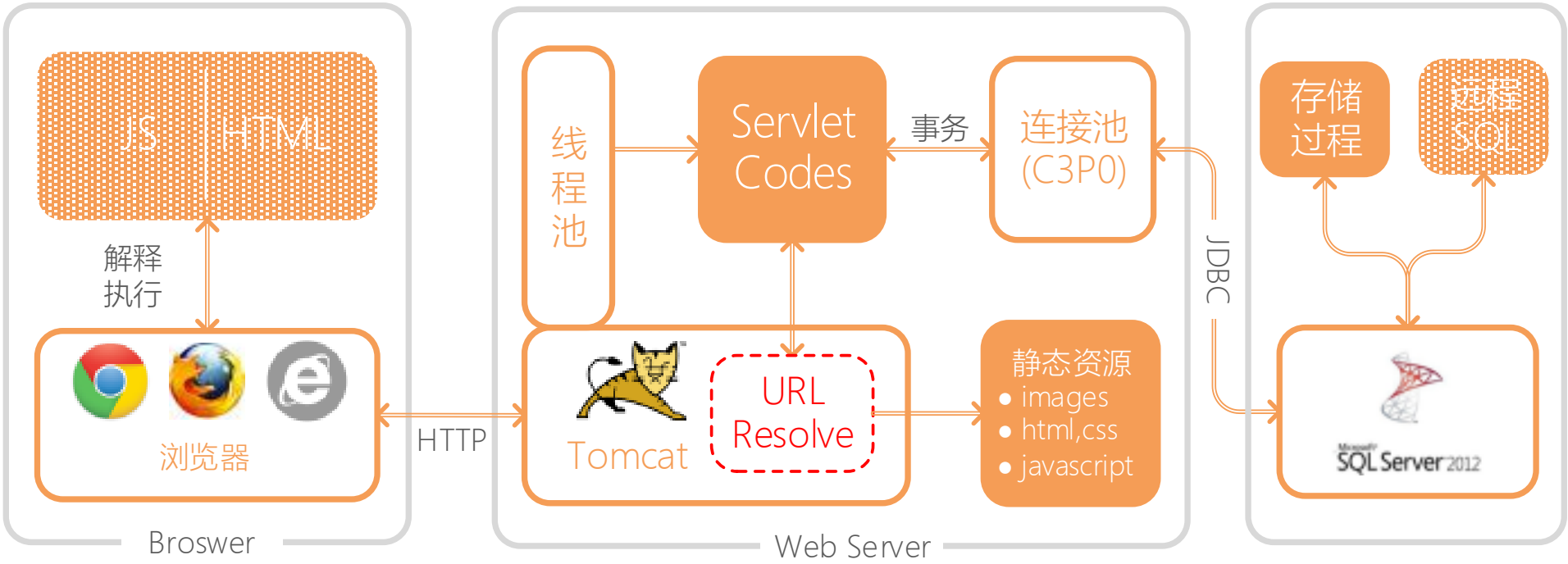
WEB应用开发原理

Frame Work: big picture



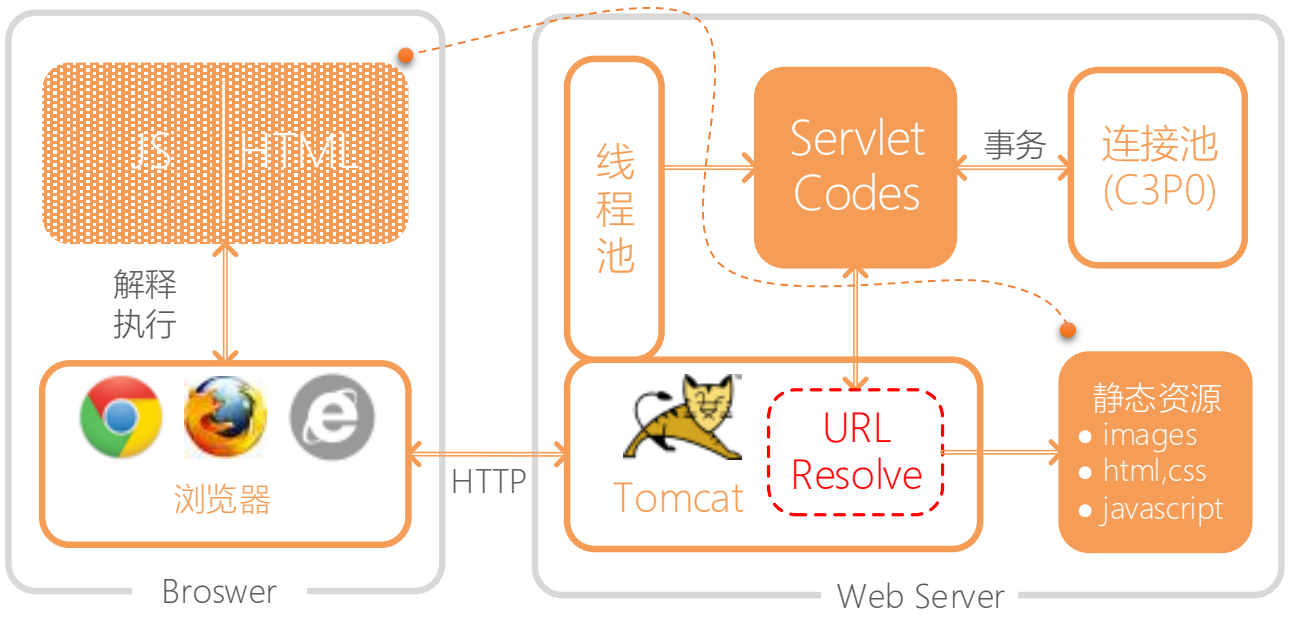
- 3个主体（逻辑上，物理上），2个**接口**
  - HTTP（Servlet）：B(browser) + W(Web Server)
  - JDBC: W(Web Server)+ D(Data Server)
- 区域
  - （不 变）基础设施 ：规定我们编程的模型
  - **（ 变 ）可编程（可控制，our works）**

3个规范：
- HTTP
- Servlet
- JDBC

导航

- HTTP
  - Browser 视角
    - 请求：同步 / 异步
  - Web Server 视角
    - 资源：静态 / 动态
    - 线程池
  - 小结
- JDBC
  - Data Server 视角
    - 数据库溯源
    - SQL及其执行
  - Web Server 视角
    - JBDC溯源
    - JDBC使用
    - 连接池
- 总结
- 拓展

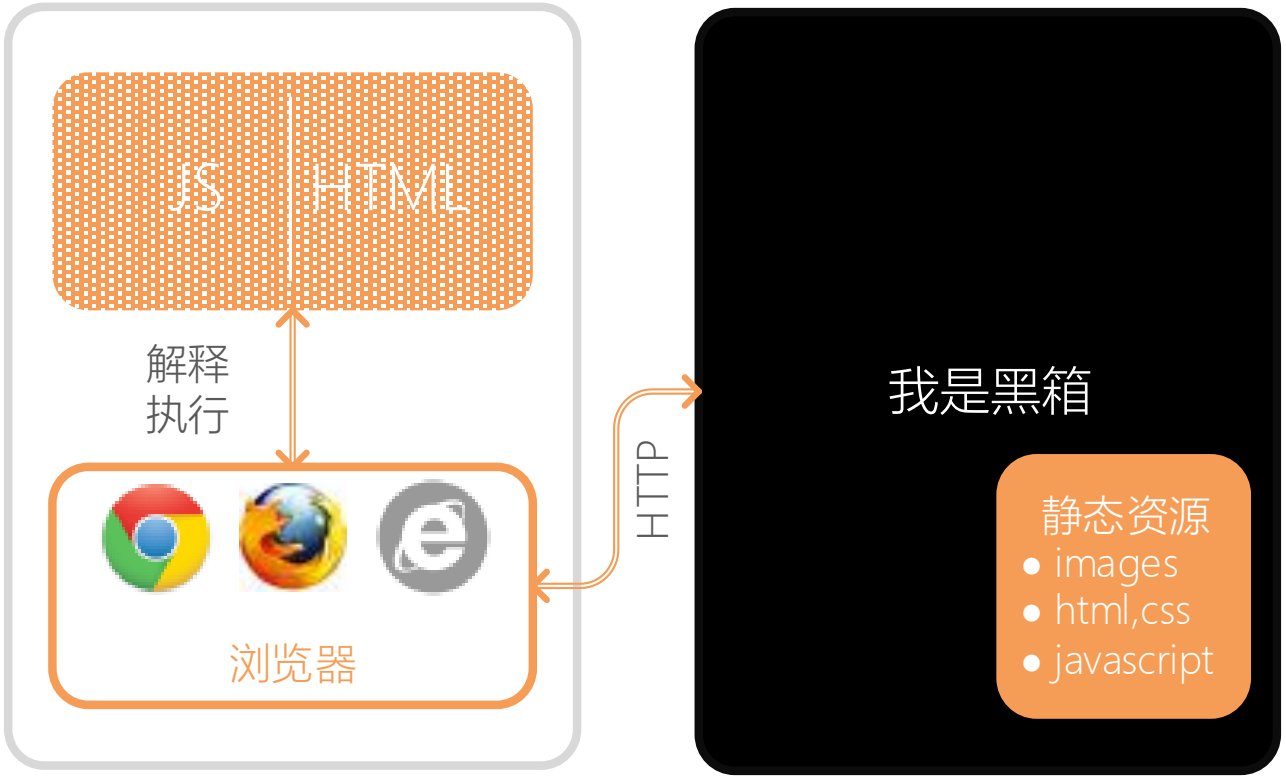
/ HTTP



- **页面 = 渲染规则(模板) + 数据（变化的部分）**
- HTTP 协议 —— WEB & HTTP
  - 应用层协议：通信(TCP) + UI功能
  - 请求-响应：浏览器主动发起请求，服务端被动返回
    - Web Service
  - 无状态：一次请求响应完毕后，服务端不保留任何信息
    - Cookie → Session

方法：post get …
请求类型：同步请求 异步请求
资源类型：动态资源 静态资源
URL

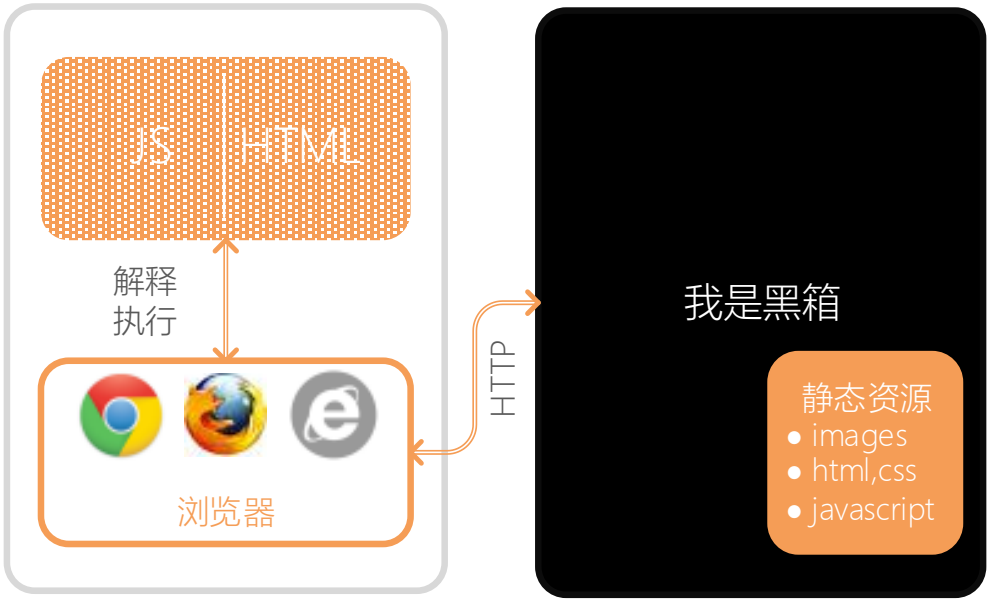
□ / HTTP / 浏览器视角



- 请求类型 ：同步 & 异步

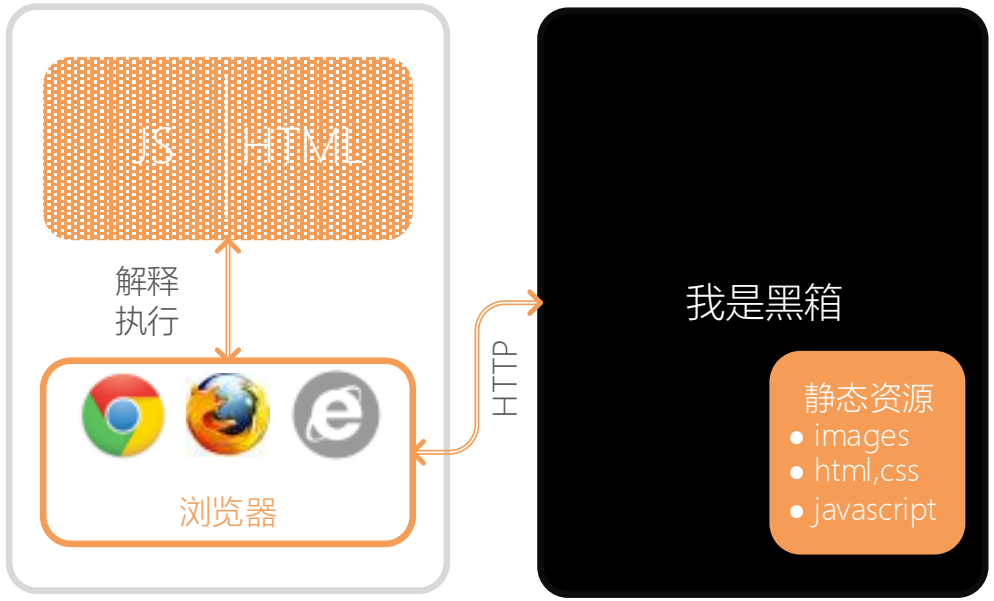
□ / HTTP / 浏览器视角 / 同步请求

- 流程：
  - 发起 | 前进、后退、刷新、提交表单
- **特点**：自动按照服务端返回的内容 刷新 **整个**页面（Frame） **-- 过程不可编程**
- 浏览器：
  - 简单地将返回内容转换到屏幕
  - 投影机的白幕（ shell ）
- 服务端：
  - 提供: 页面完整形式 ← **渲染规则** ← 数据
    - **服务端渲染**
  - 掌控一切 → 投影机的主机
- 原因： 早期的PC机性能有限



□ / HTTP / 浏览器视角 / 异步请求

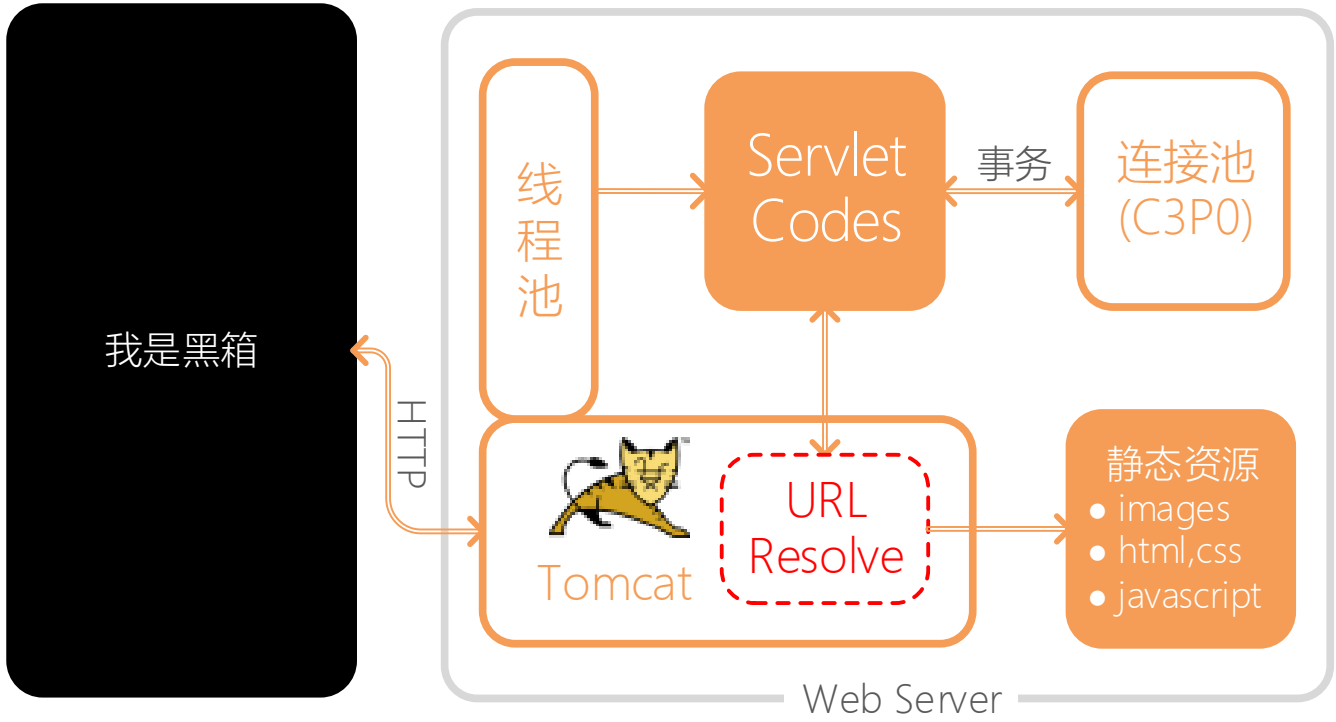
- 流程：
  - 发起：XMLHTTPREQUET协议
  - JS原生
  - Jquery
- 特点：**页面不会被浏览器刷新，过程可编程**
- 浏览器：决定对数据的**渲染规则**
- 服务端： 只输出数据
- 原因
  - PC性能提升，提供可能
  - 要求更好的体验



□ / HTTP / 浏览器视角 / 例子1

- 内容：分别用同步和异步的方式请求同一个URL
- 结论：请求类型--同步 / 异步
  - 浏览器决定
  - 只影响浏览器行为（刷新 or not）
  - 对服务器透明

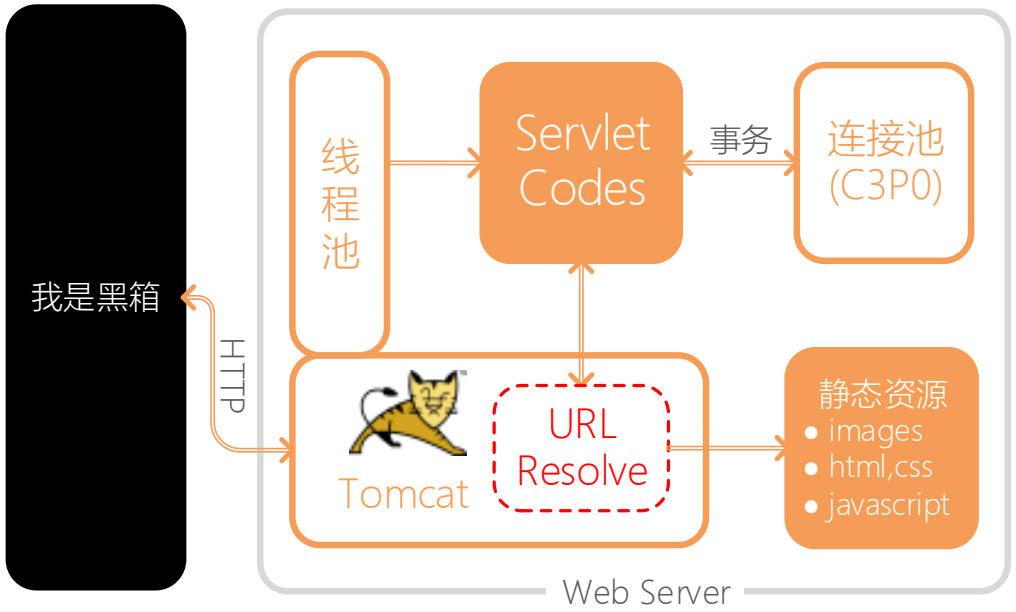
□ / HTTP / Web Server 视角



- 请求**资源**类型：静态资源 & 动态资源请求

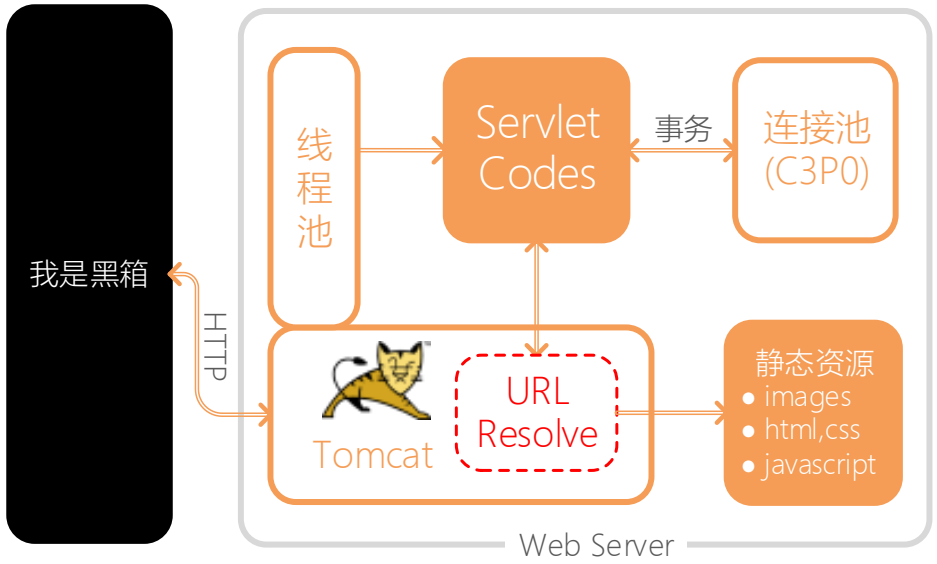
/ HTTP / Web Server 视角 / URL Resolve

- 输入：URL
- 规则
  - Tomcat默认
    - web.xml
    - xxxx.ServletContainerInitialze 接口
  - 框架扩展
    - Struts2 ： Struts2.xml
    - SpringMVC : 注解
- 输出
  - 资源类型：静态 or 动态
  - 下一步的处理程序



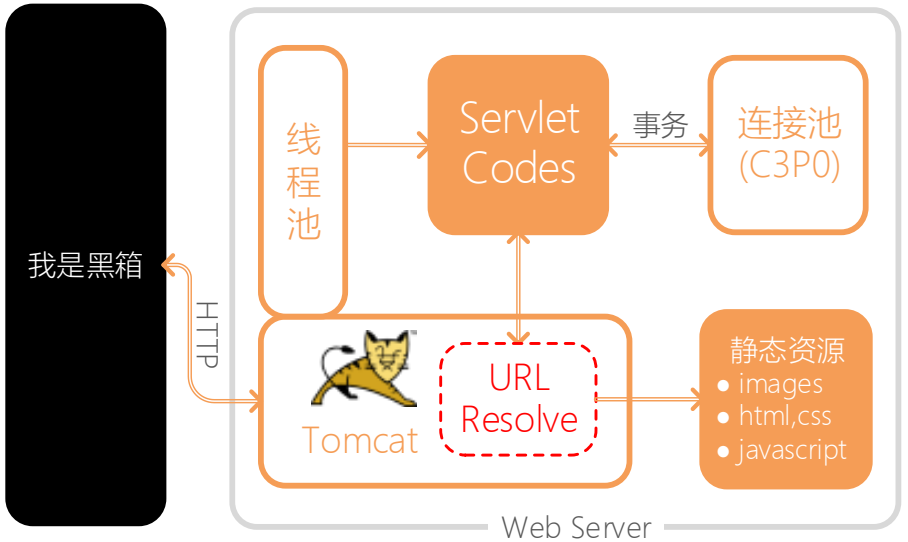
□ / HTTP / Web Server 视角 / 静态资源

- **固定**的处理过程（Server 自动处理）
  - 从文件系统读取文件
  - 发送给浏览器端
  - 具体：磁盘A → **内存A→内存B→**网卡接口→1010… →高低电平→…
  - 过程**不可编程**



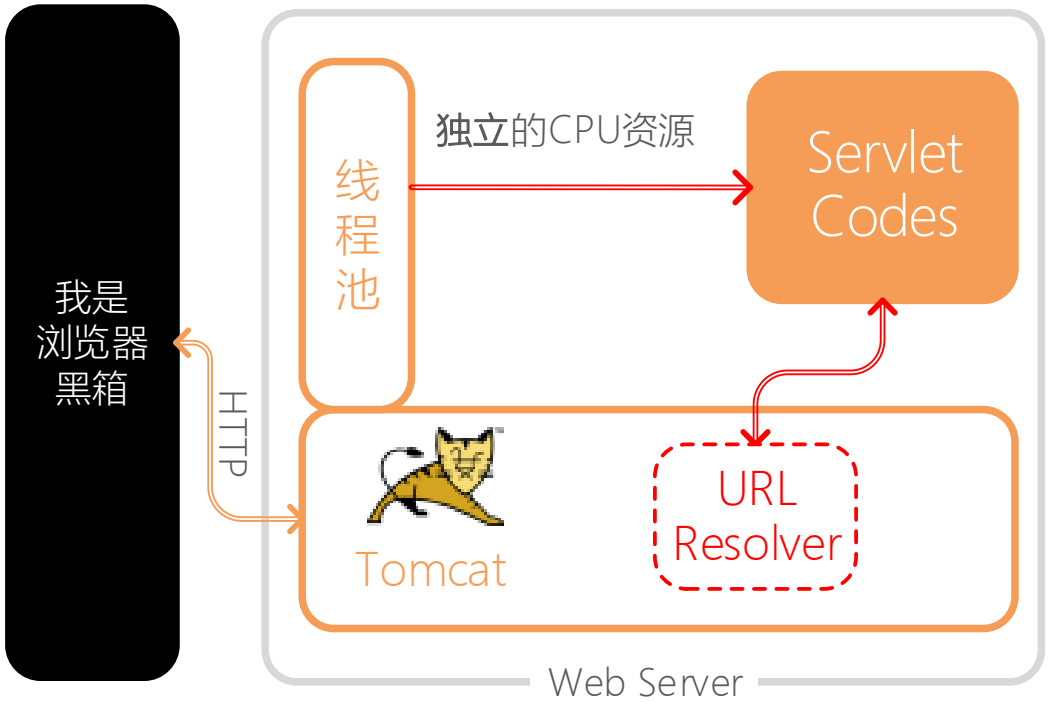
/ HTTP / Web Server 视角 / 动态资源#

- 流程：可编程的处理过程
- 上下文：Servlet规范
  - CGI (Apcaceh C语言)
  - WSGI (python)
  - 你自己的规范（ 实现自己的Web Server ）
- 类型
  - Servlet Code
    - doGet(req, resp)
    - doPost (req, resp)
  - .jsp → Servlet Code
- 会返回什么呢？
  - HTML 字符串（渲染规则 + 数据）
  - JSON 字符串 （数据）



□ / HTTP / Web Server 视角 / 连接池

- 并发处理
  - 多线程/进程
    - prefork(Apache)
    - 线程池(Apache Tomcat)
  - 事件机制 + 回调函数
    - Node
    - libevent
  - 协程
    - Go
    - Scala
    - python



□ / HTTP / Web Server 视角 / 例子2

- 内容：对同一个URL分别按按静态和动态处理
- 结论请求**资源**类型--静态 / 动态
  - Web Server 决定
  - 只影响 Web Server 的行为
    - 响应静态文件
    - 执行 **用户代码**
  - 对浏览器透明

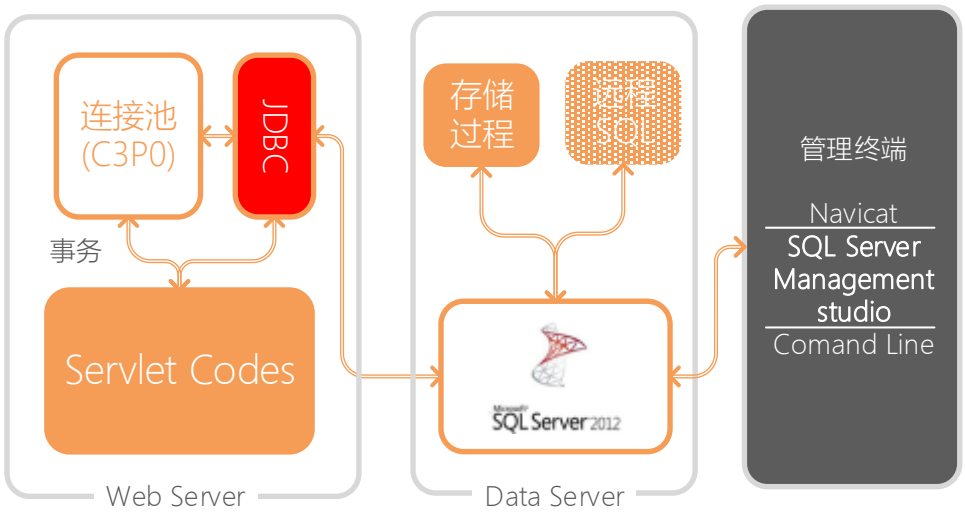
/ HTTP / 小结

B	同步请求	异步请求
网络	URL	
S	静态资源	**动态资源**

- 任意组合
  - 同步 / 异步 对服务端透明
  - 静态 / 动态 对浏览器透明
- 正常情况
  - 同步请求需要后台返回HTML内容（ 数据 + 渲染规则 ）
  - 异步请求需要后台返回JSON (数据)
  - 数据：静态资源没有变化，动态资源会有变化

□ / JDBC

- Data Server视角
  - 溯源
  - SQL及其执行
- Web Server视角
  - JDBC 溯源
  - JDBC 使用

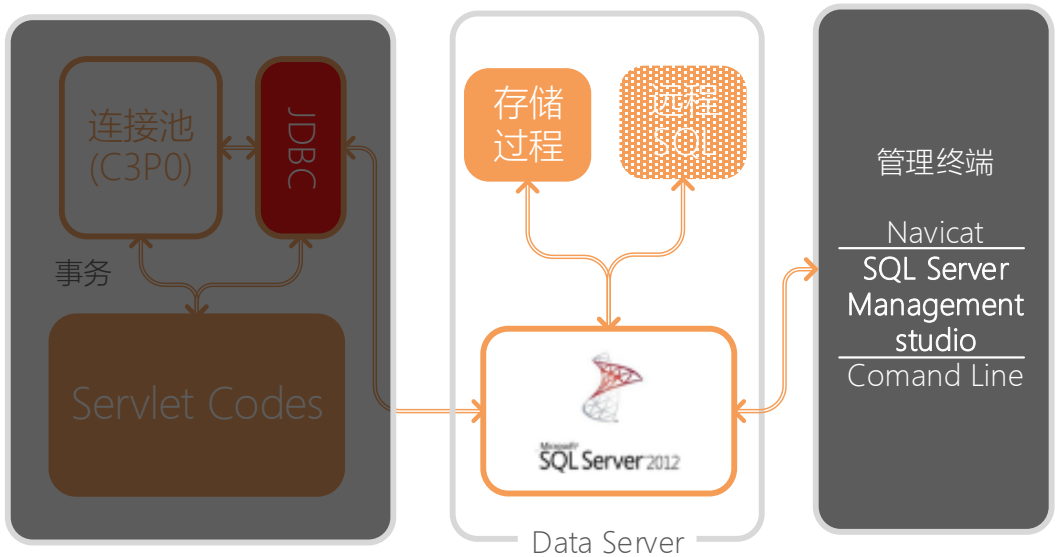


□ / JDBC / Data Server 视角 / 溯源

- 需求
  - **数据持久**化 → 文件系统
  - 数据量变大,方便的CRUD
    → 专职的数据库管理软件
    → **关系数据库**
- 其他数据库
  - 大数据→分布式计算系统
    - 分布式共享内存
    - 内存数据量变大
  - Key/Value数据库：redis、mogodb

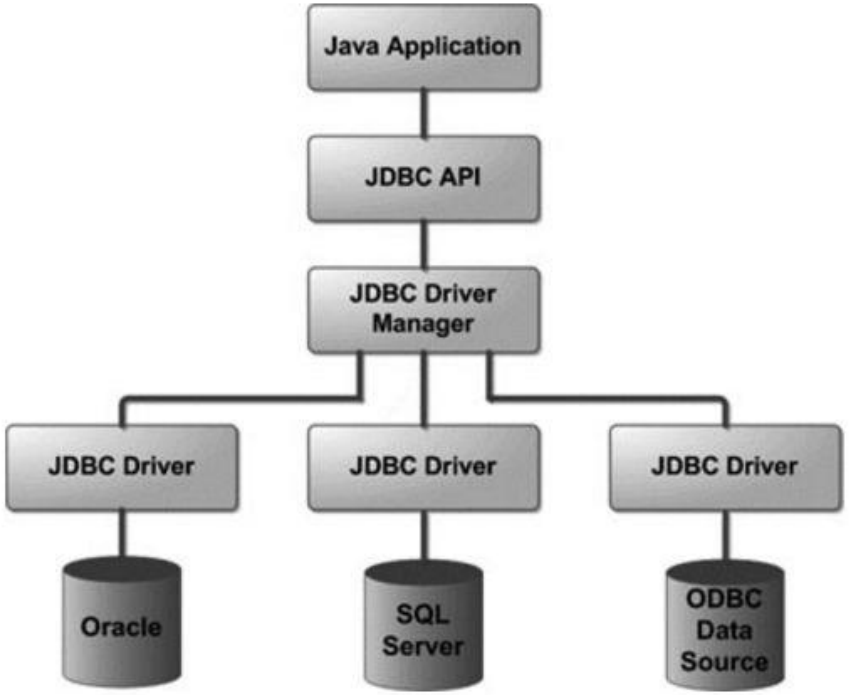
□ / JDBC / Data Server 视角 / SQL及其执行

- SQL 语句规范（语法）
  - 表：主键（唯一），外键（可选集合）
  - 性质：描述性语言（HTML, VHDL）
    - 误区：逐行处理
- 执行方式
  - 区分： 客户端 / 服务端
  - 远程SQL
  - 远程 + 本地SQL
    - 视图
    - **存储过程**



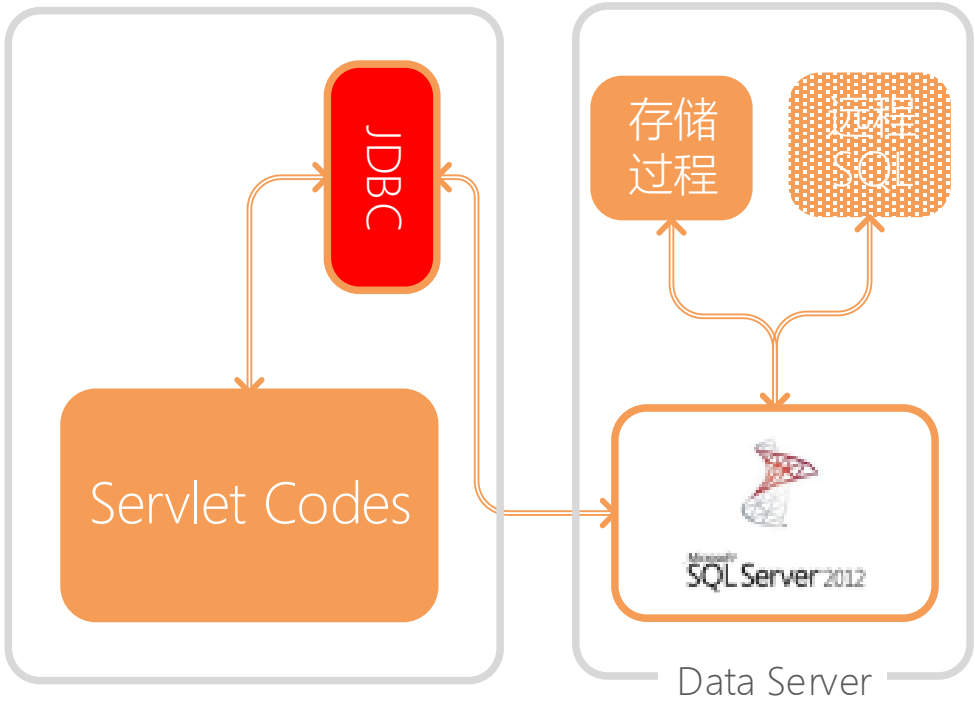
□ / JDBC / Web Server视角 / 溯源#

- 数据库的访问协议
  - 各家不同
  - 如何为JAVA提供一套统一API？
    - JDBC
    - ODBC
- 面向接口编程
  - 用户调用→统一接口→ JDBC API
  - 各家自己**差异**实现驱动
    → JDBC Driver



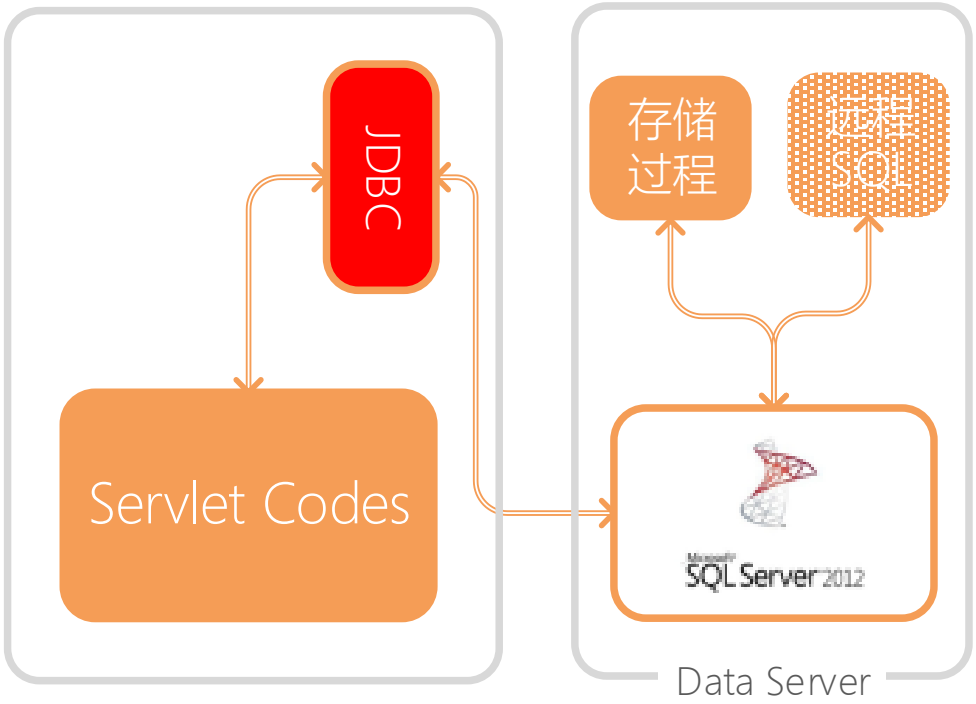
□ / JDBC / Web Server视角 / JDBC 使用#

- 加载驱动
  - 反射
- 流程
  - 获取连接
  - 构造SQL语句
    - 纯拼接
    - 预编译
  - 执行SQL语句
  - **关闭连接**
- 事务
  - 关闭自动提交
  - 期间有没有网络传输？
    - 批量提交



□ / JDBC / Web Server视角 / 连接池

- 加载驱动
  - 反射
- 流程
  - 获取连接
  - 构造SQL语句
    - 纯拼接
    - 预编译
  - 执行SQL语句
  - **关闭连接**
- 事务
  - 关闭自动提交
  - 期间有没有网络传输？
    - 批量提交



□ / 总结

- 同一个URL
  - 请求类型由浏览器决定，也只影响浏览器的行为
  - 请求资源类型由服务器决定，也只影响服务器的行为
- 同步还是异步
  - 混合 → 全异步
  - 功能需求 工程化需求
- 注意区分数据库的客户端和服务端
- 软件工程演化规律：变与不变（Servlet, JDBC）
  - 可以固定下来做为基础设施
  - 为需要变的地方留下接口：规范（文档和实现）
  - 框架 or 容器 是规范的实现

□ / 拓展1 / 一种理解模型

```
Int main（）{
    初始化过程；
     while（1）{
            服务阶段；
     }
     销毁阶段；
}
```

- **初始化阶段（设置全局上下文）**
- 服务阶段
  - **处理输入请求，设置请求上下文**
  - **执行用户代码**
- 销毁阶段

□ / 拓展2 / 更复杂的情况

- 框架
  - Struts2, SpringMVC
  - Hibernate, Mybaties
- 分布式
  - **层次：**
    - **业务逻辑分布**
      - 计算分布
      - 存储分布
  - 思想：SOA（Service-Oriented Architecture）
    - 服务分布部署，独立演进
    - 服务自动发现与治理
  - 实现:
    - 对内：Rpc
      - RMI
      - Hession
    - 对外：Web Service
      - SOAP: HTTP之上 、WSDL
      - **RESTful HTTP本身 、URL（更严格） + JSON**