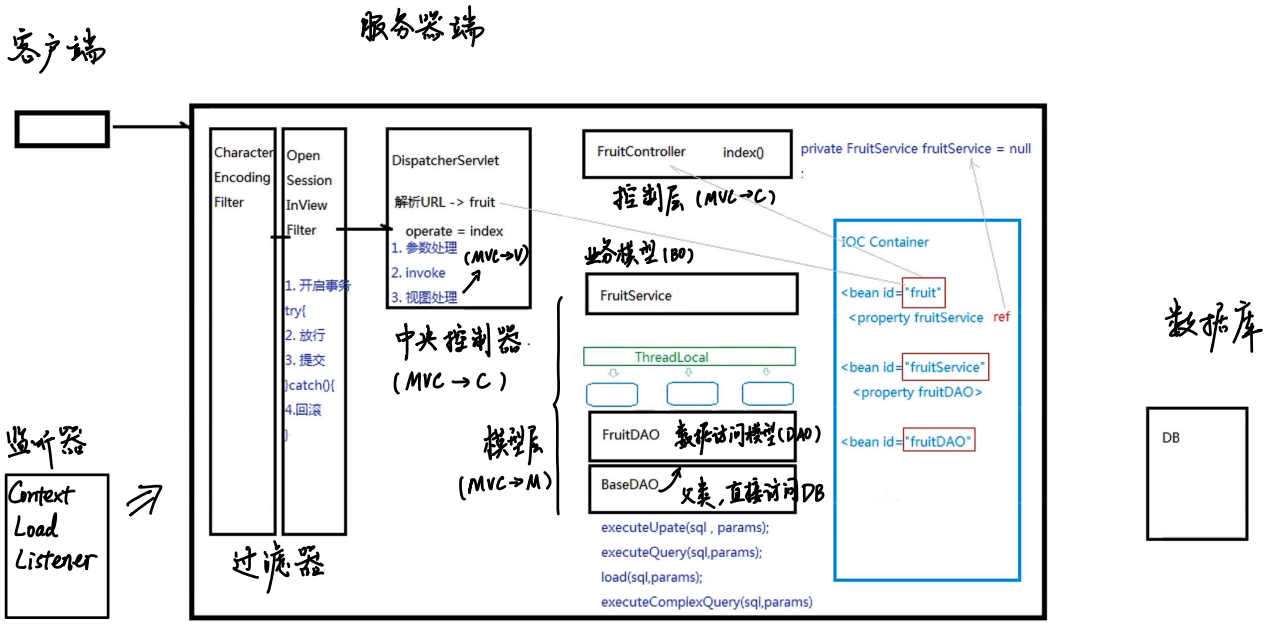

流程：

1. 客户端向服务器端发请求 URL = "localhost:8080/fruit.do".
2. 两个过滤器拦截请求，处理后放行
3. 中央控制器 DispatcherServlet 收到请求，解析 URL，得到 path 为 "fruit"，opercite = "index".
   通过 IOC 容器得知 path "fruit" 对应的组件为 FruitController 控制器，并通过反射（invoke 方法）调用控制器中 index 方法.
4. 在 FruitController 的 index() 方法内部，会调用 FruitService 内部的方法
5. FruitService 调用 FruitDAO 方法，FruitDAO 的父类 BaseDAO 会去访问数据库 DB

各组件作用：

1. CharacterEncodingFilter：收到请求后先拦截，将 servlet request 编码设置后（如"UTF-8"）再放行

2. OpenSessionInViewFilter：收到请求后拦截，为了保证事务原子性，设置如下代码：

   ① 阻止事务自动提交

   try { ② 放行 ③ 提交 } catch { ④ 回滚 }.

   保证了事务提交以 Service 中方法为整体而非单个 DAO 方法（即一个业务方法成功意味着该业务所有 DAO 都成功，失败则所有 DAO 都失败）

3. DispatcherServlet：拦截所有"*.do"的请求，并做三件事：

   ① 参数处理

   ② invoke 调用对应 Controller 中方法

   ③ 视图处理

4. IOC 容器：作用：控制反转 + 依赖注入，做两件事：

   ① 保存 id 和对应实例对象的映射关系（如给出 id="fruit"，能映射到 FruitController 实例）

   ② 为不同层之间注入依赖 { property + ref：说明依赖关系
   通过反射方法注入，ref 注入到 property 这一属性中
   （A.field = A 依赖的实例 B）

5. FruitController：水果模块的控制器（属于 MVC 中的 C），调用各种业务方法的组合。

6. FruitService：业务层（属于 MVC 中的 M），实现各种业务方法（如登录，购物等）
   每个方法是多个 DAO 方法的组合。

   注：多个 DAO 间信息如何传递（保证事务原子性，让多个 DAO 知道是属于同一个业务）？
   通过 ThreadLocal：保证同个线程的连接是性一的，如果多个 DAO 想共享1个连接（connection），只要和线程相关，就是共享的。

7. FruitDAO：数据访问层（属于MVC中的M），访问数据库，实现增删查改等

8. ContextLoadLister：监听器，作用是在整个application启动时就创建IOC容器，使得后续访问速度提升