

Migrating from (Spring Data) JPA to Spring Data JDBC

by Jens Schauder

Copyright © 2020 VMware, Inc. or its affiliates.

vmware

I Have Questions!

Who Loves JPA?



spring

Have You Used Spring Data JDBC in Production?



spring

Have You Used Spring Data JDBC?



spring

Do you consider migrating?



spring

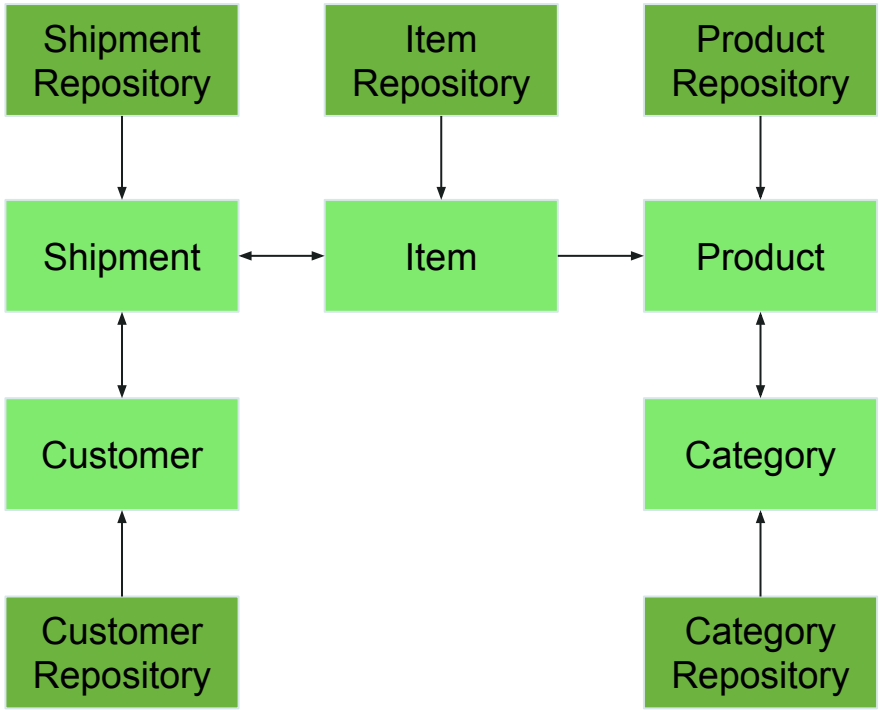
Why?



spring

Before We Start

Assume a Serious project


Shipment Repository
Item Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

You Better Have Tests!

Baby Steps

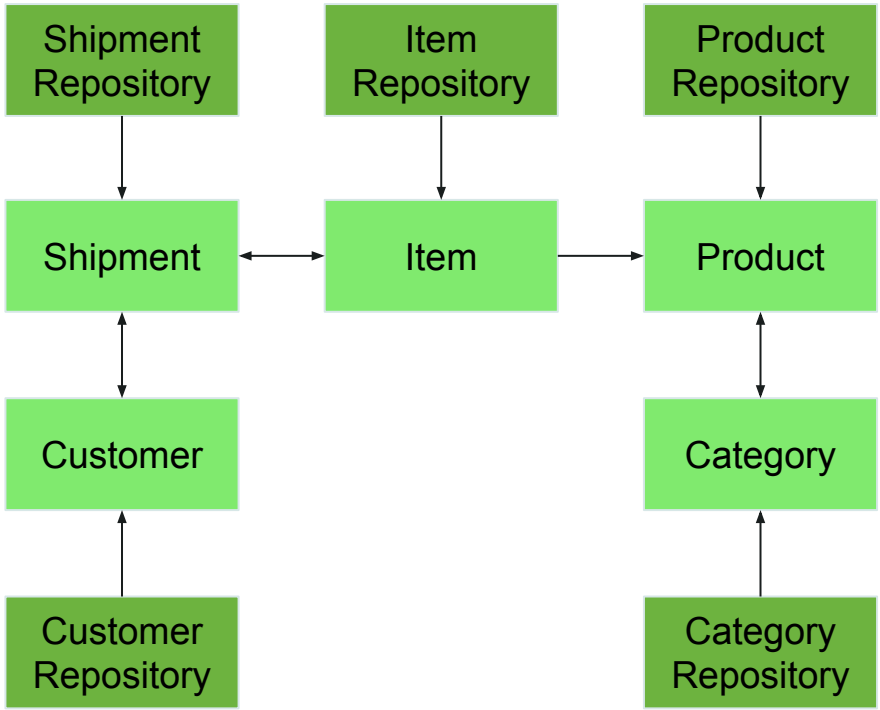
Mikado Method

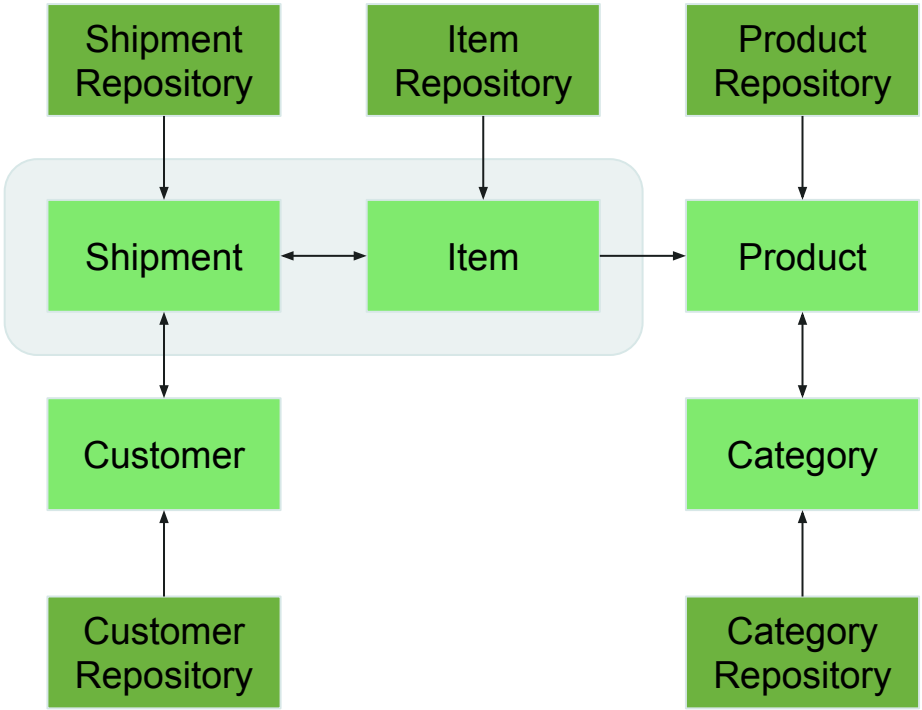
SD JDBC uses Aggregates

Breadth-First or Depth First

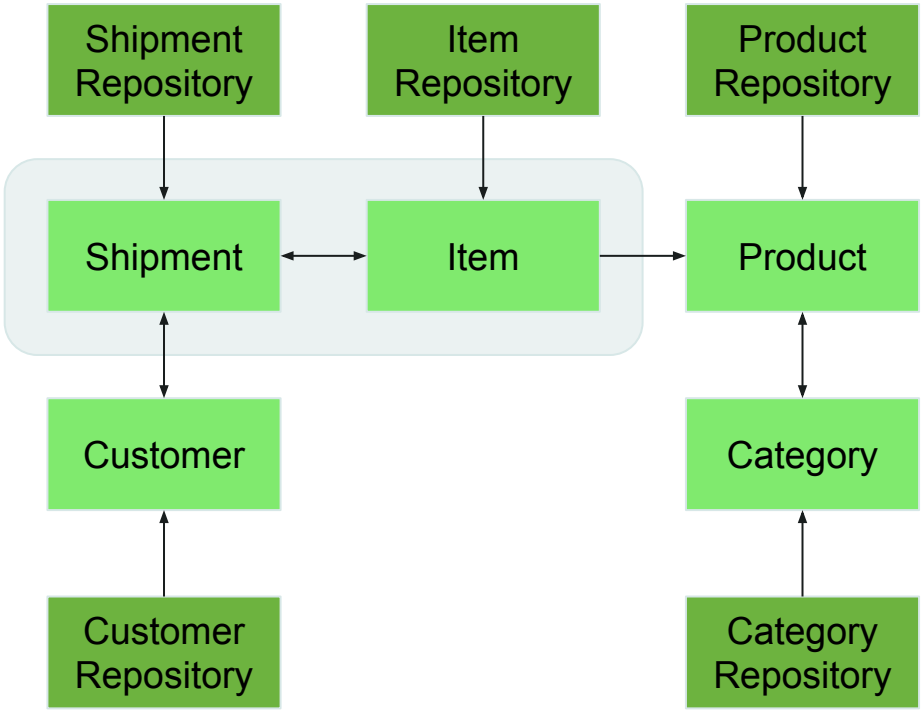
Lets Go!

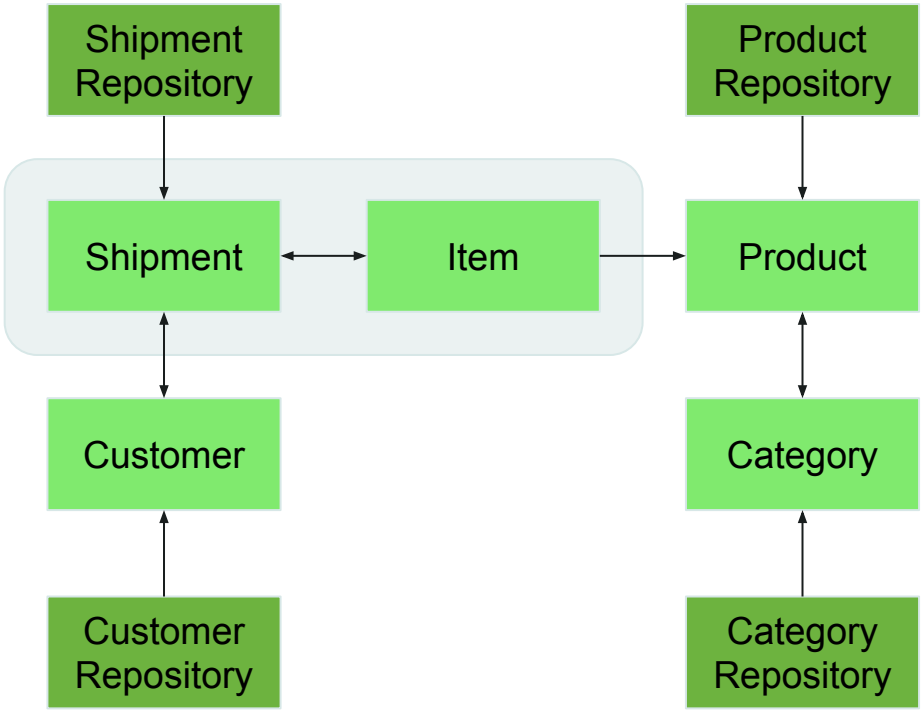
Identify Aggregates


Shipment Repository
Item Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository


Shipment Repository
Item Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

Remove extra Repositories

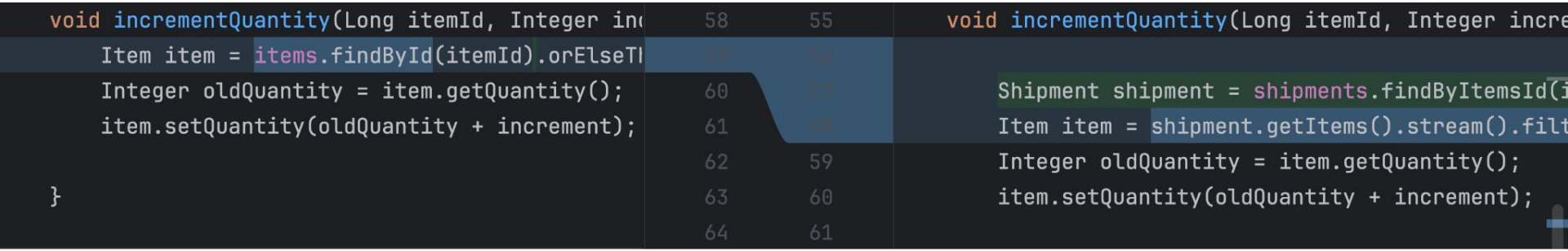

Shipment Repository
Item Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

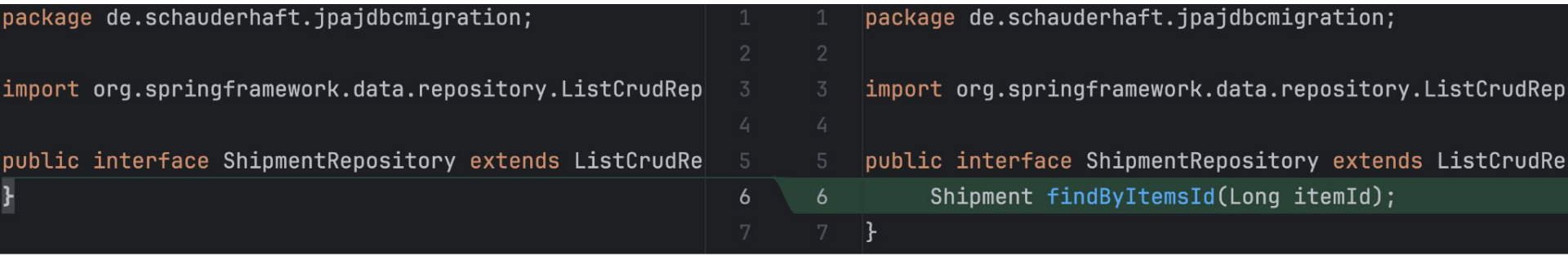
```java
package de.schauderhaft.jpajdbcmigration;

import org.springframework.data.repository.ListCrudRepository;

public interface ItemRepository extends ListCrudRepository<Item, Long> {
}
```


void incrementQuantity(Long itemId, Integer inc
Item item = items.findById(itemId).orElseTh
Integer oldQuantity = item.getQuantity();
item.setQuantity(oldQuantity + increment);
}
58
60
61
62
63
64
55
57
59
60
61
void incrementQuantity(Long itemId, Integer incre
Shipment shipment = shipments.findByItemsId(i
Item item = shipment.getItems().stream().filt
Integer oldQuantity = item.getQuantity();
item.setQuantity(oldQuantity + increment);

spring

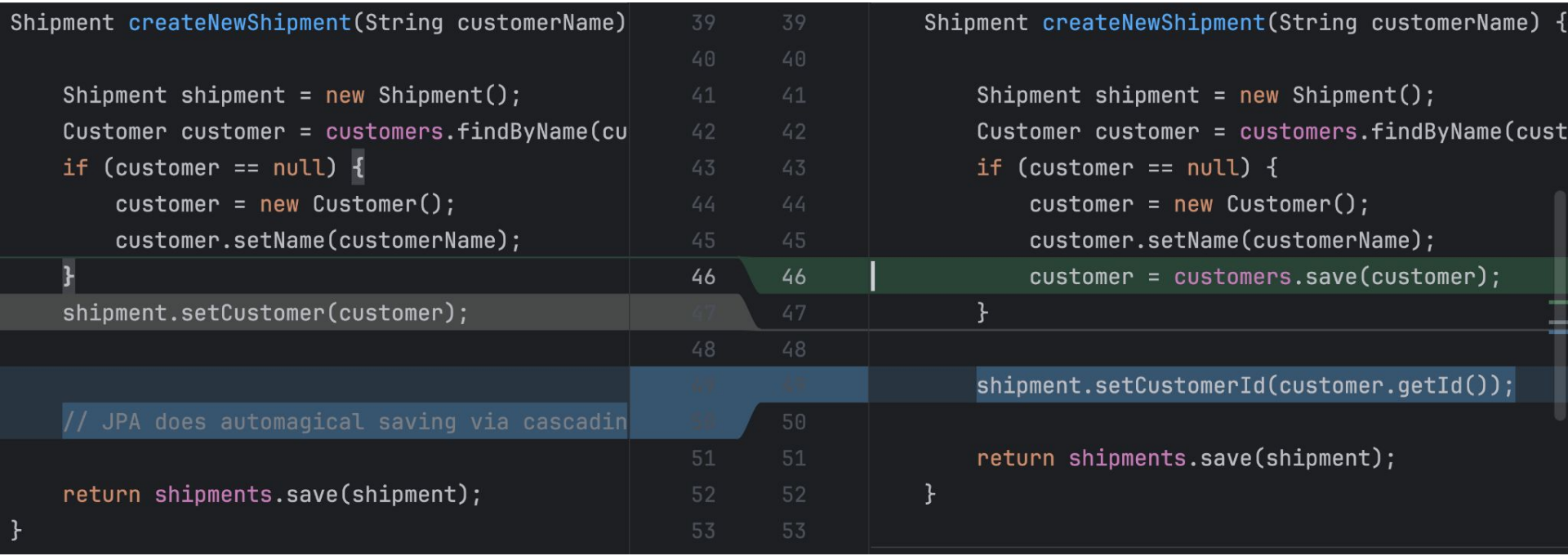

package de.schauderhaft.jpajdbcmigration;
import org.springframework.data.repository.ListCrudRep
public interface ShipmentRepository extends ListCrudRe
}
package de.schauderhaft.jpajdbcmigration;
import org.springframework.data.repository.ListCrudRep
public interface ShipmentRepository extends ListCrudRe
Shipment findByItemsId(Long itemId);
}

spring

Make `save` explicit


void assignProductCategories(Long productId, Lo
Product product = products.findById(product
List<ProductCategory> categoryList = new Ar
for (Long categoryId : categoryIds) {
categoryList.add(ProductCategory.of(pro
}
product.setCategories(categoryList);
}
List<Product> productsByCategoryName(String cat
@ 63
63 @
64
65
66
67
68
69
70
71
72
73
74
void assignProductCategories(Long productId, Long
Product product = products.findById(productId
List<ProductCategory> categoryList = new Arra
for (Long categoryId : categoryIds) {
categoryList.add(ProductCategory.of(produ
}
product.setCategories(categoryList);
products.save(product);
}

spring


Shipment createNewShipment(String customerName)
Shipment shipment = new Shipment();
Customer customer = customers.findByName(cu
if (customer == null) {
customer = new Customer();
customer.setName(customerName);
}
shipment.setCustomer(customer);
// JPA does automagical saving via cascadin
return shipments.save(shipment);
}
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
Shipment createNewShipment(String customerName) {
Shipment shipment = new Shipment();
Customer customer = customers.findByName(cust
if (customer == null) {
customer = new Customer();
customer.setName(customerName);
customer = customers.save(customer);
}
shipment.setCustomerId(customer.getId());
return shipments.save(shipment);
}

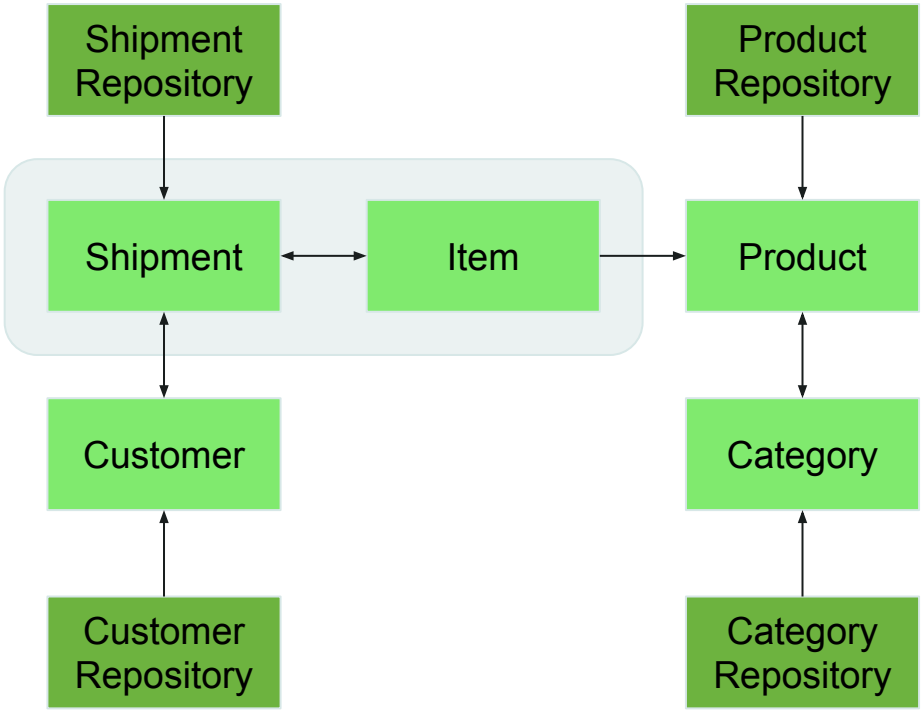
spring

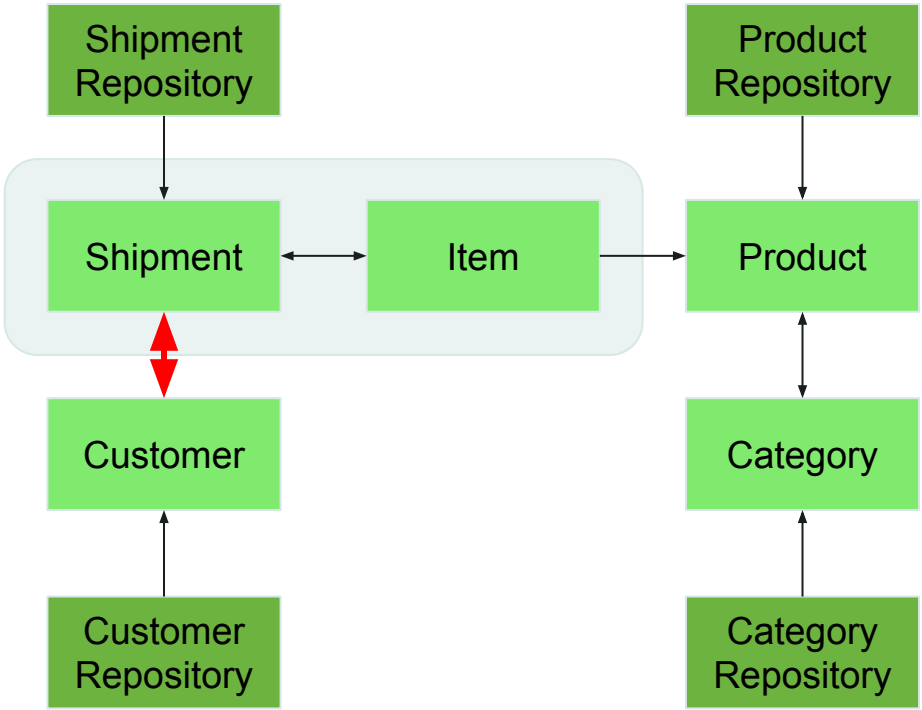
Remove `cascade` between Aggregates

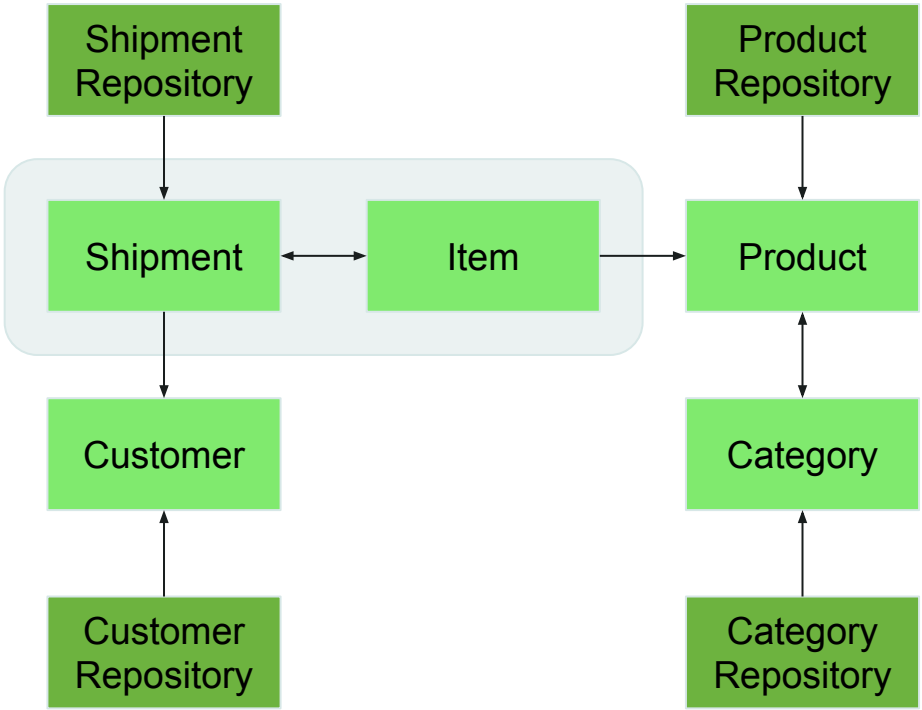
Bidirectional-References

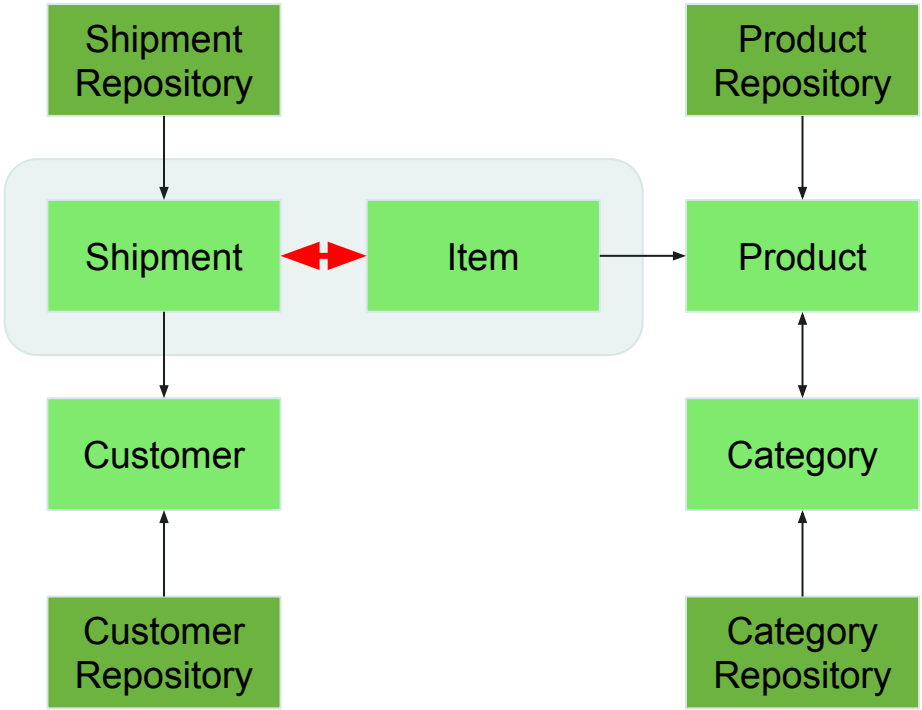
=>

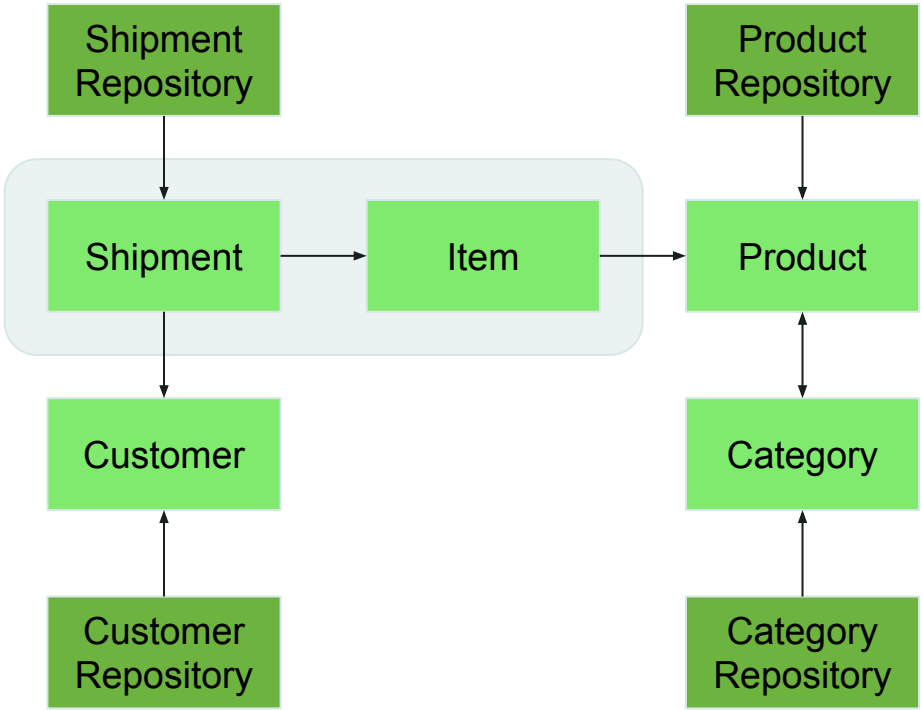
Unidirectional-References

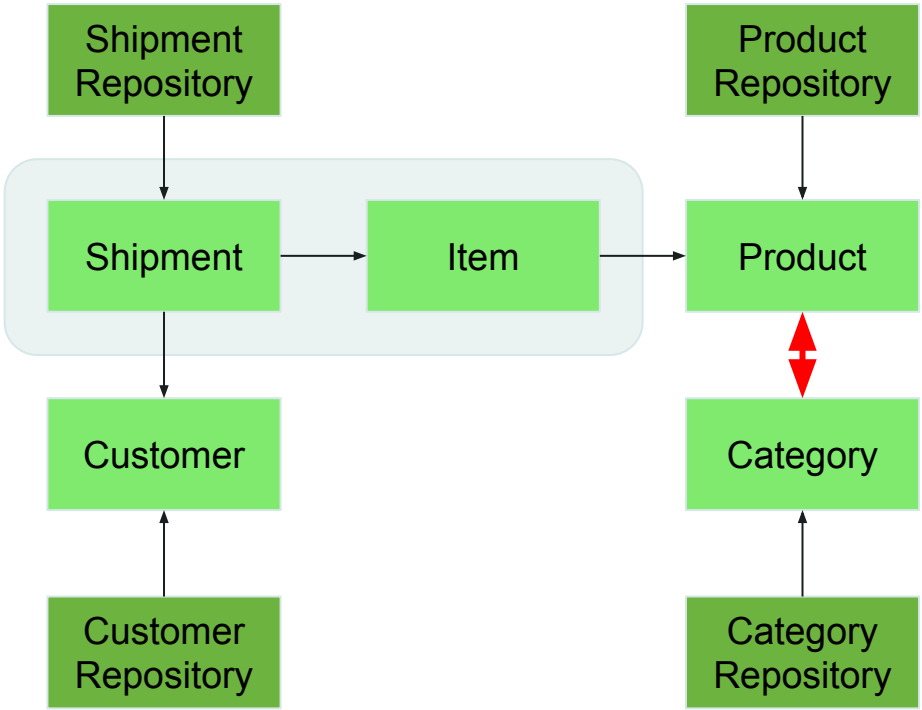

Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

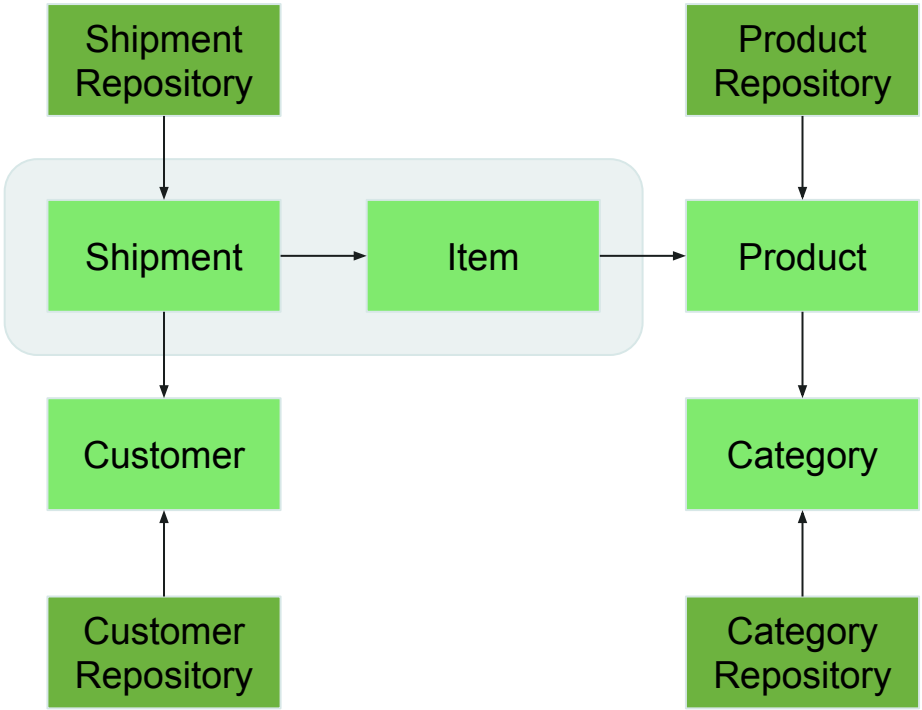

Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

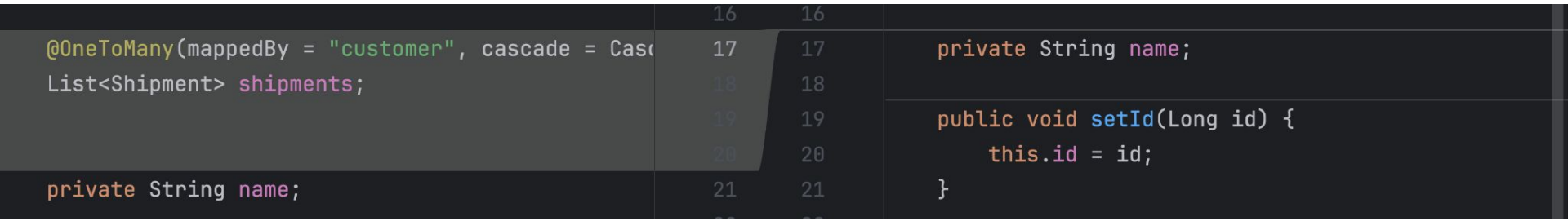

Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

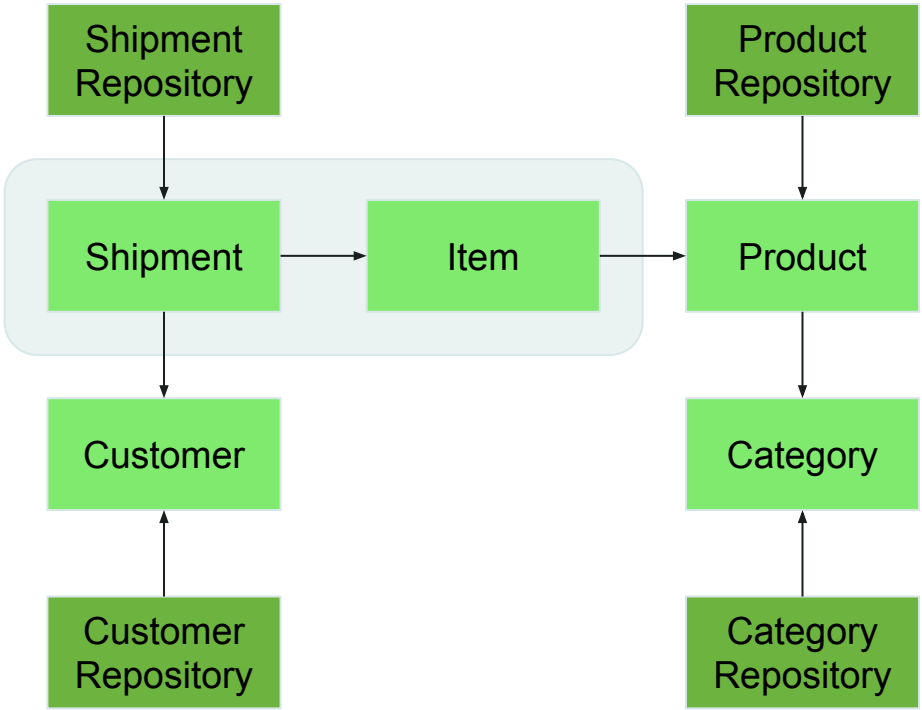

Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository

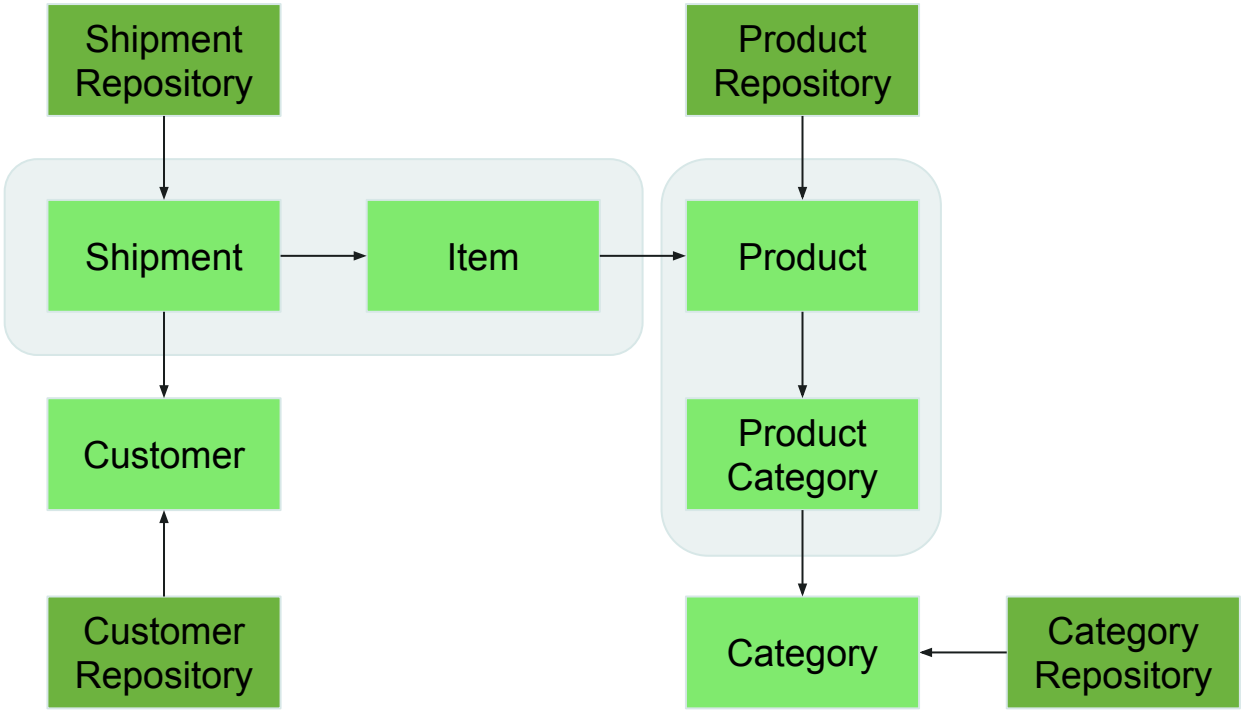

Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository


@OneToMany(mappedBy = "customer", cascade = Cas
List<Shipment> shipments;
private String name;
16
17
18
19
20
21
16
17
18
19
20
21
private String name;
public void setId(Long id) {
this.id = id;
}

spring

Make M:N Join Tables explicit


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Category
Customer Repository
Category Repository


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Product Category
Customer Repository
Category
Category Repository

```java
@Entity
@Table(name = "product_categories")
public class ProductCategory {

    static ProductCategory of(Product product, Category category) {

        ProductCategory productCategory = new ProductCategory();
        productCategory.setProduct(product);
        productCategory.setCategory(category);

        return productCategory;
    }


    @Id
    @ManyToOne
    @JoinColumn(name = "products_id")
    Product product;

    @Id
    @ManyToOne
    @JoinColumn(name = "categories_id")
    Category category;
```

spring

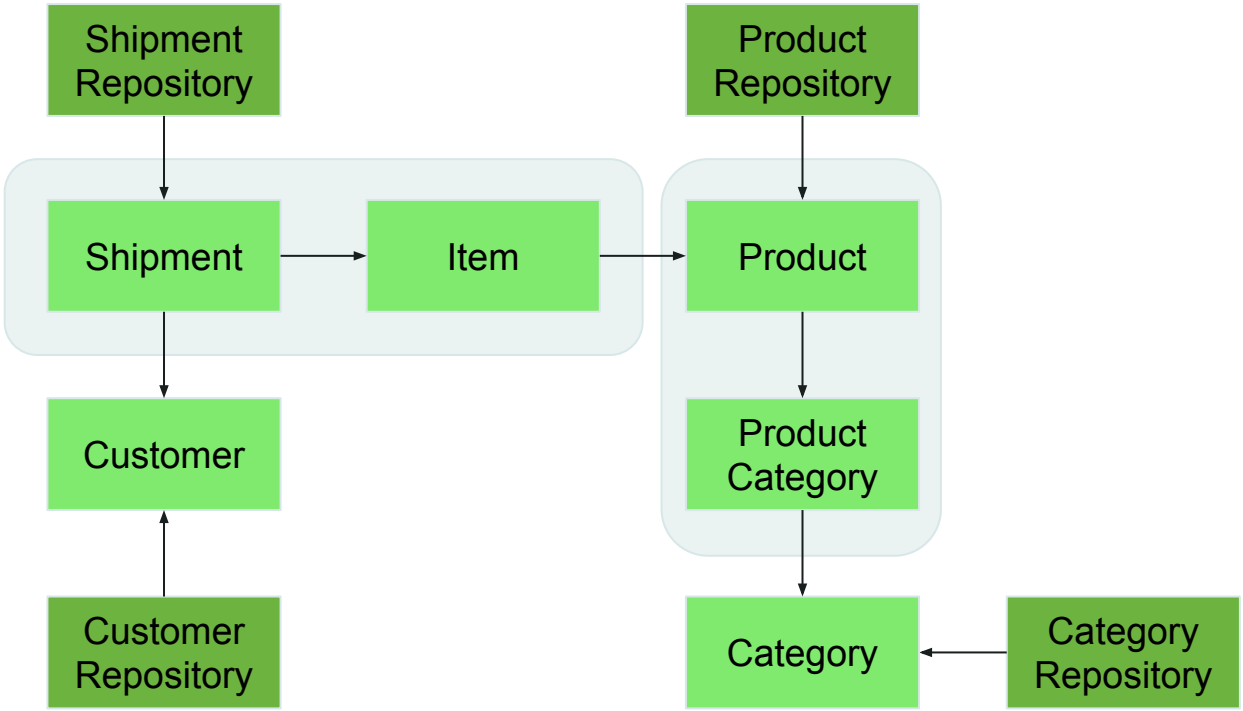
```
    String name;                                                    19  18      String name;
    @ManyToMany(cascade = CascadeType.ALL)                          20  19
    @JoinTable(joinColumns = {@JoinColumn(name = "pro               21  20      @OneToMany(cascade = CascadeType.ALL, mappedBy =
    private List<Category> categories = new ArrayList               22  21      private List<ProductCategory> categories = new Ar
    private String description;                                     23  22      private String description;
```

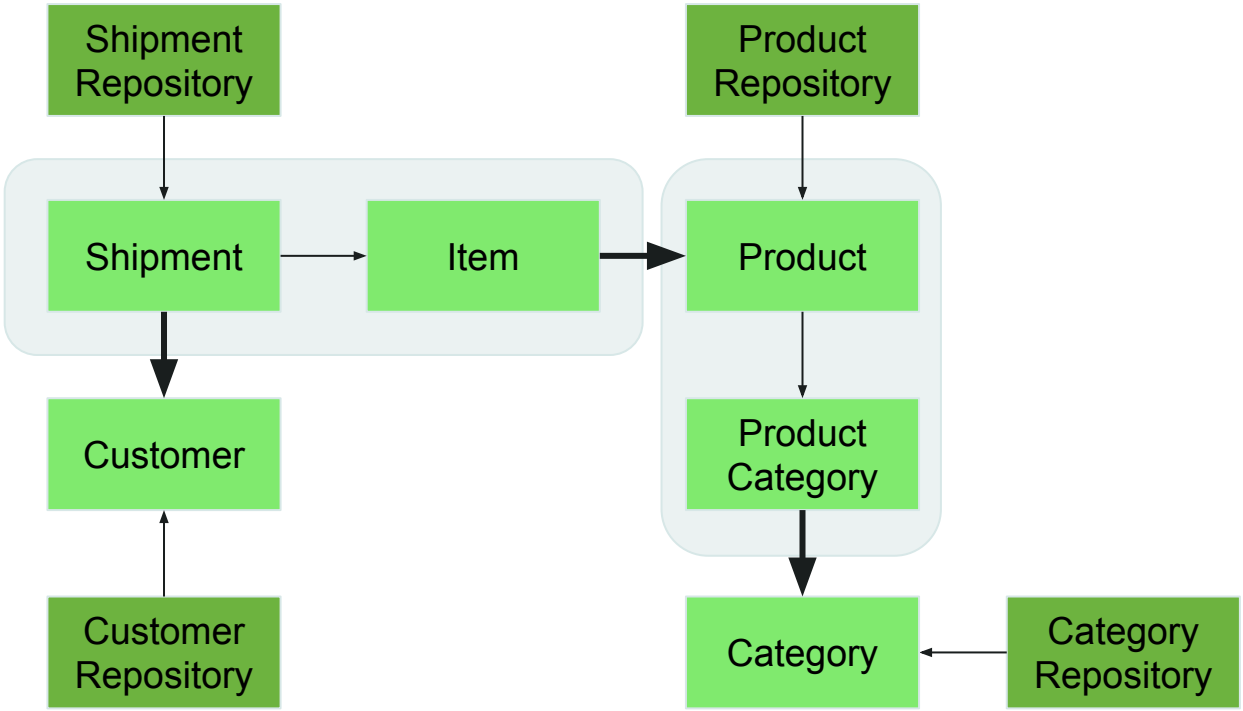
spring

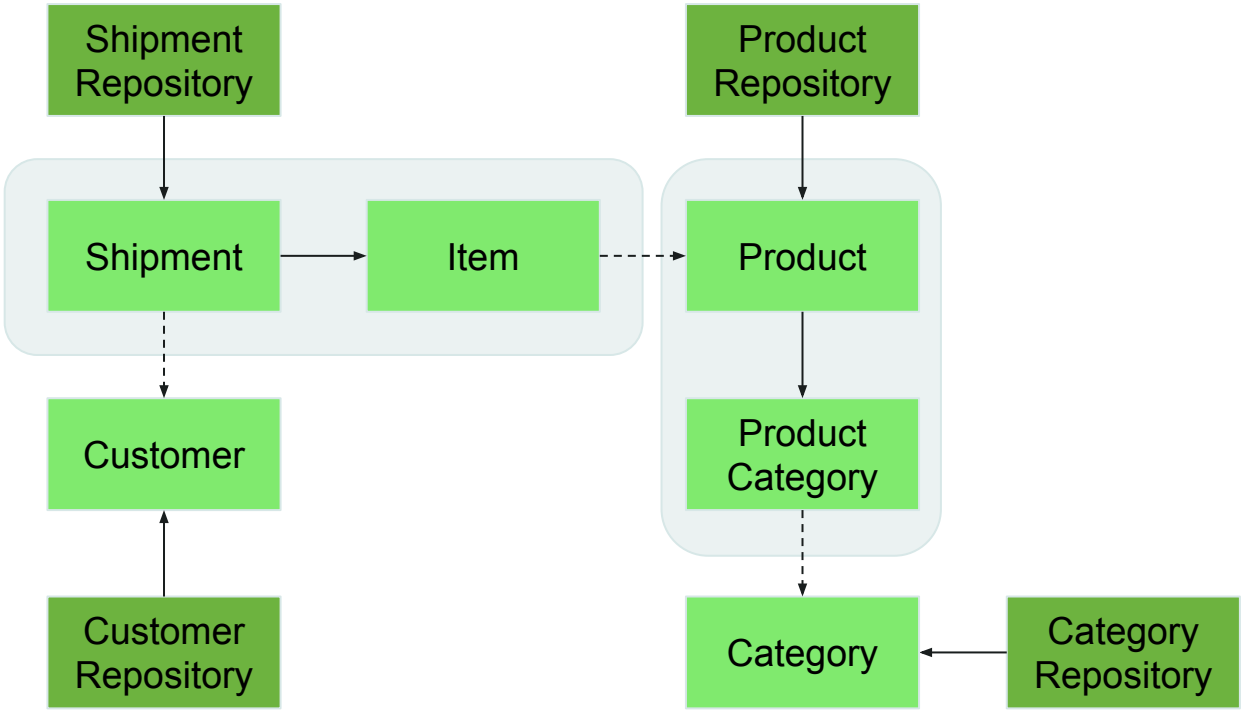
Cross Aggregate References

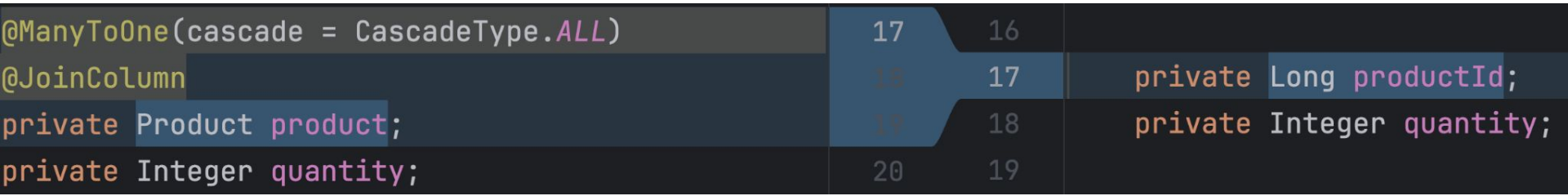
=>

Id


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Product Category
Customer Repository
Category
Category Repository


Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Product Category
Customer Repository
Category
Category Repository

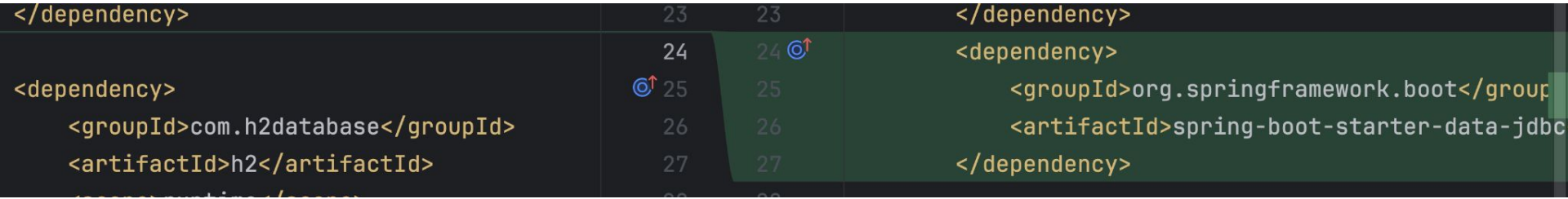

Shipment Repository
Product Repository
Shipment
Item
Product
Customer
Product Category
Customer Repository
Category
Category Repository


@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn
private Product product;
private Integer quantity;
17
19
20
16
17
18
19
private Long productId;
private Integer quantity;

spring

Spring Data JDBC

Add Spring Data JDBC


23 23
24 24
25 25
26 26
27 27
</dependency>
<dependency>
<groupId>com.h2database</groupId>
<artifactId>h2</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jdbc</artifactId>
</dependency>

spring

Create Id Generator

```java
@Component
public class IdGenerator implements BeforeConvertCallback<Category> {

    @Autowired
    JdbcTemplate jdbcTemplate;

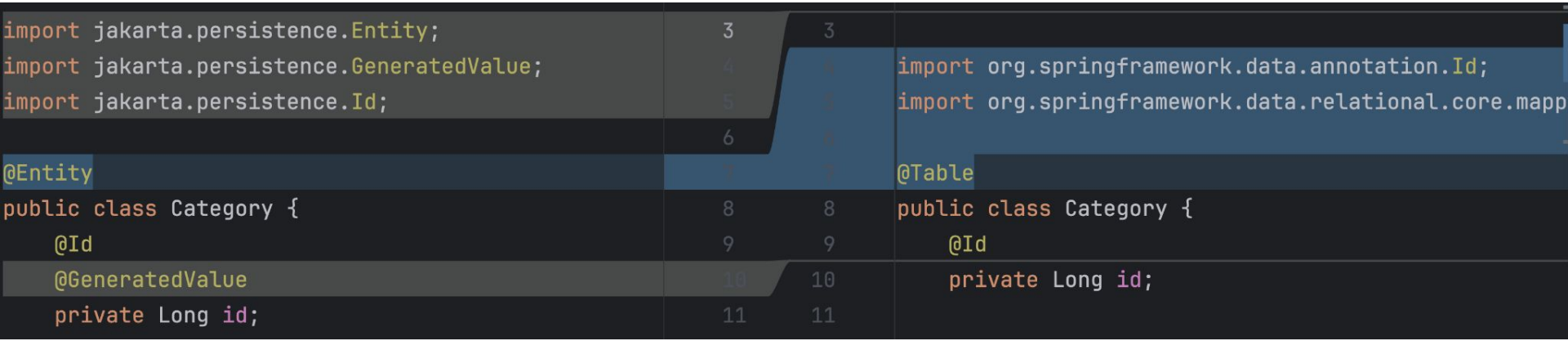
    @Override
    public Category onBeforeConvert(Category category) {

        if (category.getId() == null) {
            category.setId(obtainId());
        }
        return category;
    }

    private Long obtainId() {
        return jdbcTemplate.queryForObject("values next value for category_seq", Long.class);
    }
}
```

Create Read/Write Converter

Migrate One Aggregate at a Time


import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
@Entity
public class Category {
@Id
@GeneratedValue
private Long id;
import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapp
@Table
public class Category {
@Id
private Long id;

spring

Conclusion

A lot of work!

Can be broken down

Mostly untangling JPA

Only new stuff with SD JDBC?

Resources

Demo Project
https://github.com/schauder/talk-migrate-jpa-to-spring-data-jdbc

Mikado Method
https://www.manning.com/books/the-mikado-method

Idea behind Spring Data JDBC

https://spring.io/blog/2018/09/24/spring-data-jdbc-references-and-aggregates

Thank you

Contact me at jens.schauder@broadcom.com



© 2020 Spring. A VMware-backed project. vmware®