

# REST Assured

## Hypermedia APIs with Spring

Oliver Gierke



# Oliver Gierke

SpringSource Engineer  
Spring Data

- ✉ [ogierke@vmware.com](mailto:ogierke@vmware.com)
- 🌐 [www.olivergierke.de](http://www.olivergierke.de)
- 🐦 [@olivergierke](https://twitter.com/olivergierke)

# Background

REST

# REST

Resources

URIs

Uniform Interface

Representations

Hypermedia

# Hypermedia

||

HATEOAS - the  
word, there's no  
pronunciation for.

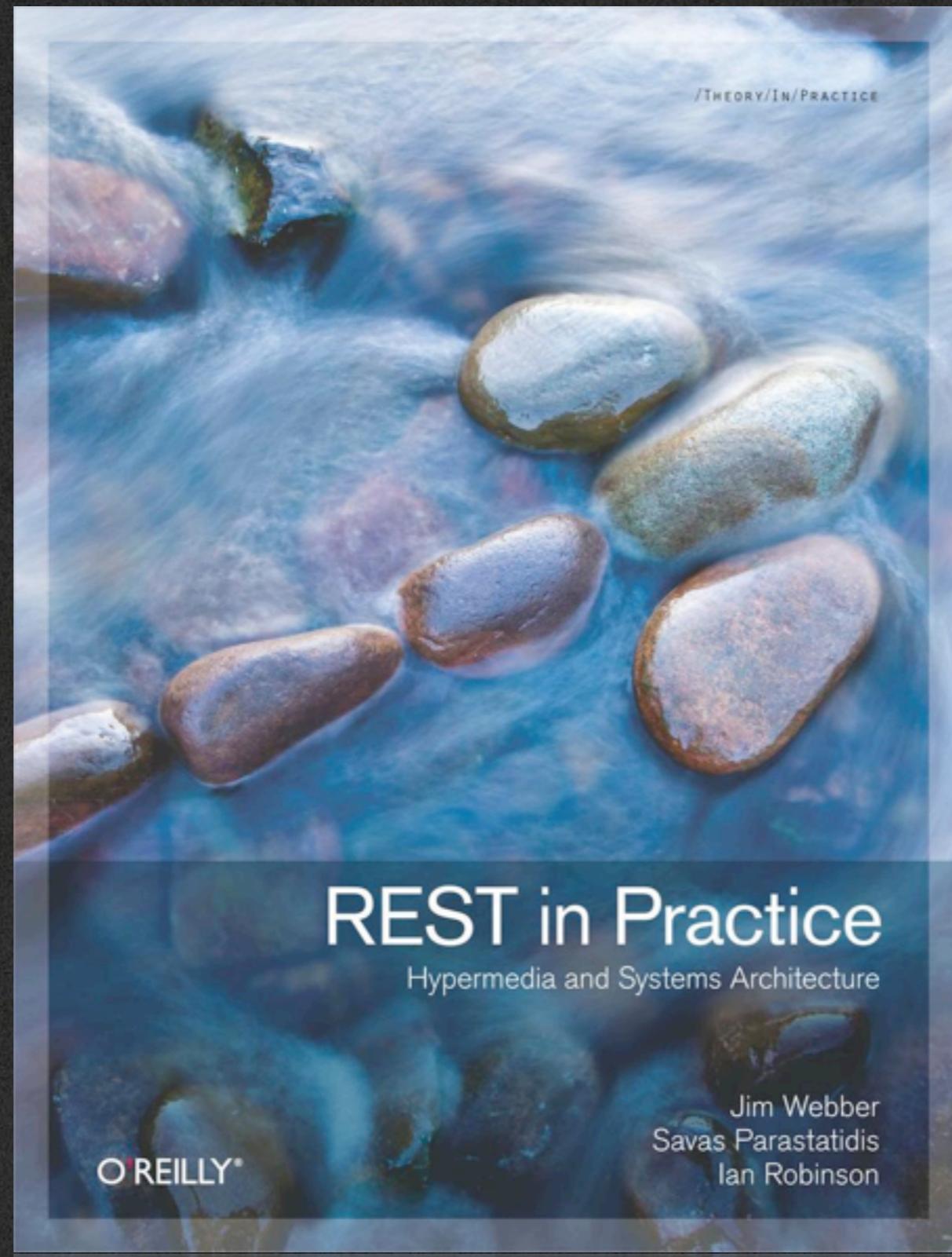
(Ben Hale, SpringOne2GX 2012)

# Hypertext

Links in representations

State navigations discoverable

REST in practice



O'REILLY

REST IN PRACTICE  
HYPERMEDIA AND SYSTEMS ARCHITECTURE  
Jim Webber, Savas Parastatidis, Ian Robinson

RESTBucks

# RESTBucks

Starbucks (like) coffee ordering

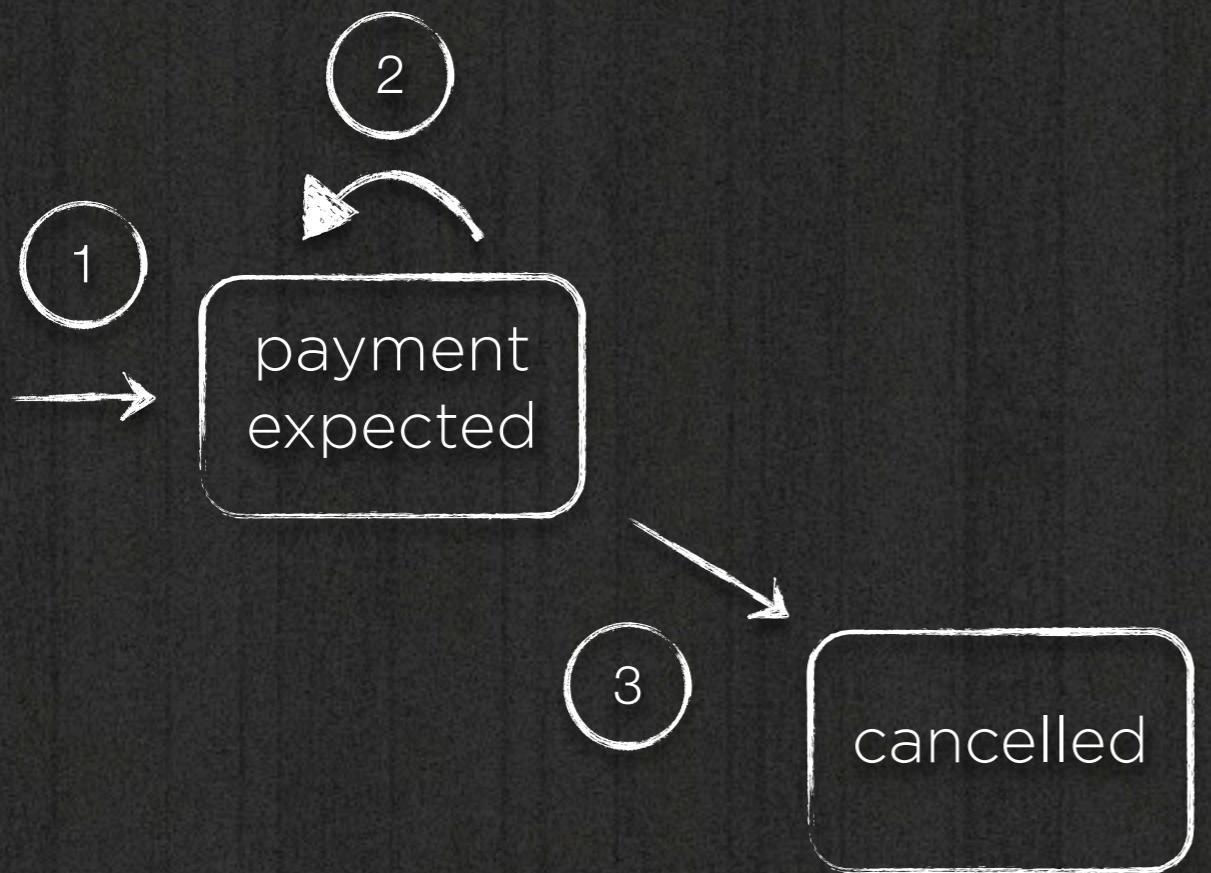
Order / Payment

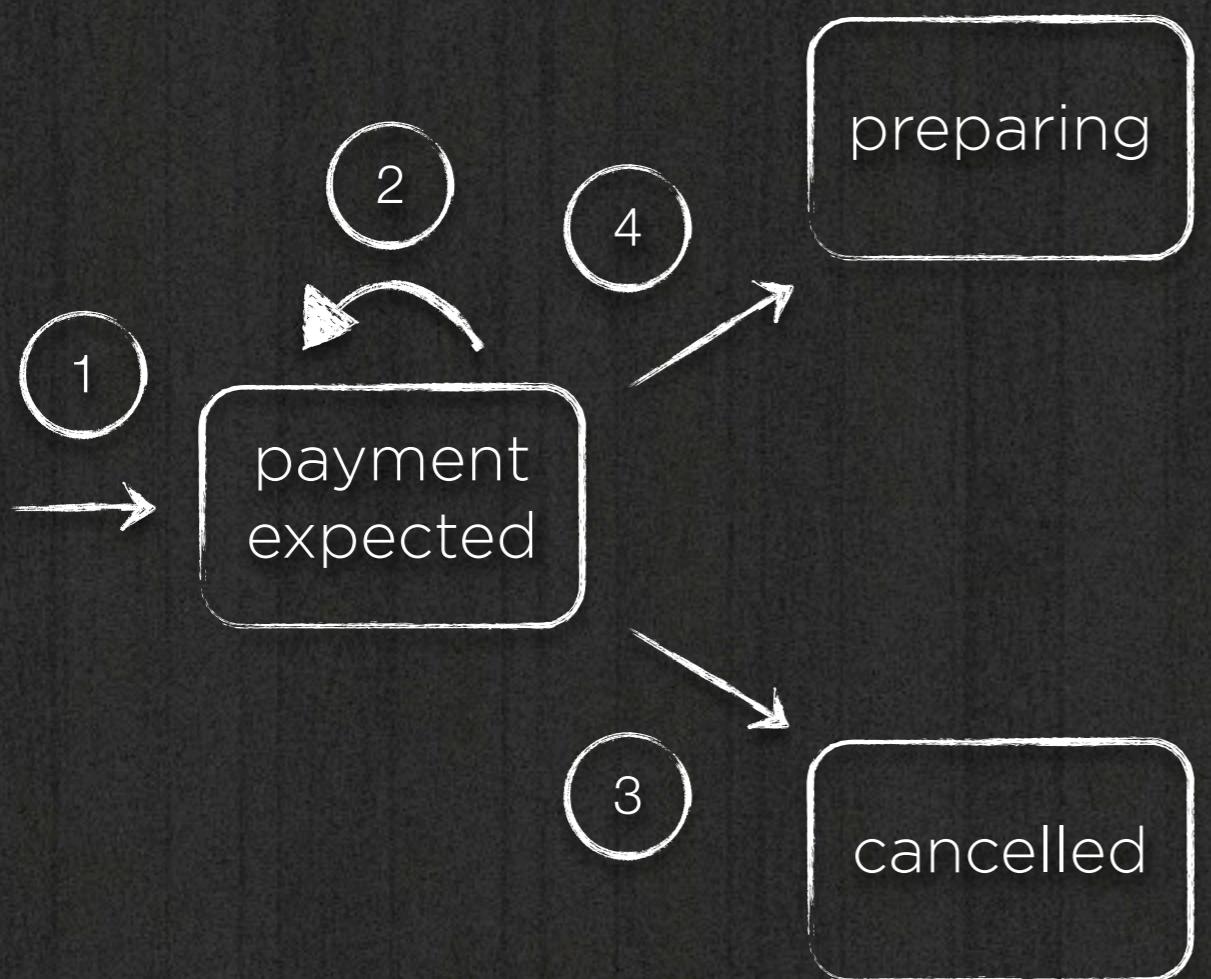


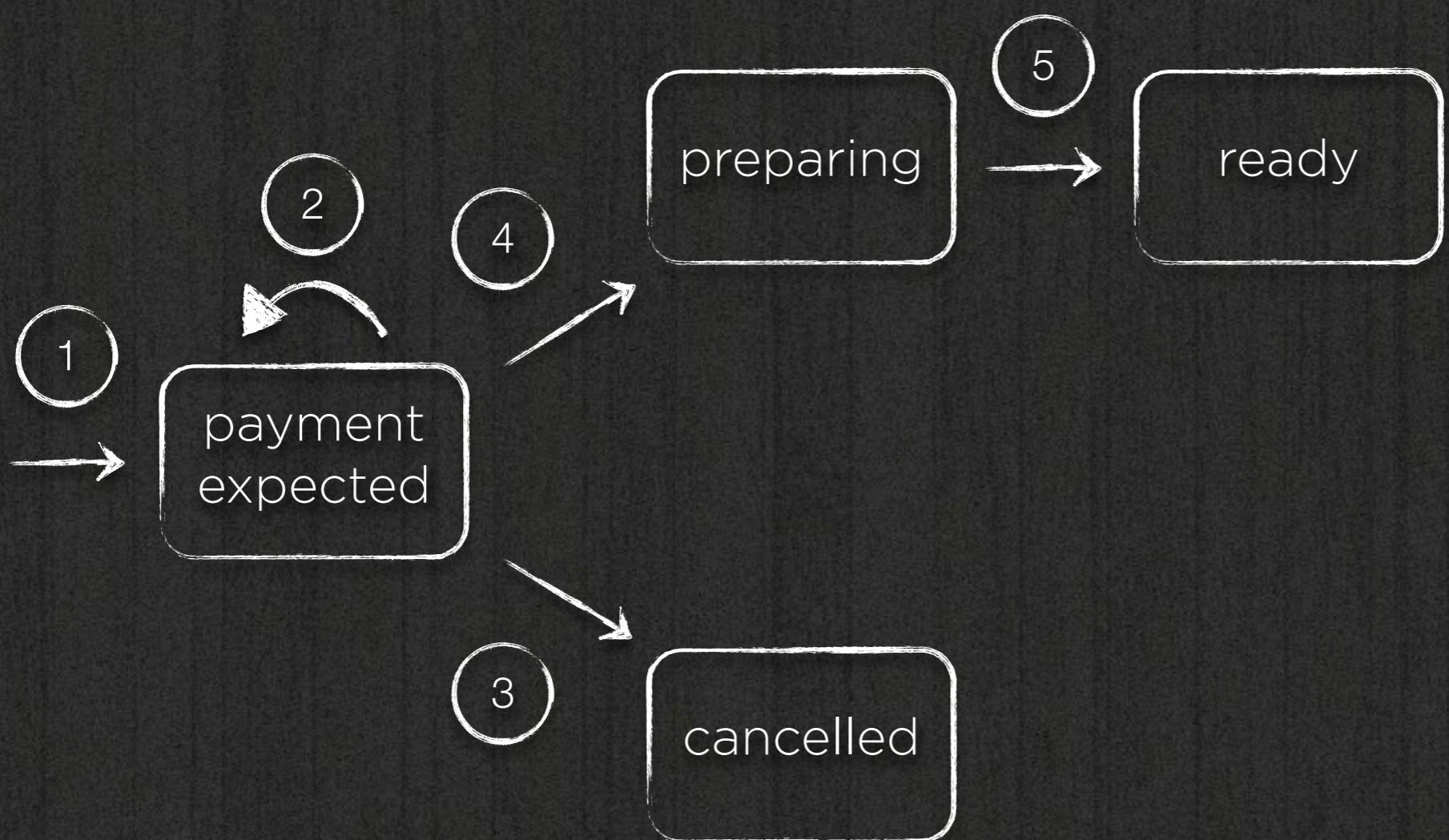
1

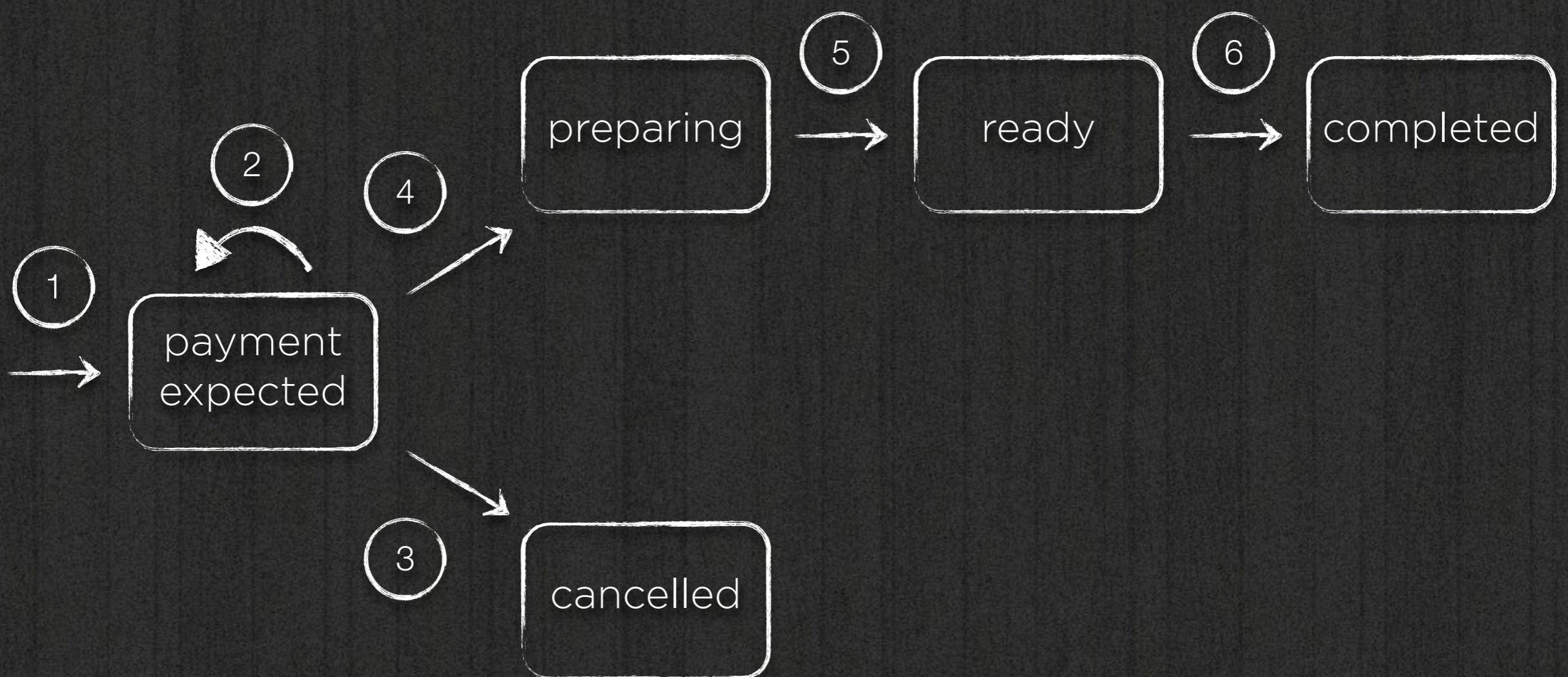
payment  
expected











Method	URI	Action	Step
<b>POST</b>	/orders	Create new order	1
<b>POST/PATCH</b>	/orders/4711	Update the order (only if "payment expected")	2
<b>DELETE</b>	/orders/4711	Cancel order (only if "payment expected")	3
<b>PUT</b>	/orders/4711/payment	Pay order	4
Barista preparing the order			
<b>GET</b>	/orders/4711	Poll order state	5
<b>GET</b>	/orders/4711/receipt	Access receipt	
<b>DELETE</b>	/orders/4711/receipt	Conclude the order process	6

# Challenges

# Challenges

How to avoid hard coding URLs?

Use link  
relations

orders	Returns all orders available in the system
order	Returns a single order
self	The uri value can be used to GET the latest resource representation of the order.
cancel	This is the URI to be used to DELETE the order resource should the consumer wish to cancel the order.
update	Consumers can change the order using a POST to transfer a representation to the linked resource.
payment	The linked resource allows the consumer to begin paying for an order. Initiating payment involves PUTting an appropriate resource representation to the specified URI.
receipt	The URI to access the receipt using GET and conclude the order by taking the receipt (use DELETE).

orders	Returns all orders available in the system
order	Returns a single order
self	The uri value can be used to GET the latest resource representation of the order.
cancel	This is the URI to be used to DELETE the order resource should the consumer wish to cancel the order.
update	Consumers can change the order using a POST to transfer a representation to the linked resource.
payment	The linked resource allows the consumer to begin paying for an order. Initiating payment involves PUTting an appropriate resource representation to the specified URI.
receipt	The URI to access the receipt using GET and conclude the order by taking the receipt (use DELETE).

Method	URI	Action	Step
POST	/orders	Create new order	1
POST/PATCH	/orders/4711	Update the order (only if "payment expected")	2
DELETE	/orders/4711	Cancel order (only if "payment expected")	3
PUT	/orders/4711/payment	Pay order	4
Barista preparing the order			
GET	/orders/4711	Poll order state	5
GET	/orders/4711/receipt	Access receipt	
DELETE	/orders/4711/receipt	Conclude the order process	6

Method	Relation type	Action	Step
POST	orders	Create new order	1
POST/PATCH	order	Update the order (only if "payment expected")	2
DELETE	order	Cancel order (only if "payment expected")	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

# Challenges

How to implement:  
"only if payment expected"?

Process  
modeling

# Root resource

The only URI known

GET /

```
{ links : [ { rel : "orders",
    href : ".../orders" } ]
}
```

# Place order

Access root resource

Follow orders link

```
$.links[?(@.rel="orders")].href
```

POST /orders

```
{ links : [ { rel : "orders",
    href : ".../orders" } ]
}
```

```
{ links : [ { rel : "self", href : ... },  
            { rel : "cancel", href : ... },  
            { rel : "update", href : ... },  
            { rel : "payment",  
              href : ".../orders/4711/payment" } ],  
  content : {  
    items : [ {  
      drink : "Cappucino",  
      size : "large",  
      milk : "semi"  
      price : 4.2  
    } ],  
    location : "take-away",  
    price : 4.2  
    status : "payment expected"  
  }  
}
```

# Trigger payment

Follow payment link

```
$.links[?(@.rel="payment")].href  
PUT /orders/4711/payment
```

```
{ links : [ { rel : "self",
              href : ".../orders/4711" } ,
             ...
             { rel : "payment",
               href : ".../orders/4711/payment" } ] ,
  content : {
    items : [ {
      drink : "Cappuccino",
      size : "large",
      milk : "semi"
      price : 4.2
    } ],
    location : "take-away",
    price : 4.2
    status : "payment expected"
  }
}
```

```
{ links : [ { rel : "self",
              href : ".../orders/4711/payment" } ,
              { rel : "order",
                href : ".../orders/4711" } ] ,
content : {
  creditCard : [ {
    number : "1234123412341234",
    cardHolder : "Oliver Gierke",
    expiryDate : "2013-11-01"
  } ],
  amount : {
    currency : "EUR",
    value : 4.2
  }
}
}
```

# Poll Order

Follow order link

```
$ .links[?(@.rel="order")].href
```

```
GET /orders/4711
```

```
ETag / If-None-Match
```

```
{ links : [ { rel : "self",
              href : ".../orders/4711/payment" } ,
              { rel : "order",
                href : ".../orders/4711" } ] ,
content : {
  creditCard : [ {
    number : "1234123412341234",
    cardHolder : "Oliver Gierke",
    expiryDate : "2013-11-01"
  } ],
  amount : {
    currency : "EUR",
    value : 4.2
  }
}
}
```

```
{ links : [ { rel : "self",
              href : ".../orders/4711" } ] ,
  content : {
    items : [ {
      drink : "Cappuccino",
      size : "large",
      milk : "semi"
      price : 4.2
    } ] ,
    location : "take-away",
    price : 4.2
    status : "preparing"
  }
}
```

# Poll Order

Use caching

ETag / If-None-Match

until...

```
{ links : [ { rel : "self",
              href : ".../orders/4711" } ,
              { rel : "receipt",
                href : ".../orders/4711/receipt" } ] ,

content : {
  items : [ {
    drink : "Cappucino",
    size : "large",
    milk : "semi"
    price : 4.2
  } ],
  location : "take-away",
  price : 4.2
  status : "ready"
}
}
```

# Access receipt

Follow receipt link

```
$.links[?(@.rel="receipt")].href
```

```
GET /orders/4711/receipt
```

```
{ links : [ { rel : "self",
              href : ".../orders/4711" } ,
              { rel : "receipt",
                href : ".../orders/4711/receipt" } ] ,

content : {
  items : [ {
    drink : "Cappucino",
    size : "large",
    milk : "semi"
    price : 4.2
  } ],
  location : "take-away",
  price : 4.2
  status : "ready"
}
}
```

# Conclude order

Follow receipt link

```
$.links[?(@.rel="receipt")].href
```

```
DELETE /orders/4711/receipt
```

# Hypermedia VS. Java Frameworks

	Spring MVC	JAX-RS
HTTP Methods	✓	✓
URI Mapping	✓	✓
Content negotiation	✓	✓
Hypermedia	?	?

Spring HATEOAS

# Spring HATEOAS

Representation models

LinkBuilder API

Representation enrichment

<http://bit.ly/spring-hateoas>

DEMO

Spring REST Bucks

# Spring REST Bucks

Sample implementation

Using Spring technologies

<http://bit.ly/spring-restbucks>

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



DEMO

# Miscellaneous

Spring MVC integration testing

REST Shell

Thank you!

# Resources

Code  
Spring HATEOAS Sample  
Spring RESTBucks  
Spring Data REST  
Spring HATEOAS  
REST Shell

# Books

REST in Practice

REST und HTTP

RESTful WebServices Cookbook

Spring Data

# Videos

Hypermedia APIs - Jon Moore

Hypermedia APIs with Spring