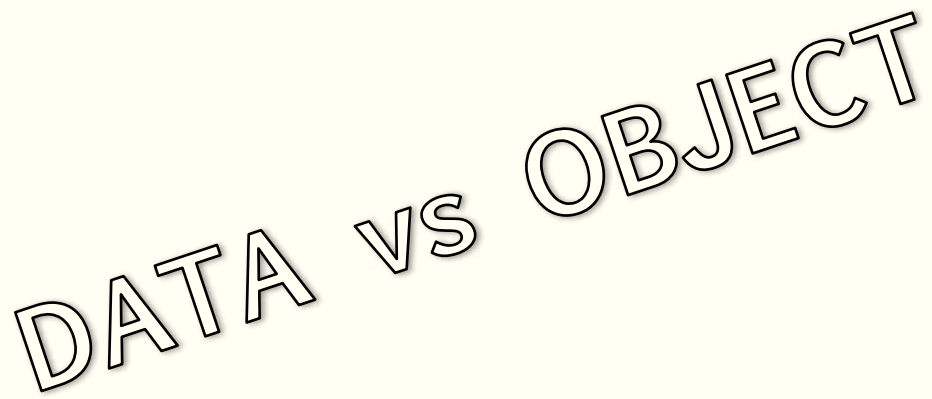
领域驱动盒马实践

领域模型：基于数据库vs基于对象

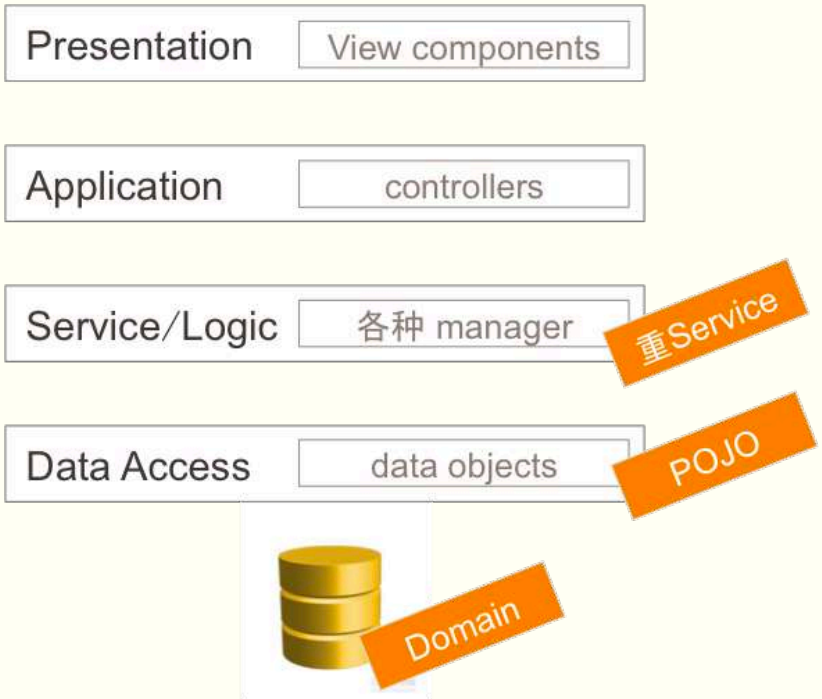
- Data Modeling:通过数据抽象系统关系，也就是数据库设计
- Object Modeling:通过面向对象方式抽象系统关系，也就是面向对象设计



领域模型：Data Modeling

- 数据字典就是领域模型
- 外键就是关系
- Manager组织逻辑
- 数据库设计>代码设计

静

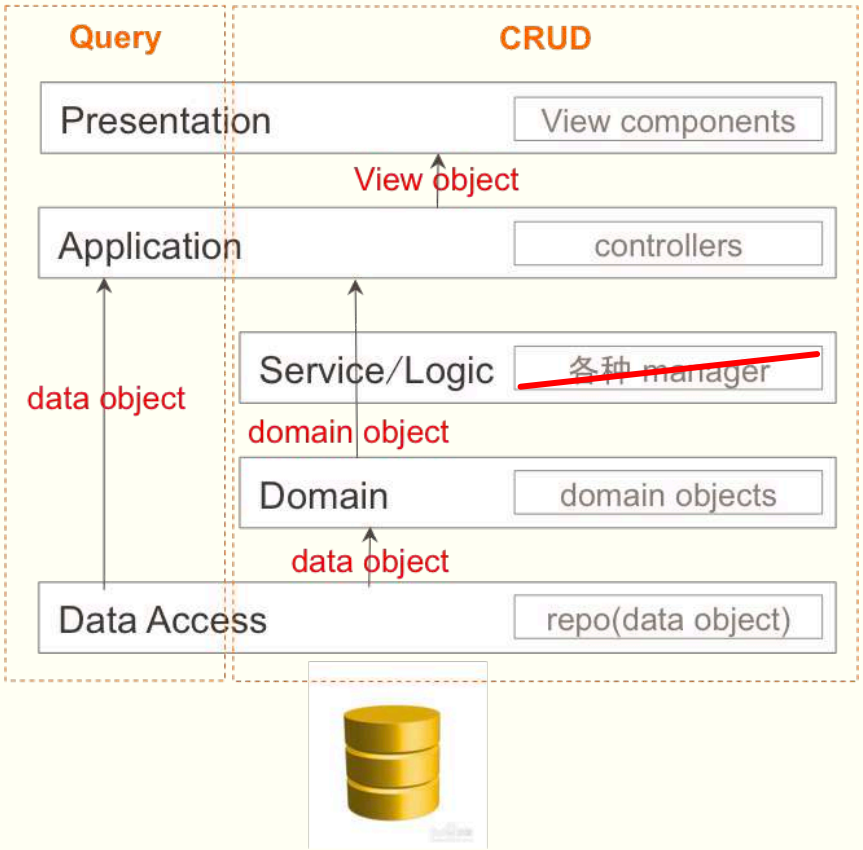


领域模型：Object Modeling

- 假设：内存无限大，永远不宕机
- 持久化无关设计： Persistence Ignorance
- 对象模型才是领域模型

动

领域模型：Object Modeling



- 现实：内存有限，总有宕机和重启
- 数据库：持久化数据（CRUD）
- 重点放在如何有效查询(QUERY)

DATA>QUERY

OBJECT>DOMAIN

领域模型：失血，贫血和充血模型

- 失血模型：基于数据库的领域设计方式其实就是典型的失血模型(POJO DATA)
- 贫血模型：盒马流程中心(DATA+METHOD)
- 充血模型：盒马基础资料（DATA+METHOD+REPO）

领域模型：失血

```
public class Father{…}
public class Son{
    private String fatherId;//son表里有fatherId作为Father表id外键
    public String getFatherId(){
        return fatherId;
    }
    ……
}
```

领域模型：贫血

```
public class Son{
    private Father father;
    public Father getFather(){return this.father;}
}
```

```
public class Father{
    private Son son;
    private Son getSon(){return this.son;}
}
```

领域模型：充血

```
public class Son{
    private Father father;
    public Father getFather(){return this.father;}
}
```

```
public class Father{
    //private Son son; 删除这个引用
    private SonRepository sonRepo;//添加一个Son的repo
    private getSon(){return sonRepo.getByFatherId(this.id);}
}
```

领域模型：依赖注入

- 依赖注入在runtime是一个singleton对象，只有在spring扫描范围内的对象（@Component）才能通过annotation（@Autowired）用上依赖注入，通过new出来的对象是无法通过annotation得到注入的
- 个人推荐构造器依赖注入，这种情况下测试友好，对象构造完整性好，显式的告诉你必须mock/stub哪个对象

领域模型：依赖注入

```
public class Father{
    private SonRepository sonRepo;
    private Son getSon(){return sonRepo.getByFatherId(this.id);}
    public Father(SonRepository sonRepo){this.sonRepo = sonRepo;}
}
```

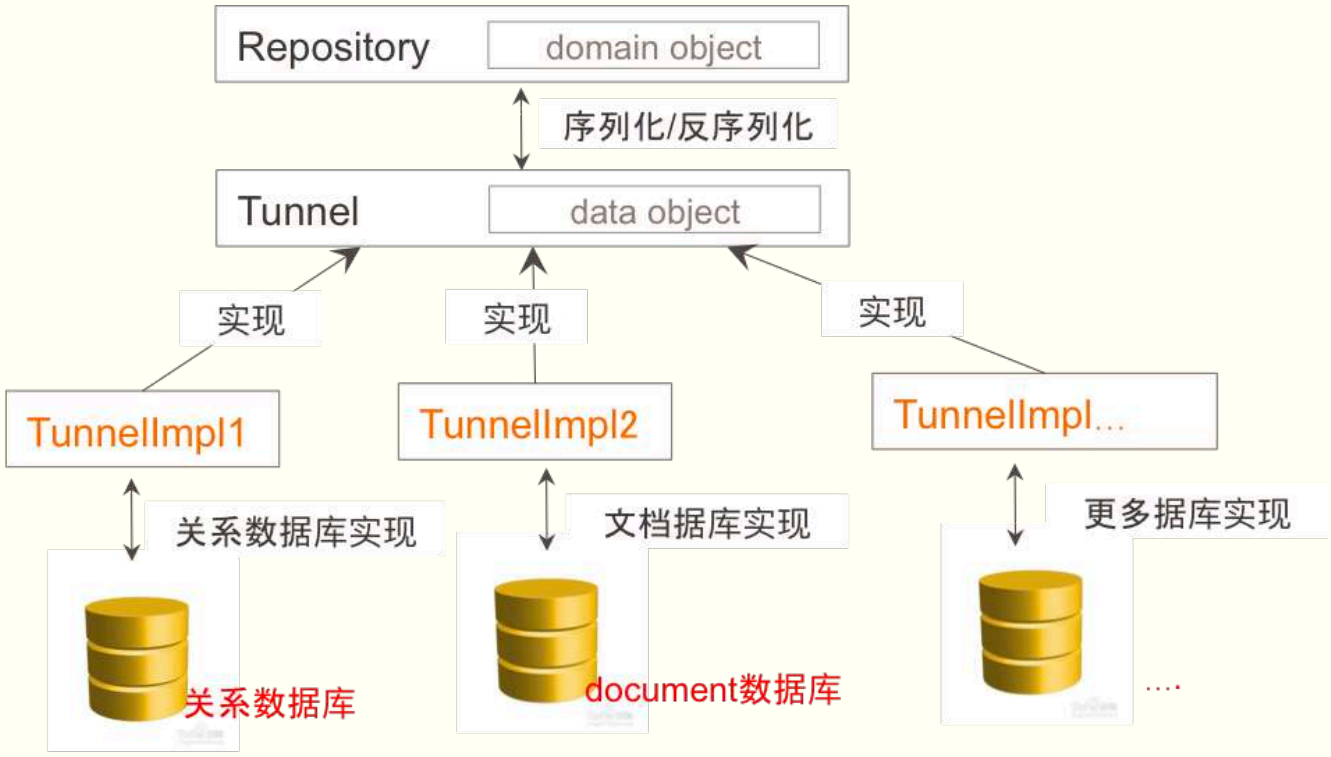
```
@Component
public class FatherFactory{
    private SonRepository sonRepo;
    @Autowired
    public FatherFactory(SonRepository sonRepo){}
    public Father createFather(){
        return new Father(sonRepo);
    }
}
```

领域模型：测试友好

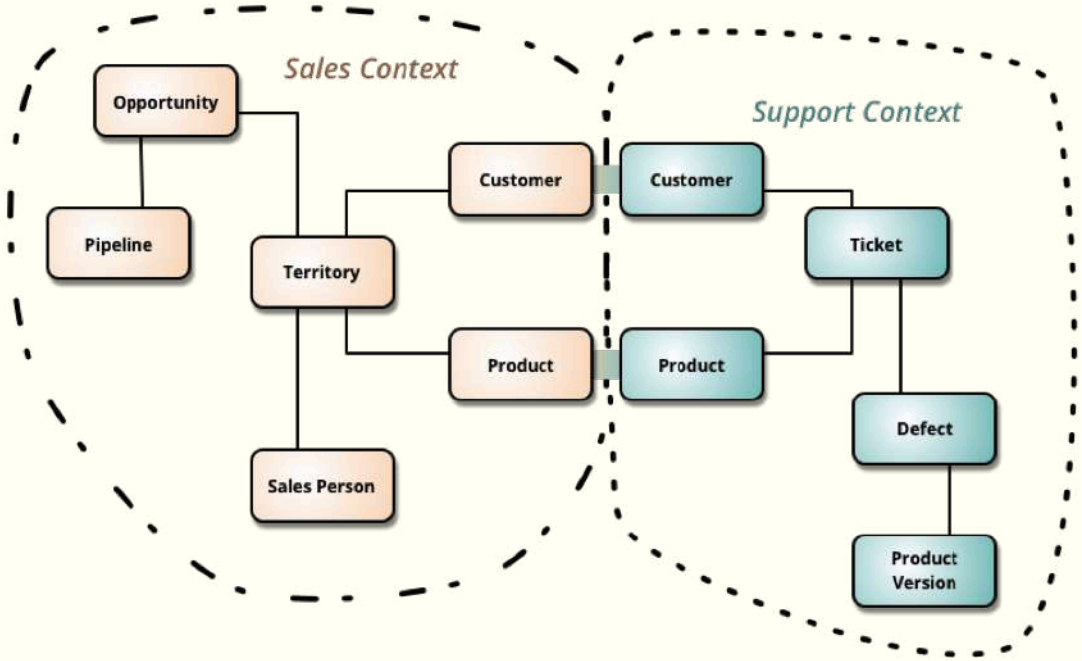
失血模型和贫血模型是天然测试友好的

```
public class Father{
    private SonRepository sonRepo;//=new SonRepository()这里不能构造
    private getSon(){return sonRepo.getByFatherId(this.id);}
    //放到构造函数里
    public Father(SonRepository sonRepo){this.sonRepo = sonRepo;}
}
```

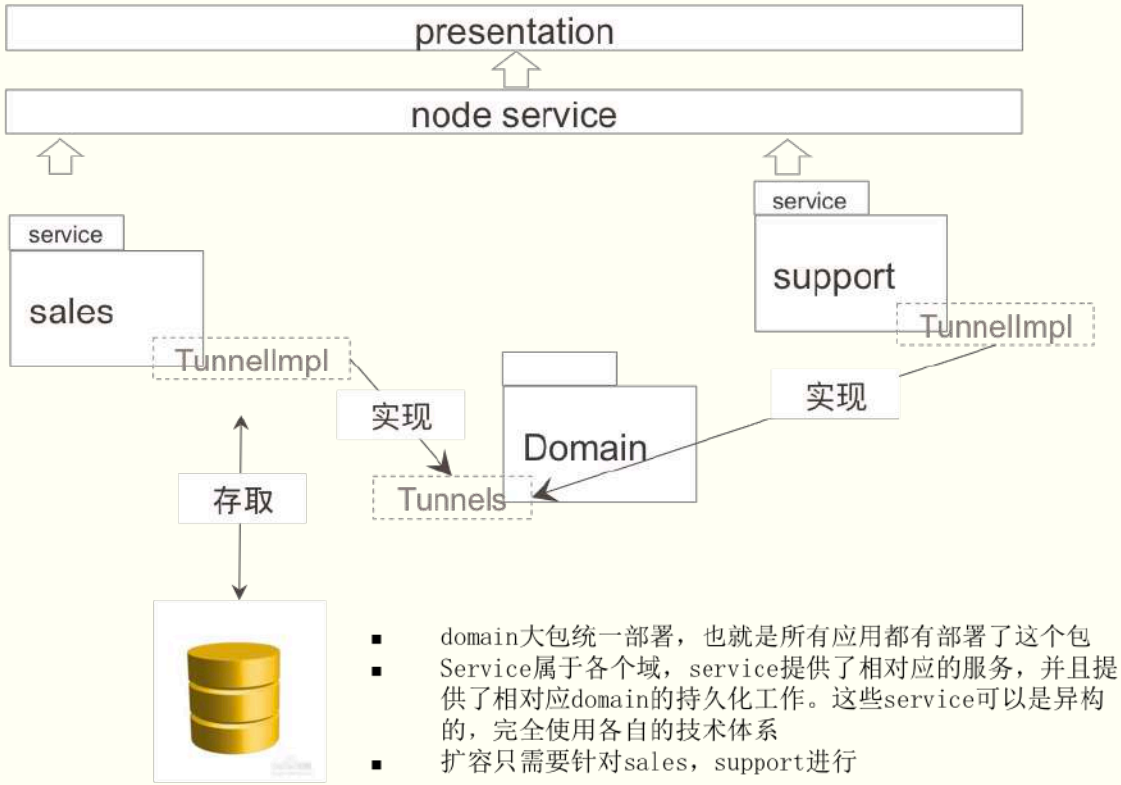
领域模型：盒马模式下的repo实现



领域模型：盒马模式下的部署结构



领域模型：盒马模式下的部署结构



- domain大包统一部署，也就是所有应用都有部署了这个包
- Service属于各个域，service提供了相对应的服务，并且提供了相对应domain的持久化工作。这些service可以是异构的，完全使用各自的技术体系
- 扩容只需要针对sales，support进行