

# domain & REST

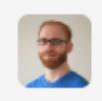
Domain-Driven APIs for the web

Oliver Gierke

 /  olivergierke

olivergierke (Oliver Gierke) 

← → ⌂ ⌂ GitHub, Inc. [US] https://github.com/olivergierke

 Search GitHub Pull requests Issues Gist  + 

  
**Oliver Gierke**  
olivergierke

+ Contributions  Repositories  Public activity 

**Popular repositories**

 <a href="#">spring-restbucks</a>	284 ★
Implementation of the sample from REST in P...	
 <a href="#">repositories-deepdive</a>	110 ★
Sample code for the Spring Data JPA repository...	
 <a href="#">rest-microservices</a>	63 ★
Sample for Spring Boot based REST microser...	
 <a href="#">whoops-architecture</a>	38 ★
Sample code for my talk "Whoops! Where did ..."	
 <a href="#">spring-rest</a>	28 ★
Sample project for Spring 3 REST style web a...	

**Repositories contributed to**

 <a href="#">spring-projects/spring-data-rest</a>	342 ★
Spring Data REST Exporter	
 <a href="#">spring-projects/spring-data-com...</a>	168 ★
Spring Data Commons. Interfaces and code s...	
 <a href="#">spring-projects/spring-data-exa...</a>	293 ★
Spring Data Example Projects	
 <a href="#">spring-projects/spring-data-jpa</a>	530 ★
Simplifies the development of creating a JPA...	
 <a href="#">spring-projects/spring-data-build</a>	8 ★
Modules to centralize common resources and ...	

**Contributions**

Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov  
M W F

Summary of pull requests, issues opened, and commits. [Learn how we count contributions.](#)

Less  More

**Organizations**

# Background

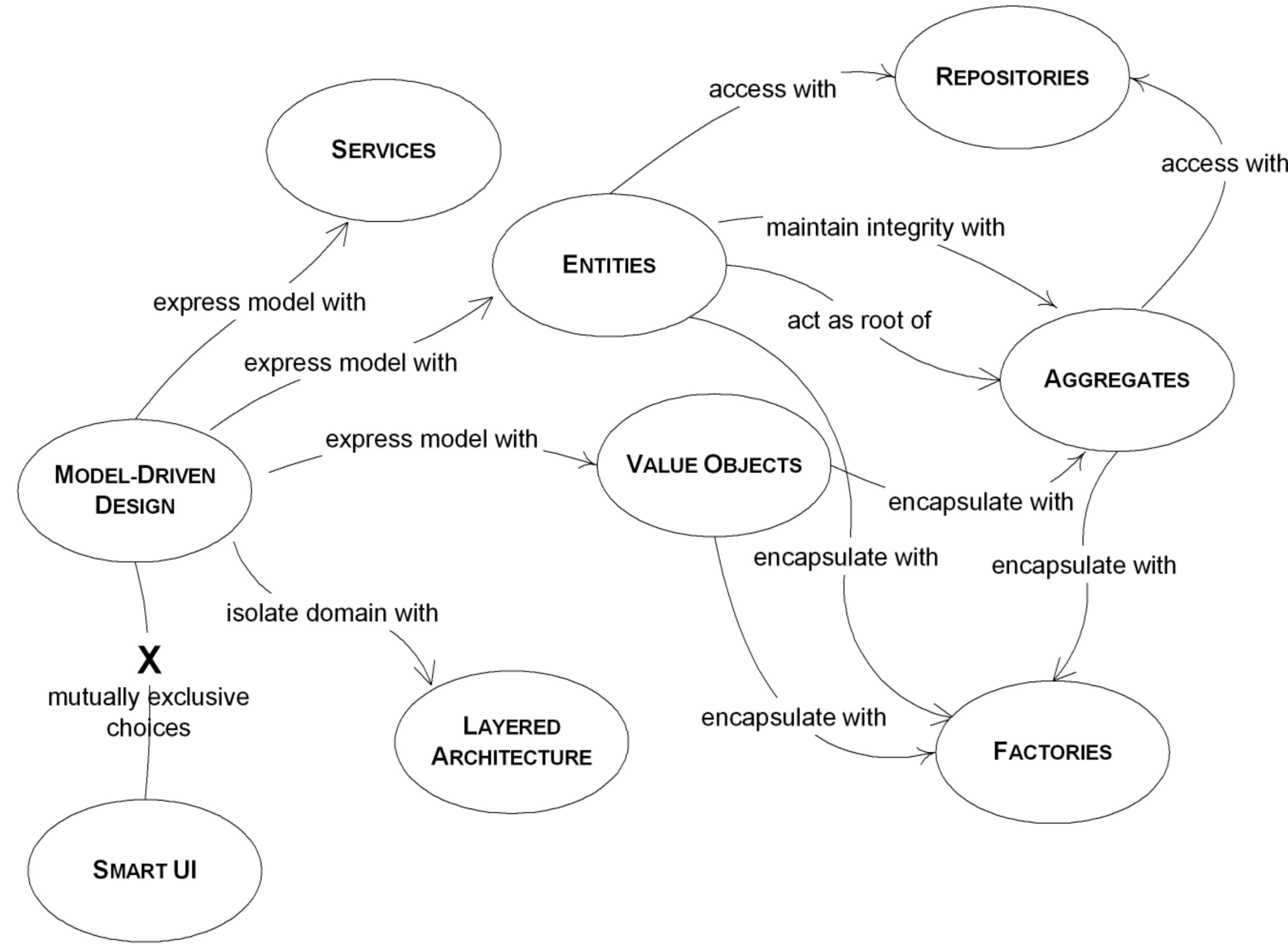
# Spring Data REST

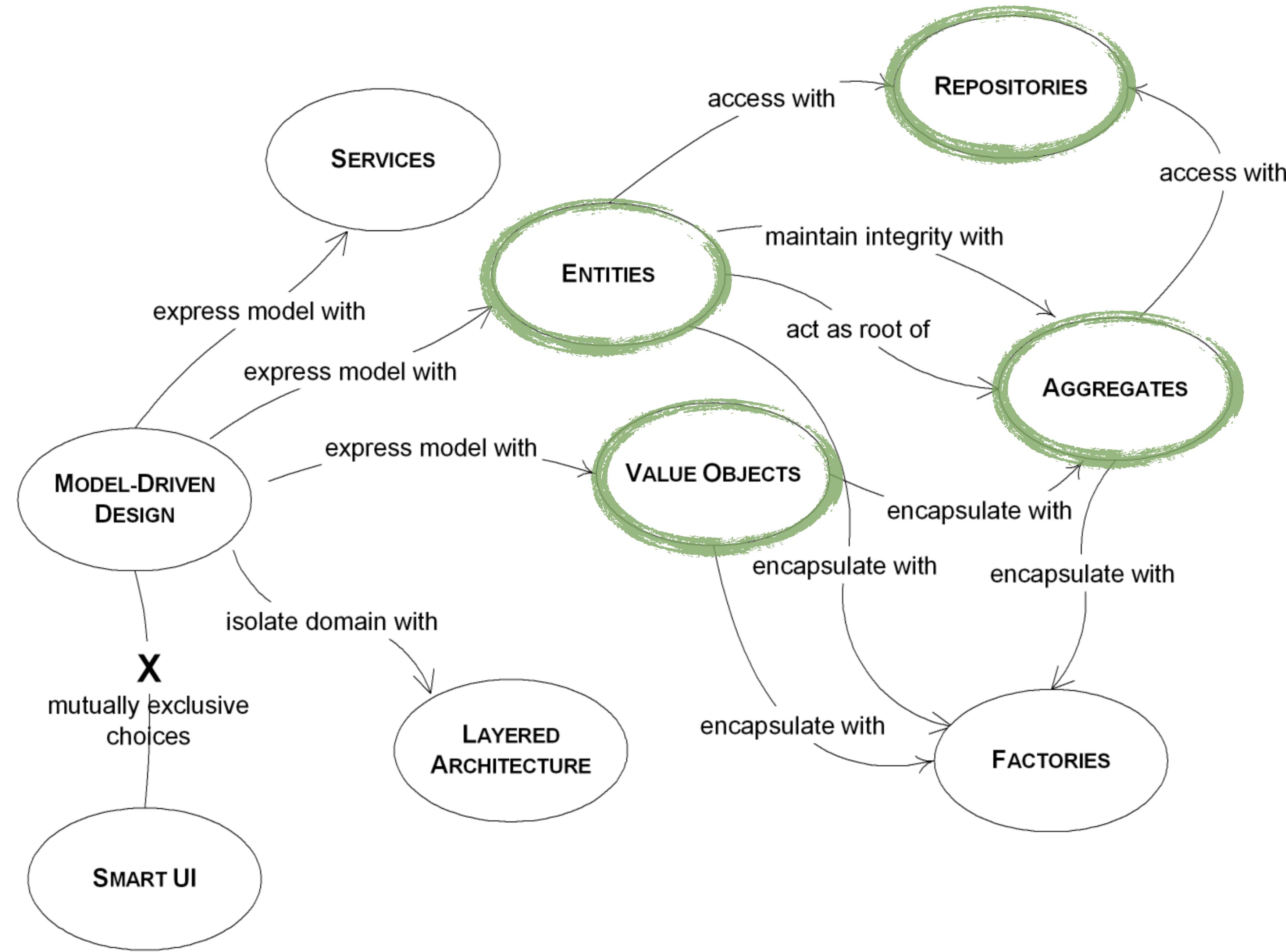
## Spring Data Repositories & Aggregates

## Spring HATEOAS Hypermedia for Spring MVC

**REST ≠  
CRUD via HTTP**

“What does it take to  
bridge the worlds of  
*DDD* & *REST*? ”





# Value objects

Value Objects are a  
PITA to build in  
some languages.

# Still, they're worth it.

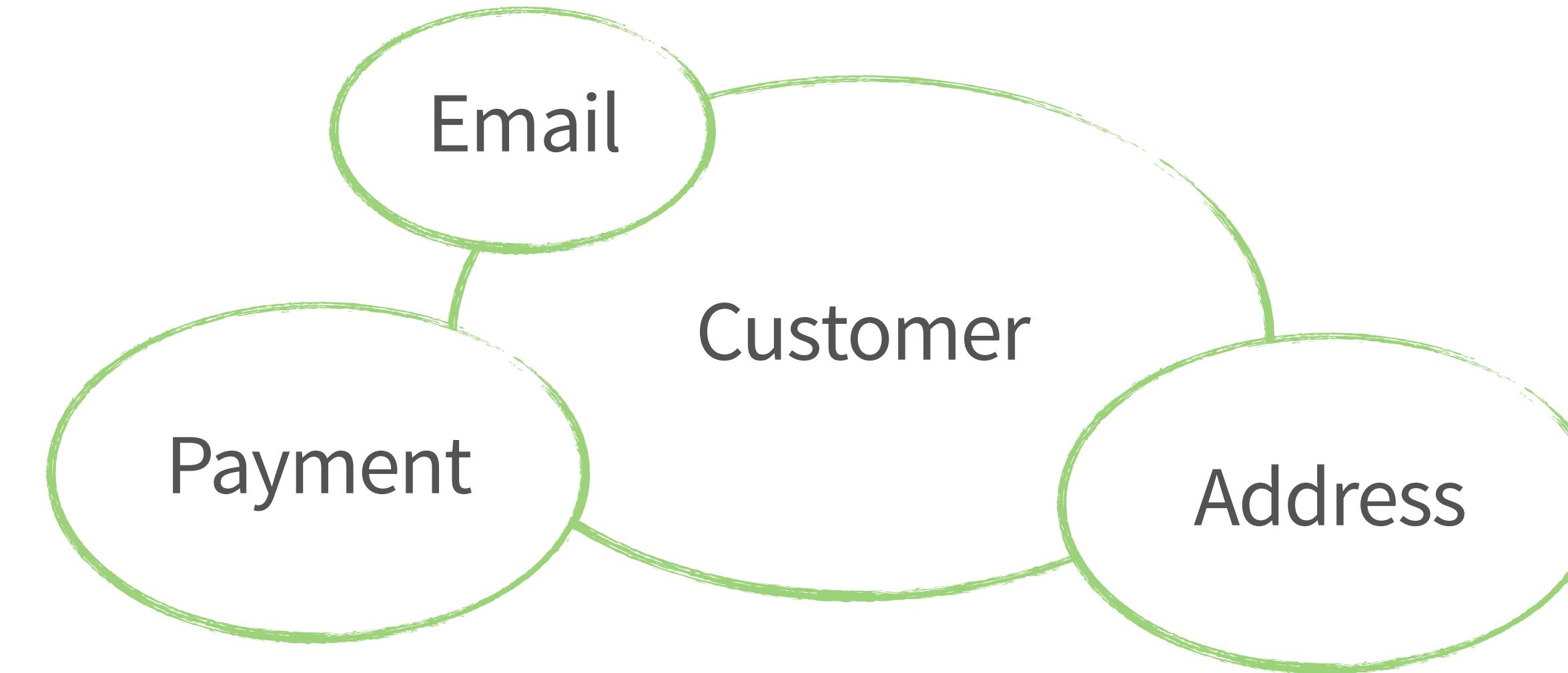
*See „Power Use of Value Objects in DDD“ by Dan Bergh Johnsson.*

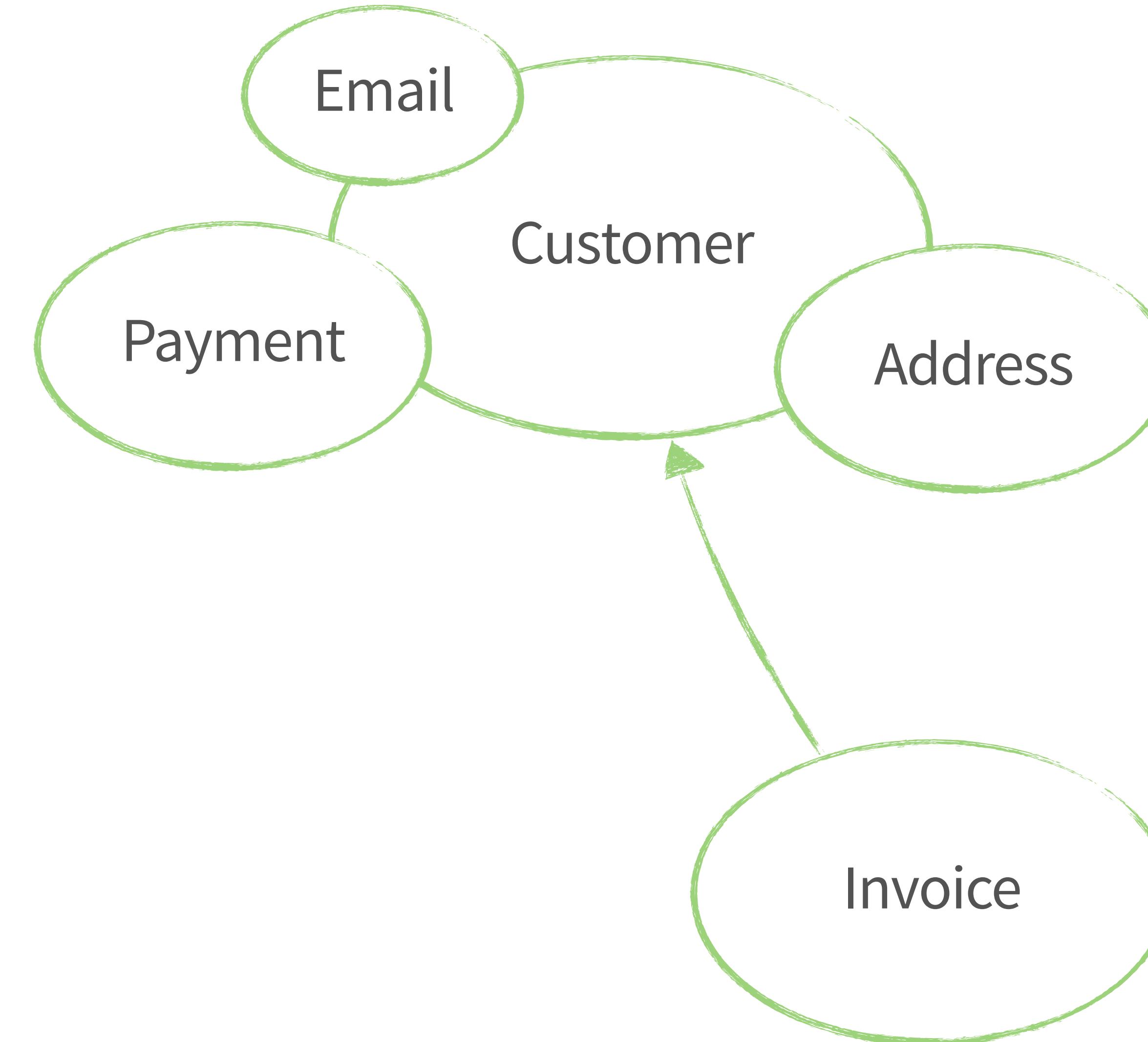
# Lombok – putting the spice back into Java.

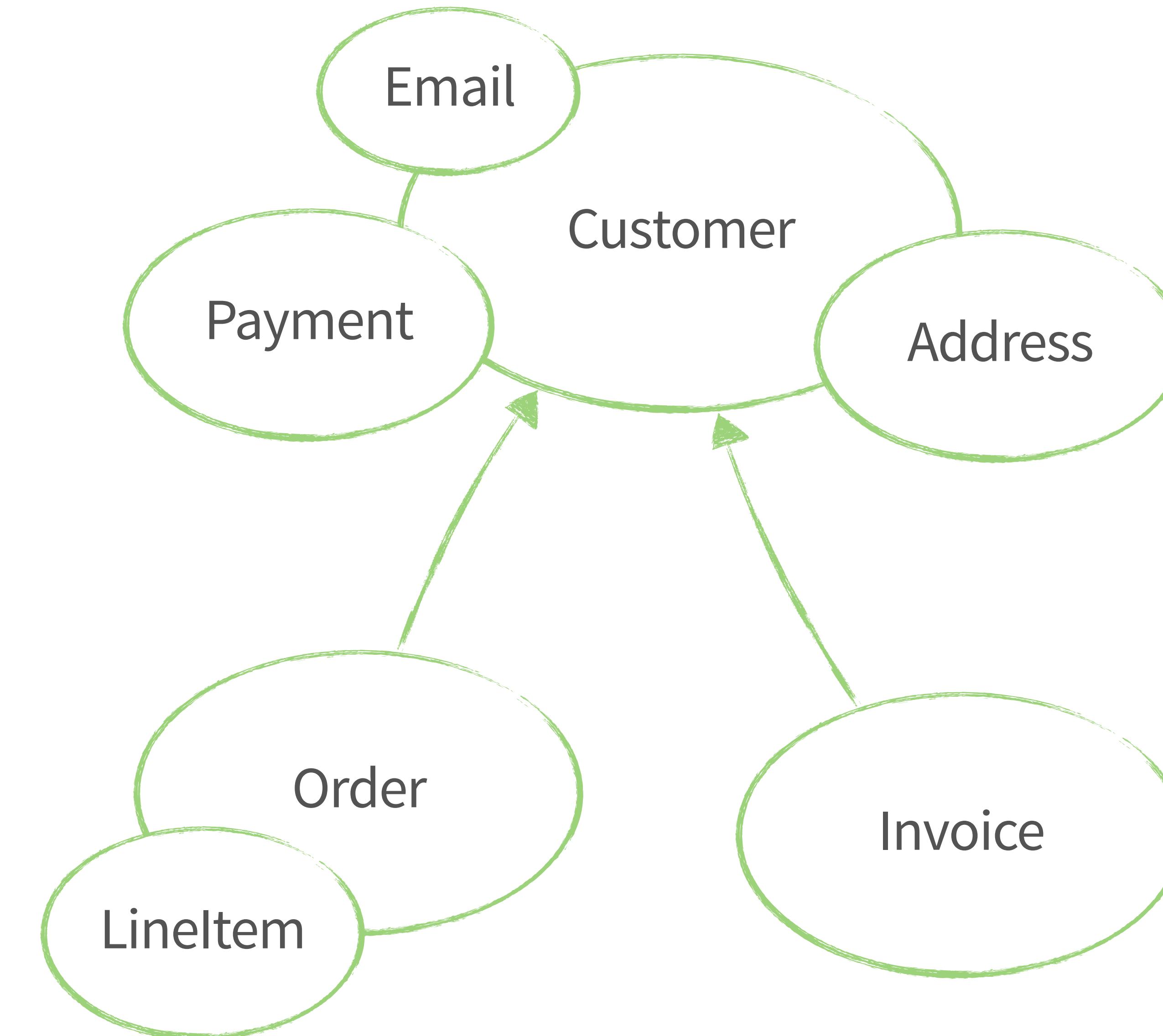
Key opponents:  
Mapping libraries  
that need to  
(de)serialize them.

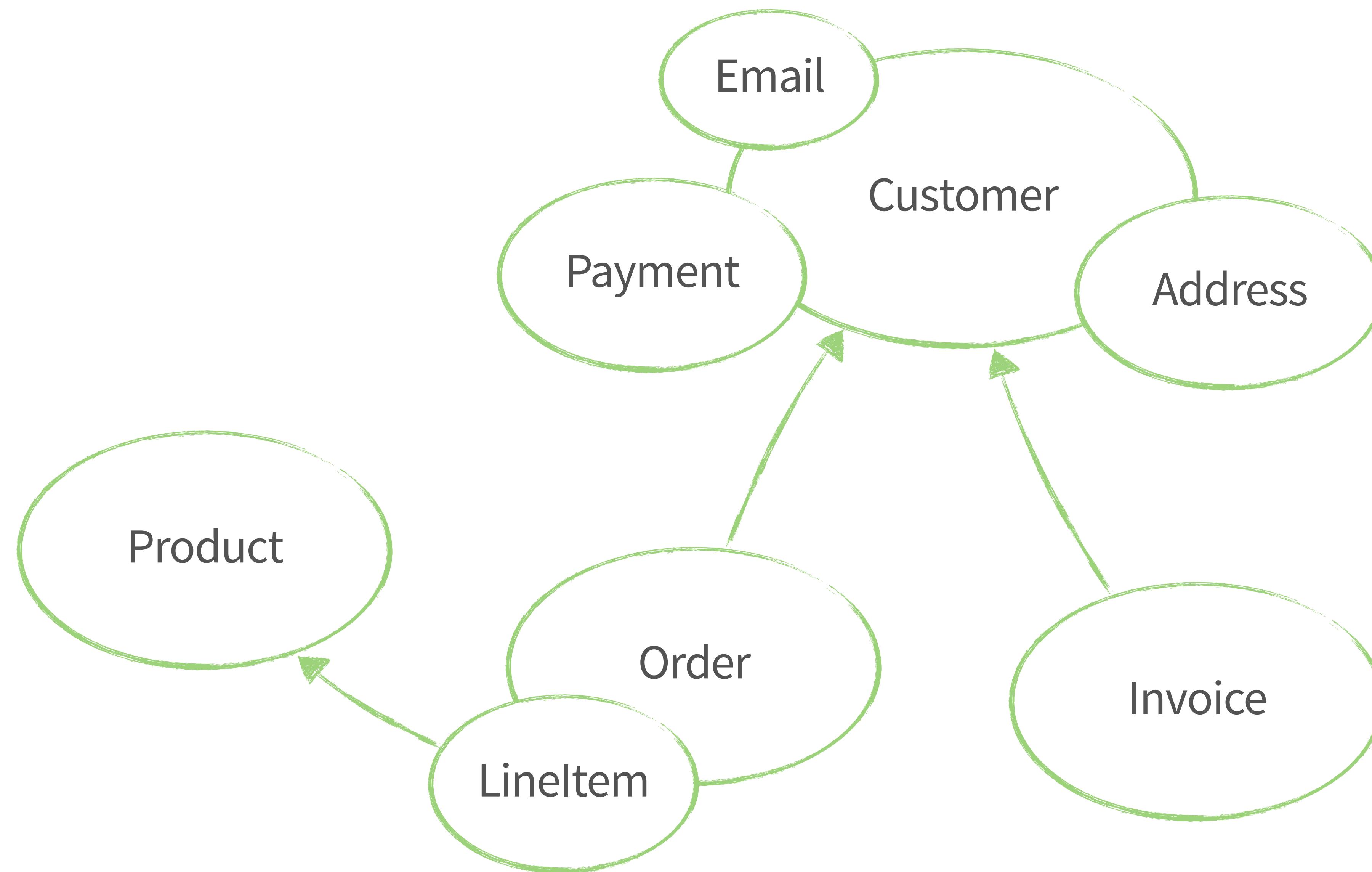
# Entities & Repositories

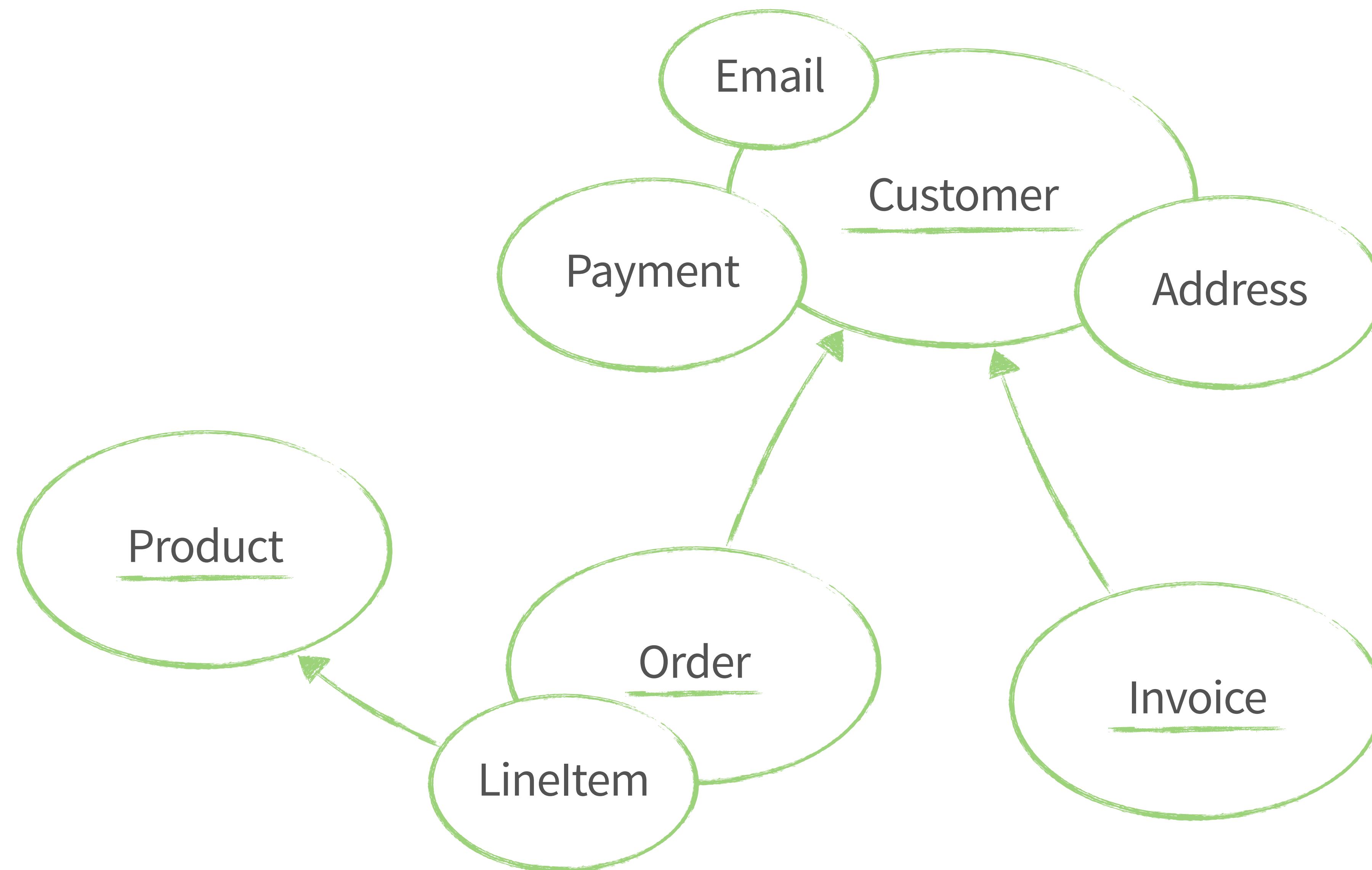












Entity +  
Repository =  
Aggregate

Aggregates form nice  
representation  
boundaries.

Aggregates become  
the key things  
to refer to.

Don't get trapped by  
datastore thinking.

Try to avoid  
bi-directional  
relationships.

# Domain Events

**Level 0: No events at all**

**Level 1: Explicit operations**

**Level 0: No events at all**

If you're calling two  
setters in a row, you're  
missing a concept.

**Level 2: Some operations as events**

**Level 1: Explicit operations**

**Level 0: No events at all**

Domain events as  
state transitions.

Expose important  
events to interested  
parties via feeds.

**Level 3: Event Sourcing**

**Level 2: Some operations as events**

**Level 1: Explicit operations**

**Level 0: No events at all**

# REST

# Representation design matters

# Aggregates

Identifiable  
Referable  
Scope of consistency

# Resources

Identifiable  
Referable  
Scope of consistency

# Hypermedia

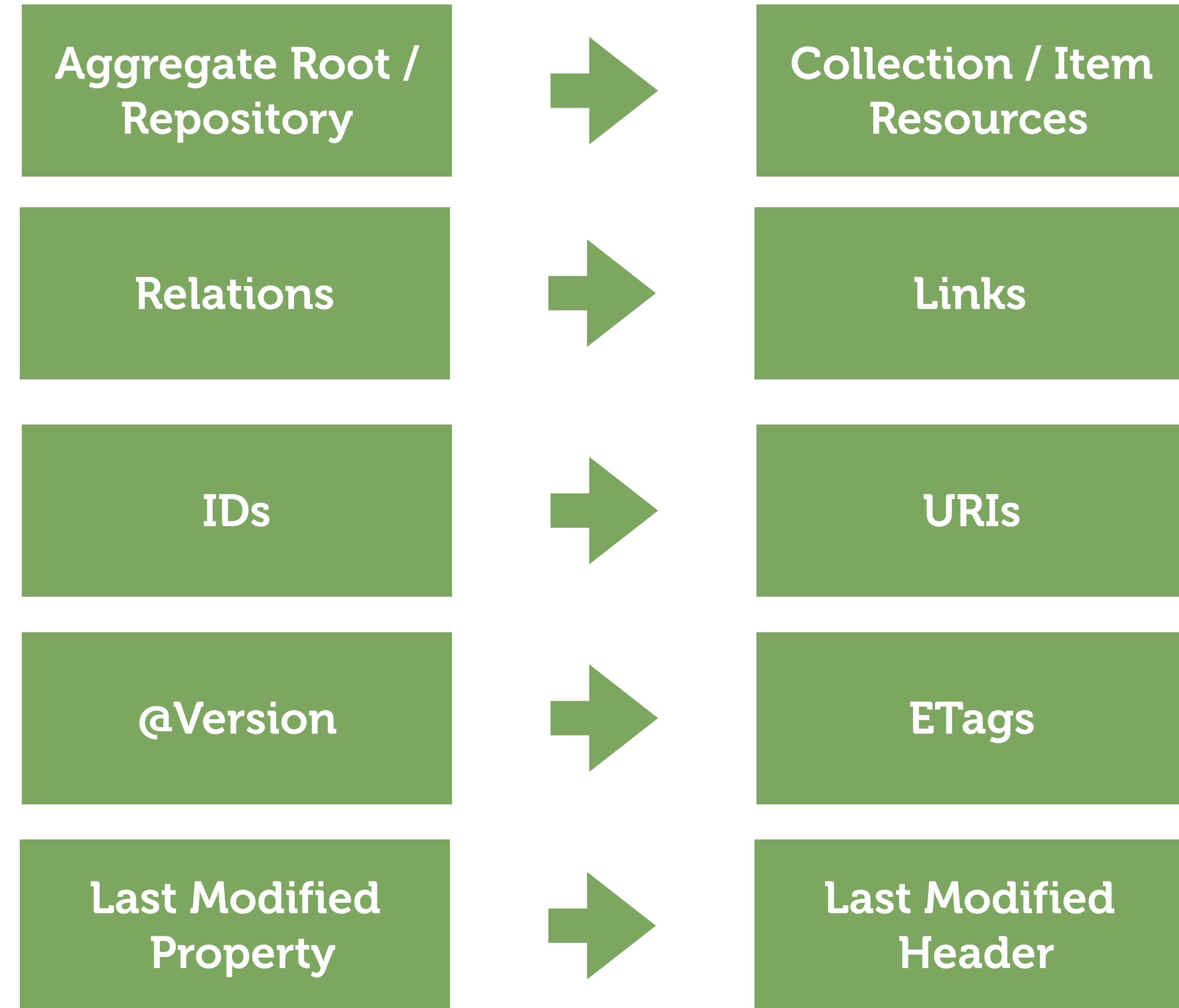
Serving data and  
navigation information  
at the same time.

Trading domain  
knowledge with protocol  
complexity in clients.

Reducing decisions in  
clients to whether a  
link is present or not.

Prefer explicit  
state transitions over  
poking at your resources  
using PATCH.

Translate domain  
concepts into web-  
appropriate ones.



# RESTBucks

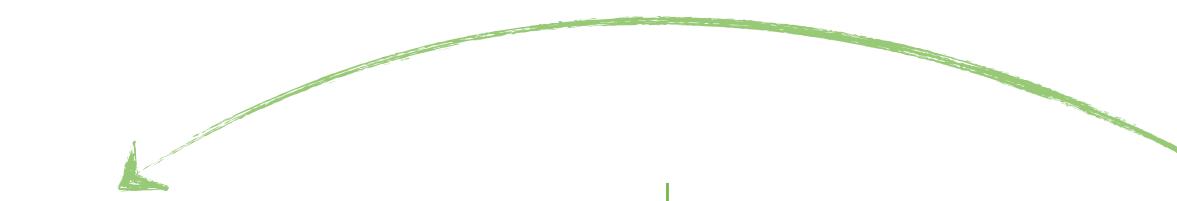


Method	URI	Action	Step
POST	/orders	Create new order	1
POST/PATCH	/orders/{id}	Update the order (only if "payment expected")	2
DELETE	/orders/{id}	Cancel order (only if "payment expected")	3
PUT	/orders/{id}/payment	Pay order (only if "payment expected")	4
Barista preparing the order			
GET	/orders/{id}	Poll order state	5
GET	/orders/{id}/receipt	Access receipt	
DELETE	/orders/{id}/receipt	Conclude the order process	6

Method	Resource type	Action	Step
POST	orders	Create new order	1
POST/PATCH	update	Update the order	2
DELETE	cancel	Cancel order	3
PUT	payment	Pay order	4
Barista preparing the order			
GET	order	Poll order state	5
GET	receipt	Access receipt	
DELETE	receipt	Conclude the order process	6

# Spring RESTBucks

	Orders	Payment
Web	Spring Data REST	Manual implementation
Service	-	Manual implementation
Repository	Spring Data	Spring Data



# JacksonCustomizations

Externalize tweaks to the  
general JSON design

Spring Data REST  
for the CRUDdy parts.

# ResourceProcessor

To conditionally sneak  
links into the default  
representation.

# Code

Spring RESTBucks - <https://github.com/olivergierke/spring-restbucks>

# Questions?