



MongoDB在跨境电商物流供应链 系统中的实践

黄亮

- 关于我
- 一个跨境电商圈真实故事
- 关于出口易物流
- 出口易新老架构演变过程
- 基于MongoDB的持久化实现
- Q&A



大家好！ 我是黄亮

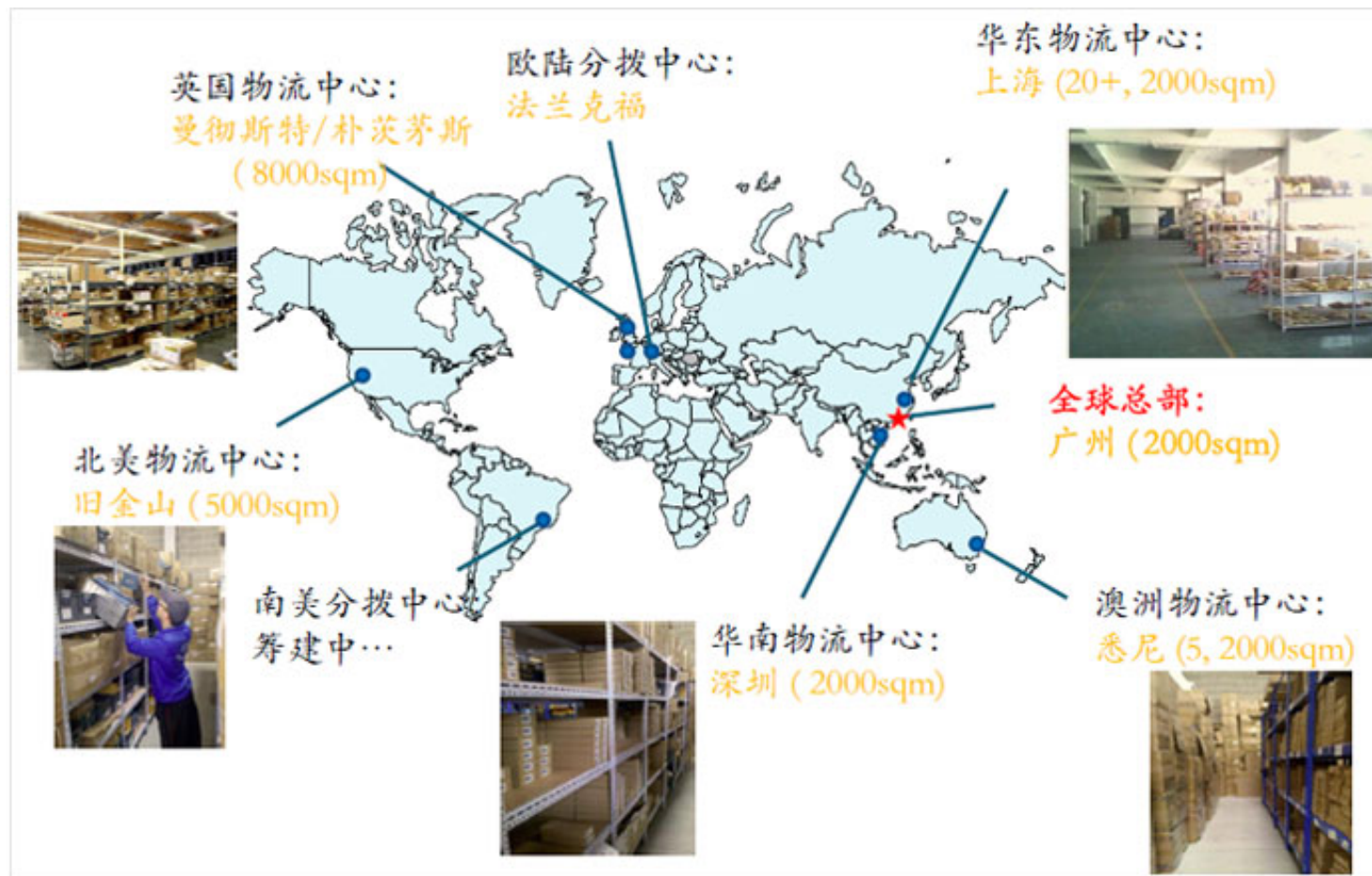
- ❖ 2011.8~2014.11 就职于中国跨境电商第一家香港上市企业DX.COM。历任高级研发工程师、研发经理、项目经理等职。
- ❖ 2014年12月~至今 就职于广州市贝法易集团。历任高级研发经理、技术总监等职。主要负责集团旗下出口易物流和M2C业务技术团队研发和管理工作。



一个跨境电商圈的真实故事

出口易物流是广州市贝法易商贸有限公司（简称贝法易）旗下，以全球仓储为核心，整合全球物流网络系统，为跨境电商卖家提供海外仓储、国际专线、国际小包、国际快递、FBA头程等物流服务以及本地化售前售后服务，解决订单管理、金融融资等难题，打造最值得信赖的跨境电商全程物流解决方案提供商。

覆盖欧美澳主要市场的服务网络



全球主流电商平台重点推荐物流服务提供商

amazon

ebay

wish

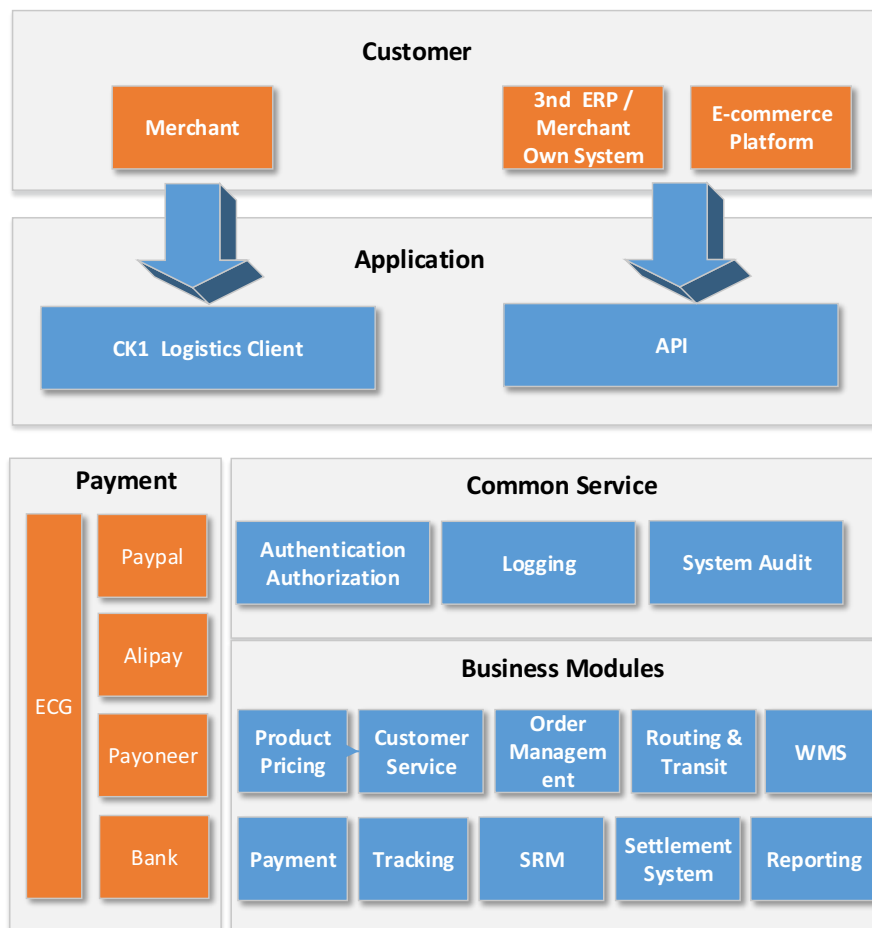
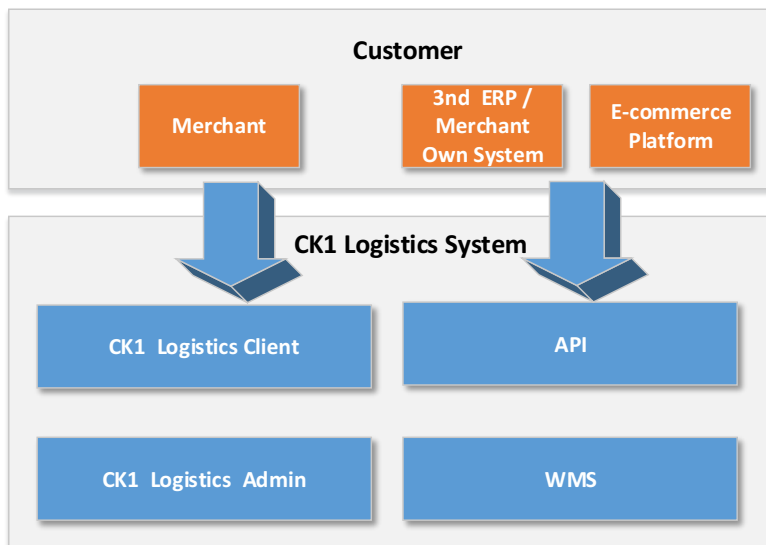
Alibaba.com®
Global trade starts here.™

Shopee

AliExpress™
Smarter Shopping, Better Living!

LAZADA
•COM•MY

出口易新老架构演变过程



出口易老业务系统特点

单体应用

前后端系统共用一套 WEB App Solution (ASP.NET + WEB Service)

单一数据库

采用MS SQLServer 数据库，核心业务功能共用一个数据库。

业务功能完整

IT系统随业务的发展不断扩展新功能。满足开展跨境电商物流业务最基本的功能性需求。

容易测试&部署

单独一个Solution，系统依赖少，一旦部署，全部功能即可测试。

不够灵活

对应用程序做任何细微的修改都需要将整个应用程序重新构建、重新部署

妨碍持续交付

系统规模大，构建和部署时间也相应地比较长，不利于频繁部署，阻碍持续交付

受技术栈限制

包括开发语言，开发工具，数据库一旦选定，无法根据实际需要作其他选择。

技术负债

系统逻辑异常复杂，随着时间推移，人员更迭，技术负债不断累积。

面向服务

根据业务模块切分不同的系统模块，系统模块采用面向服务架构。服务与服务通过明确的接口定义进行通讯

领域驱动设计

每个业务模块团队负责一个领域或业务功能相关的全部开发。核心领域根据DDD中明确定义的规则实现。

独立部署、升级、扩展和替换

每个服务可以单独部署，透明升级，不影响整个系统。

异构/采用多种语言

每个服务开发团队，可以选择自己熟悉开发语言，数据库，开发工具和开发架构。

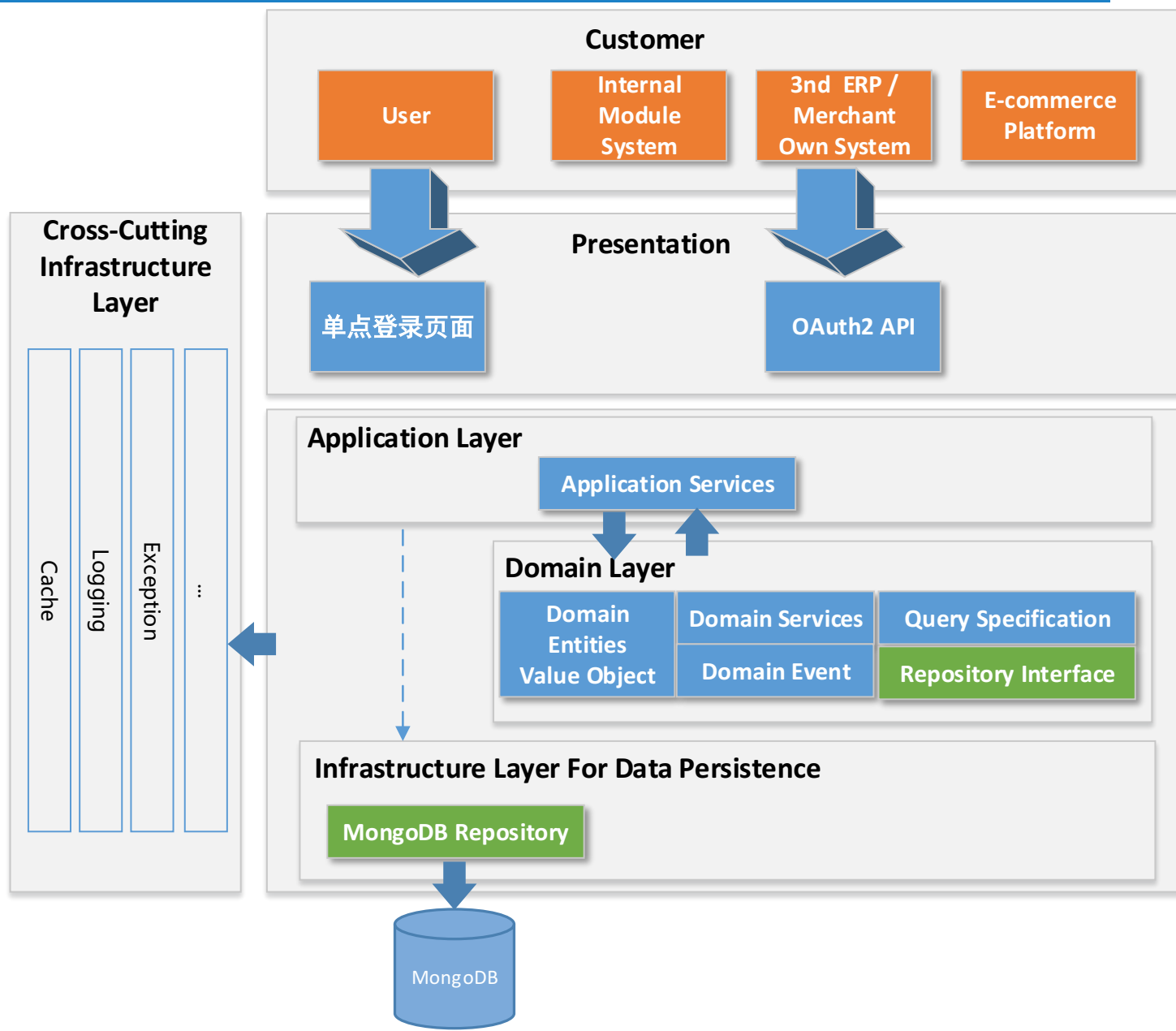
每个服务都需要统一的
登录认证。

身份认证
Authentication

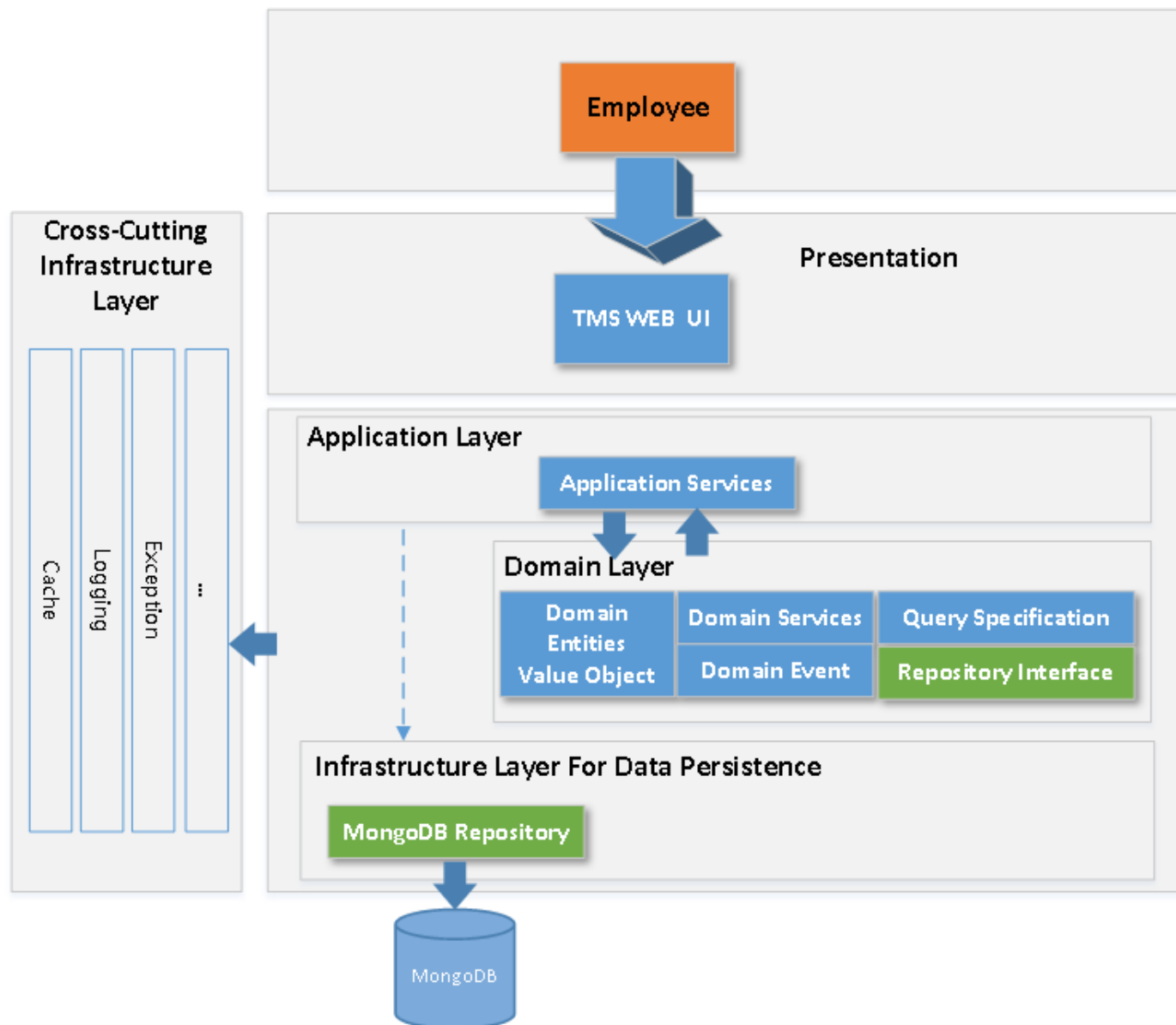
鉴权
Authorization

不同的用户使用相同的
服务模块都需要鉴权

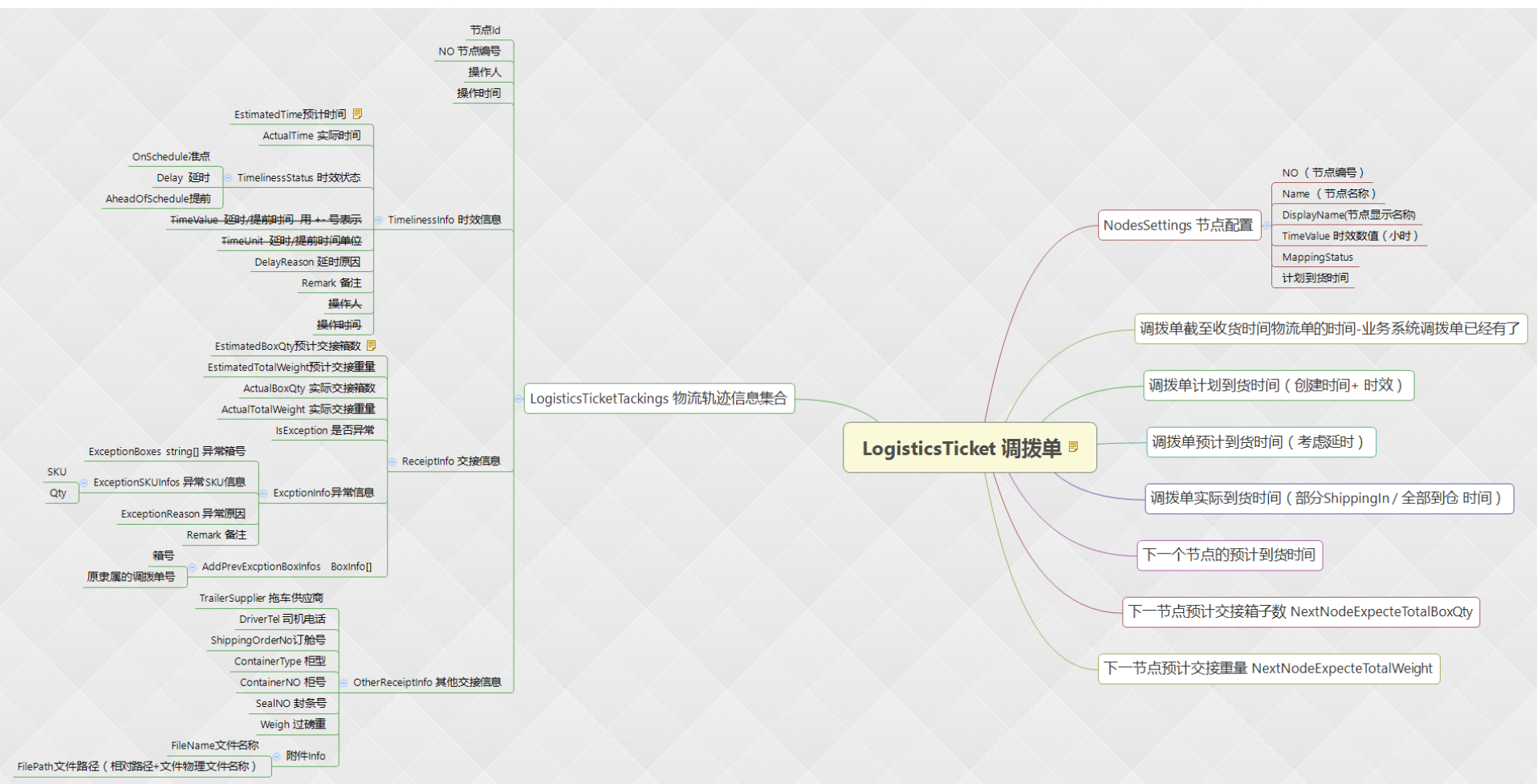
新架构下第一个服务模块Portal



第二个服务模块TMS系统



TMS系统调拨单聚合根示意图



为什么选择MongoDB ?



1

非事务紧密型。错误数据容忍性相对比较高。

2

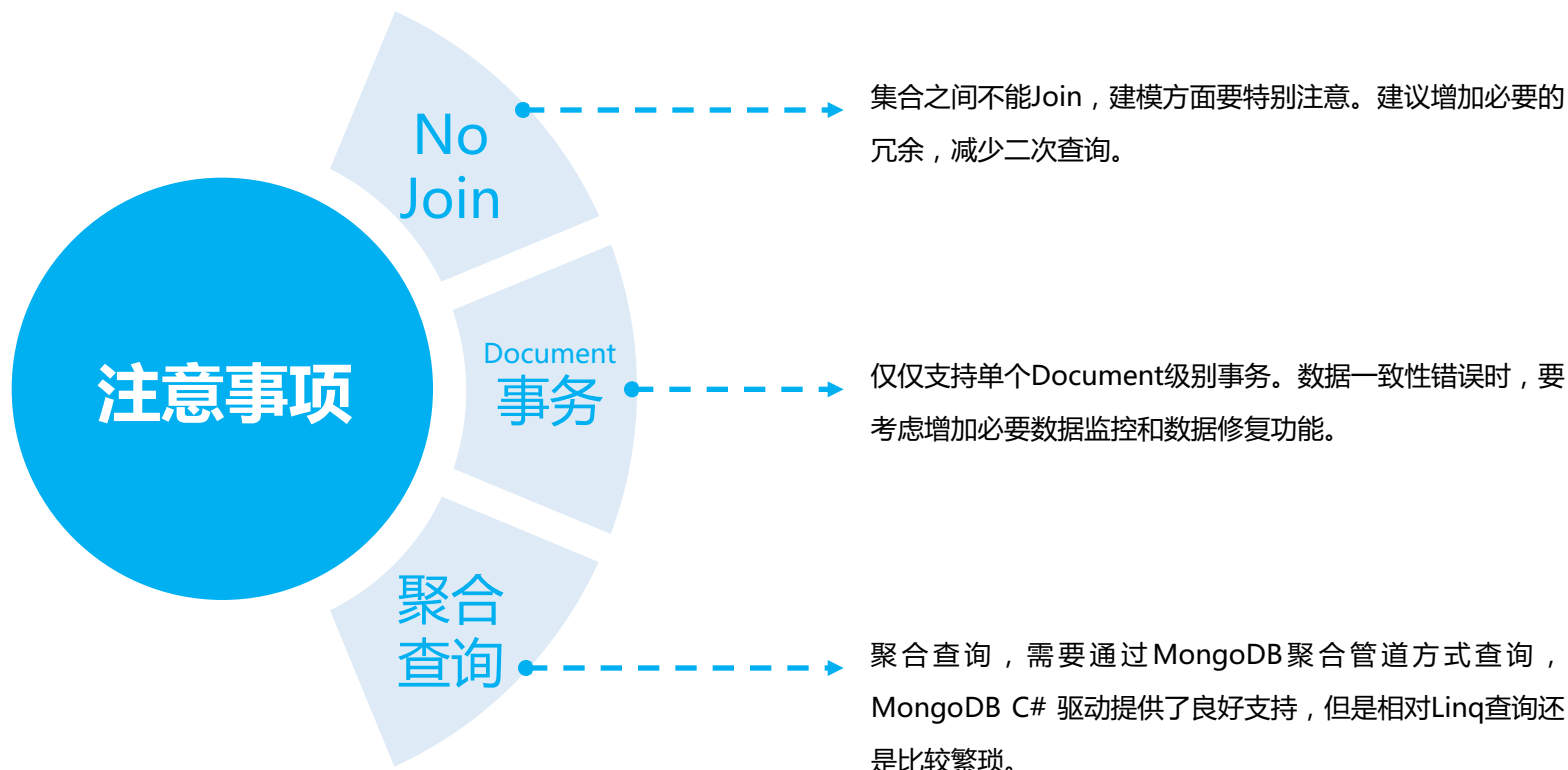
团队成员有使用MongoDB开发经验。对基于MongoDB方面的建模需要考虑的必要冗余有一定的了解。

3

Portal 模块数据库读大于写，基于MongoDB读写方面的高性能，解决了高并发下系统卡顿问题。

4

TMS 系统模型之间关系复杂，采用传统关系数据库，势必增加一堆表。采用MongoDB，可以把复杂的模型，通过一个Document存储到一起。





一、仓储 Repository

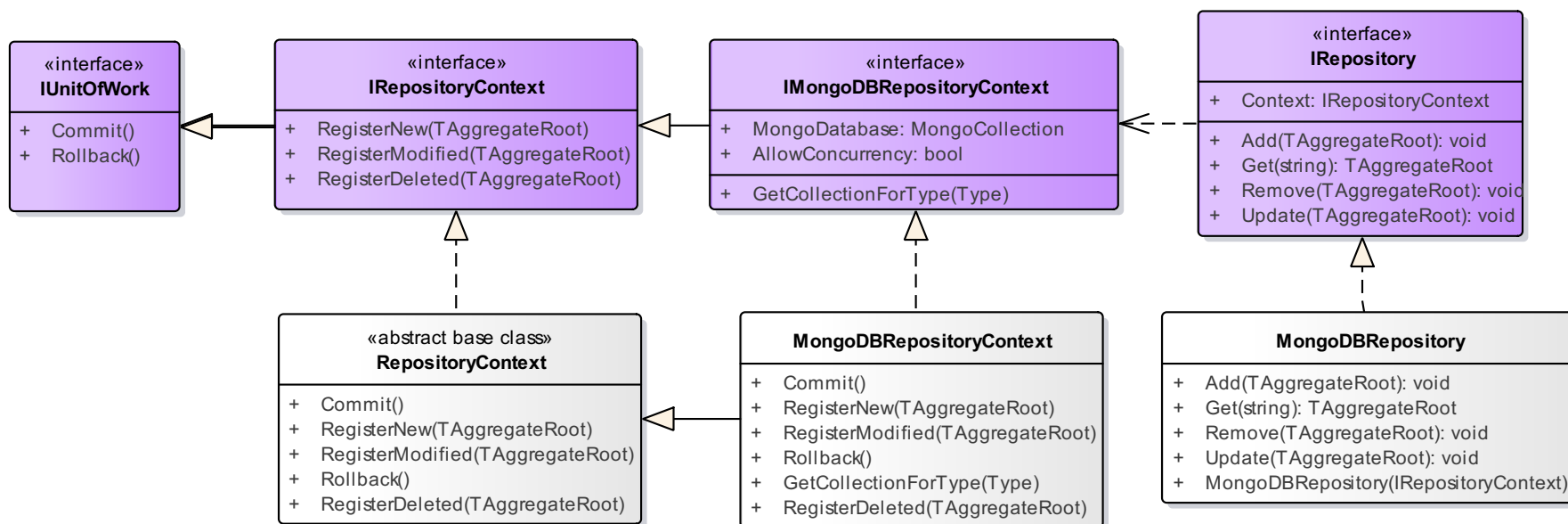
- 仓储限定在对整个聚合根的操作上
- 提供聚合根的持久化和重建（查询）

二、仓储上下文 Repository Context

- 负责事务处理。每个聚合根的仓储都会关联到同一个仓库上下文。（但是 MongoDB 不支持事务，我们提供了虚拟实现）
- 仓储上下文应用了工作单元模式（Unit Of Work 模式）

仓储及其上下文类图

class Class Model



一、领域模型采用POCO (POJO)

- 简单的CLR对象 (简单的Java对象)
- 不继承任何持久化框架中的基类，或实现任何持久化框架中的接口。
- 领域层不引用MongoDB类库。
- MongoDB仓库层使用lambda expression 实现类的Map

二、ID 生成器

- 有多种ID生成器可供选择。GuidIdGenerator, ObjectIdGenerator,String ObjectIdGenerator, etc
- 我们ID一律使用String类型。所以直接使用MongoDB的 StringObjectIdGenerator

三、多态类的Map

- 如果把多态类（继承）映射到MongoDB，需要指定已知类型

```
BsonClassMap.RegisterClassMap<AirLogisticsTicket>();  
BsonClassMap.RegisterClassMap<FirstExpressLogisticsTicket>();  
BsonClassMap.RegisterClassMap<InlandLogisticsTicket>();  
BsonClassMap.RegisterClassMap<OceanLogisticsTicket>();  
BsonClassMap.RegisterClassMap<OneSelfDeliverFirstLogisticsTicket>();  
BsonClassMap.RegisterClassMap<SpecialLogisticsTicket>();|
```

四、一些需要了解的约定（Convention）

- NamedIdMemberConvention 可以指定类的哪些属性可以作为ID
- IgnoreExtraElementsConvention 可以忽略Document中不存在于类中的字段。否则会抛出异常
- EnumRepresentationConvention 可以指定枚举序列化的方式，我们都指定为BsonType.String

一、聚合框架 (Aggregation Framework)

- MongoDB 2.2版本引入了此功能
- 是数据聚合的一个新框架。
- 一是对文档进行“过滤”，也就是筛选出符合条件的文档;
二是对文档进行“变换”，也就是改变文档的输出形式。
其他的也包括按照某个指定字段分组和排序等。
- 是MapReduce的替代方案，比MapReduce简单
- 该框架使用声明性管道符号来支持类似SQL 中的 Group by 操作的功能。不需要自己编写自定义的JavaScript。

二、管道操作符

Name	说明
\$project	数据投影，主要用于重命名、增加和删除字段
\$match	过滤操作，筛选符合条件文档，作为下一阶段的输入
\$limit	限制经过管道的文档数量
\$skip	从待操作集合开始的位置跳过文档的数目
\$unwind	将数组元素拆分为独立字段
\$group	对数据进行分组
\$sort	对文档按照指定字段排序
\$geoNear	会返回一些坐标值，这些值以按照距离指定点距离由近到远进行排序。这个在地理信息系统中比较常用。

三、管道表达式：组聚合操作符

<u>\$addToSet</u>	Returns an array of all the unique values for the selected field among for each document in that group.
<u>\$first</u>	Returns the first value in a group.
<u>\$last</u>	Returns the last value in a group.
<u>\$max</u>	Returns the highest value in a group.
<u>\$min</u>	Returns the lowest value in a group.
<u>\$avg</u>	Returns an average of all the values in a group.
<u>\$push</u>	Returns an array of all values for the selected field among for each document in that group.
<u>\$sum</u>	Returns the sum of all the values in a group.

Bool类型聚合操作符

<u>\$and</u>	Returns true only when all values in its input array are true.
<u>\$or</u>	Returns true when any value in its input array are true.
<u>\$not</u>	Returns the boolean value that is the opposite of the input value.

比较类型聚合操作符

<u>\$cmp</u>	Compares two values and returns the result of the comparison as an integer.
<u>\$eq</u>	Takes two values and returns true if the values are equivalent.
<u>\$gt</u>	Takes two values and returns true if the first is larger than the second.
<u>\$gte</u>	Takes two values and returns true if the first is larger than or equal to the second.
<u>\$lt</u>	Takes two values and returns true if the second value is larger than the first.
<u>\$lte</u>	Takes two values and returns true if the second value is larger than or equal to the first.
<u>\$ne</u>	Takes two values and returns true if the values are not equivalent.

字符串类型聚合操作符

<u>\$concat</u>	Concatenates two strings.
<u>\$strcasecmp</u>	Compares two strings and returns an integer that reflects the comparison.
<u>\$substr</u>	Takes a string and returns portion of that string.
<u>\$toLower</u>	Converts a string to lowercase.
<u>\$toUpper</u>	Converts a string to uppercase.

日期类型聚合操作符

<u>\$dayOfYear</u>	Converts a date to a number between 1 and 366.
<u>\$dayOfMonth</u>	Converts a date to a number between 1 and 31.
<u>\$dayOfWeek</u>	Converts a date to a number between 1 and 7.
<u>\$year</u>	Converts a date to the full year.
<u>\$month</u>	Converts a date into a number between 1 and 12.
<u>\$week</u>	Converts a date into a number between 0 and 53
<u>\$hour</u>	Converts a date into a number between 0 and 23.
<u>\$minute</u>	Converts a date into a number between 0 and 59.
<u>\$second</u>	Converts a date into a number between 0 and 59. May be 60 to account for leap seconds.
<u>\$millisecond</u>	Returns the millisecond portion of a date as an integer between 0 and 999.

一个栗子

```
/* mongodb json
db.LogisticsTicket.aggregate(
{$match:{"TrackingFinished":false,Transportation:{$in:["Air","Ocean"]},NextNodeInfo:{$ne:null},"NextNodeInfo.ExpecteArriveOn":{$lt:new Date()}}},
{$project:{"_id":0,FromWarehouse:1,ToWarehouse:1,Transportation:1,"NextNodeInfo":1,"TrackingFinished":1}},
{$group:{"_id":{"from":"$FromWarehouse",to:"$ToWarehouse",transportation:"$Transportation",nodeName:"$NextNodeInfo.Name"},count:{$sum:1}}},
{$sort:{from:1,to:1,transportation:1,nodeName:1}}
)
*/
```

```

var list = new List<LogisticsMonitoringInfo>();

var project = new BsonDocument()
{
    {"$project", new BsonDocument(){ {"_id", 0},
                                      {"FromWarehouse", 1},
                                      {"ToWarehouse", 1},
                                      {"Transportation", 1},
                                      {"CurrentNodeInfo", 1},
                                      {"TrackingFinished", 1}
                                    }
    }
};

var match = new BsonDocument()
{
    {"$match", new BsonDocument()
    {
        {"TrackingFinished", false},
        {"Transportation", new BsonDocument(){ {"$in", new BsonArray{"Air", "Ocean"}}}},
        {"CurrentNodeInfo", new BsonDocument(){ {"$ne", BsonNull.Value } }},
        {"NextNodeInfo.ExpectedArriveOn", new BsonDocument(){ {"$lt", DateTime.UtcNow}}},
        {"NextNodeInfo.IsJudgeTimeliness", true}
    }
    }
};

var group = new BsonDocument()
{
    {"$group", new BsonDocument(){ {"_id", new BsonDocument()
    {
        {"from", "$FromWarehouse"},
        {"to", "$ToWarehouse"},
        {"transportation", "$Transportation"},
        {"nodeName", "$CurrentNodeInfo.Name"}
    }
    },
    {"count", new BsonDocument()
    {
        {"$sum", 1}
    }
    }
    }
    }
};

var sort = new BsonDocument()
{
    {"$sort", new BsonDocument(){ {"from", 1}, {"to", 1}, {"transportation", 1}, {"nodeName", 1}}}
};

var pipeline = new[] { match, project, group, sort };
var args = new AggregateArgs { Pipeline = pipeline };
var result = this._mongoDBRepositoryContext.GetCollectionForType(typeof(LogisticsTicket)).Aggregate(args);
foreach (var doc in result)
{
    var from = doc["_id"]["from"].AsString;
    var to = doc["_id"]["to"].AsString;
    Transportation transportation = (Transportation)Enum.Parse(typeof(Transportation), doc["_id"]["transportation"].AsString);
    var nodeName = doc["_id"]["nodeName"].AsString;
    var count = doc["count"].AsInt32;
    NodeCounter nodeCounter = new NodeCounter(nodeName, count);
    var info = new LogisticsMonitoringInfo(from, to, transportation, nodeCounter);

    list.Add(info);
}

return list;

```

总结

- 对于大多数的聚合操作，聚合管道可以提供很好的性能和一致的接口。
- 使用起来比较简单，和MapReduce一样，它也可以作用于分片集合。
- 输出的结果只能保留在一个文档中，要遵守BSON Document大小限制（当前是16M）。
- 管道对数据的类型和结果的大小会有一些限制，对于一些简单的固定的聚集操作可以使用管道，但是对于一些复杂的、大量数据集的聚合任务还是使用MapReduce。



谢谢！