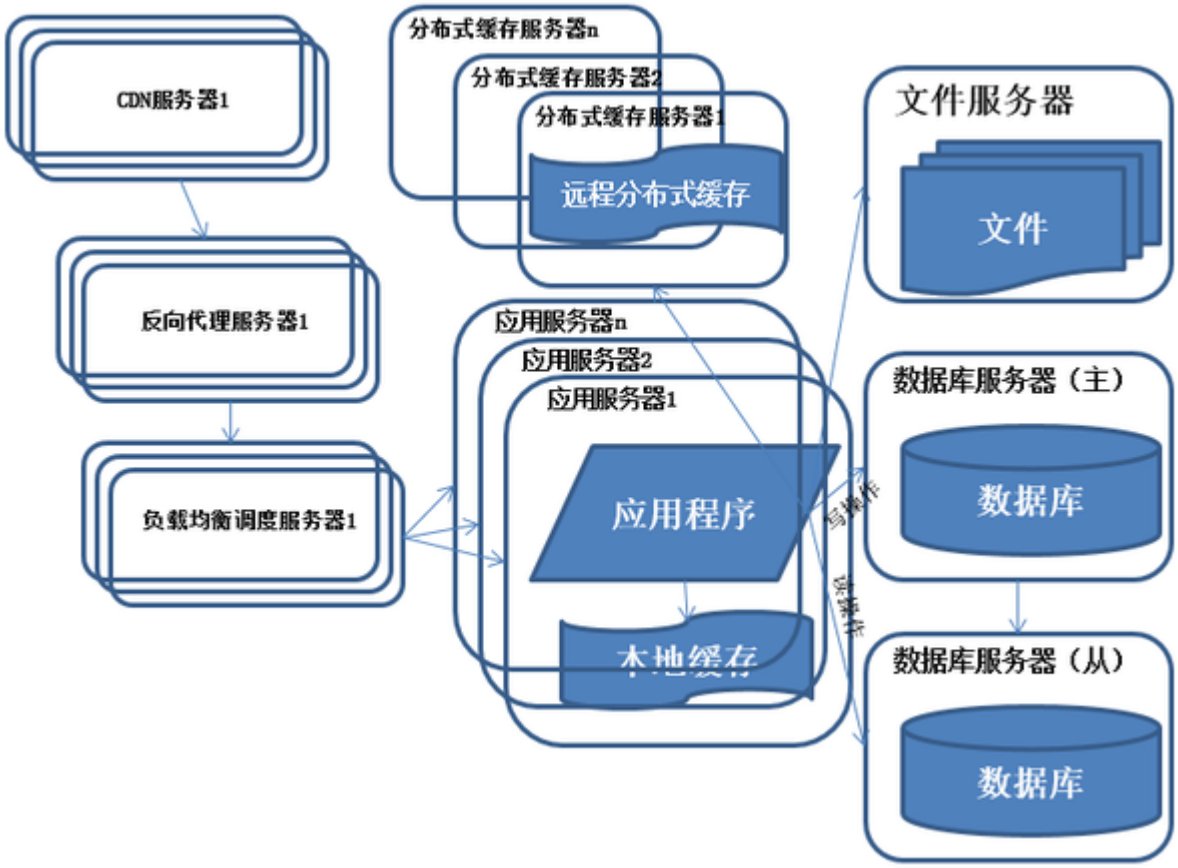
4. 系统架构演化历程-使用应用服务器集群



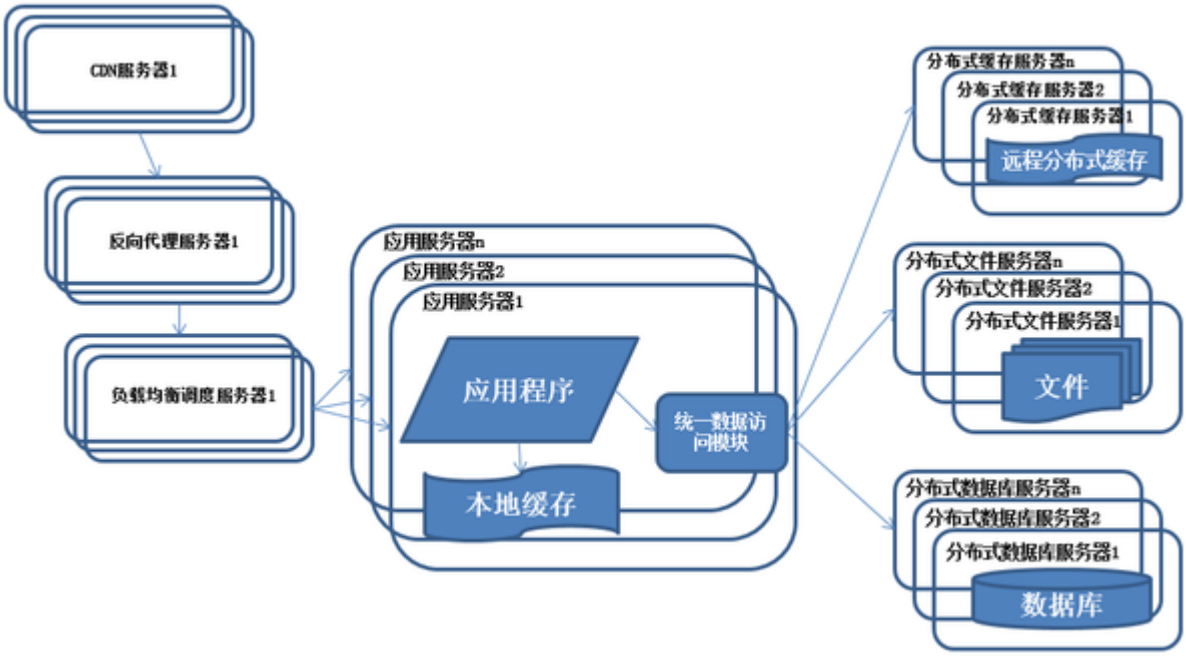
5. 系统架构演化历程-数据库读写分离



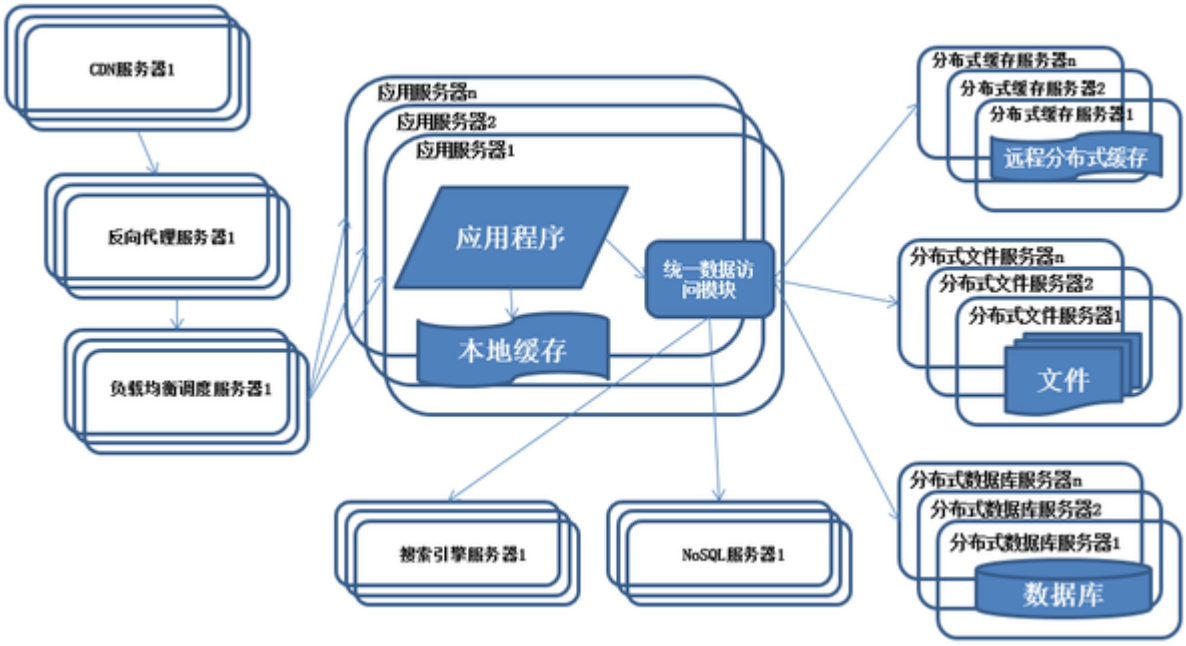
6. 系统架构演化历程-反向代理和CDN加速



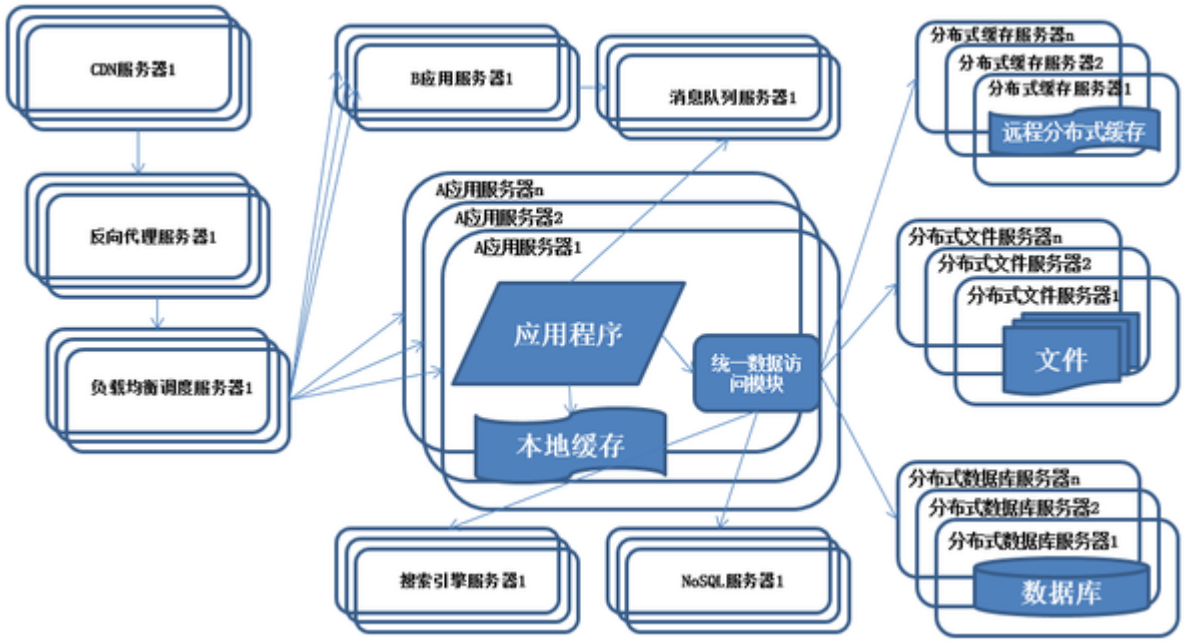
7. 系统架构演化历程-分布式文件系统和分布式数据库



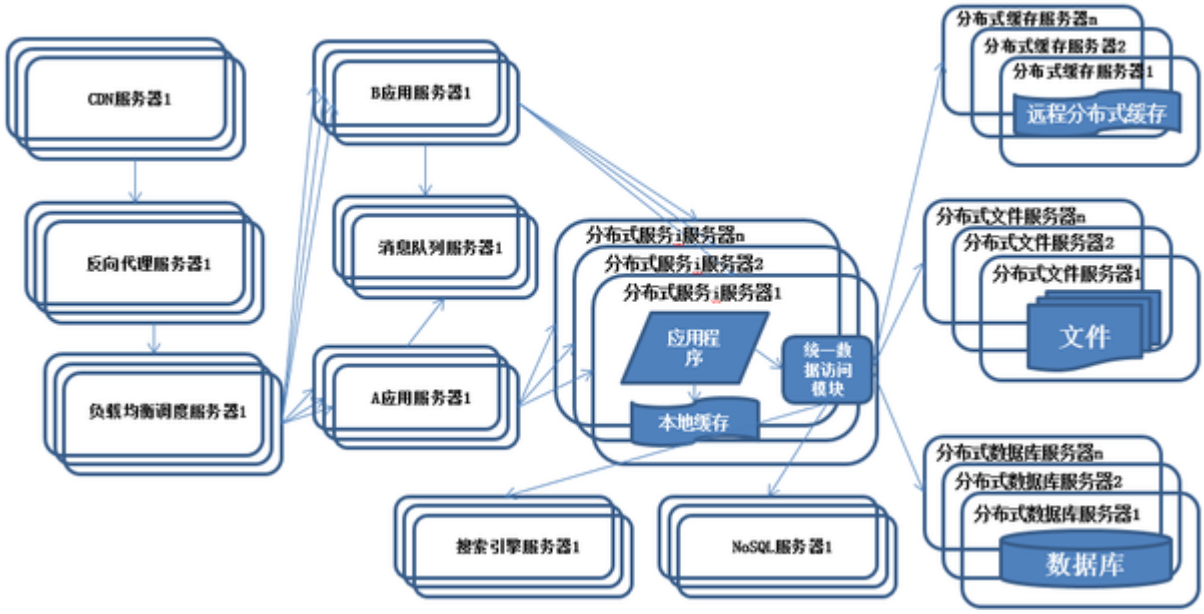
8. 系统架构演化历程-使用NoSQL和搜索引擎



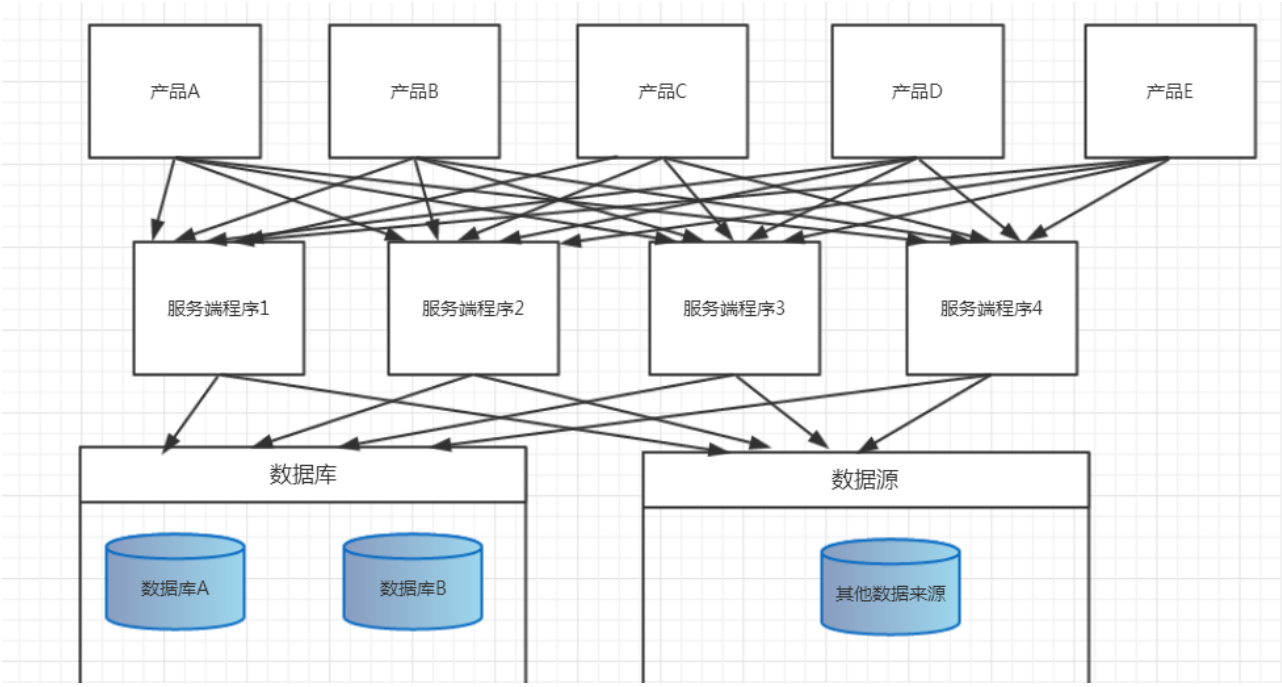
9. 系统架构演化历程-业务拆分



10. 系统架构演化历程-分布式服务



4. 产品线的架构 EJB



15. Browser 中的安全

- XSS(Cross-site scripting跨站点脚本攻击)是一种web应用中典型的电脑安全漏洞
- 预防
  - 防止输入输出的编码转义
  - 安全验证不信任的HTML输入
  - 禁用脚本
  - 新兴技术：Content Security Policy, Javascript sandbox, auto-escaping templates
- SQL Injection 是一种SQL注入攻击
- 类型
  - 不正确过滤转义字符
  - 不正确的类型处理

    - SQL盲注
    - 条件响应
- 缓解方案：
    - 使用参数化的Statement
    - 处理所有特殊意义的字符
    - 类型检查
    - 数据库权限
- 通过设置多个子域名绕过Web Storage限制，实现无限量存储