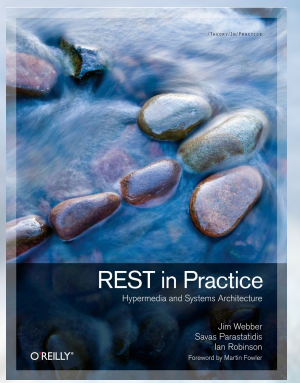
Hypermedia Design Walkthrough

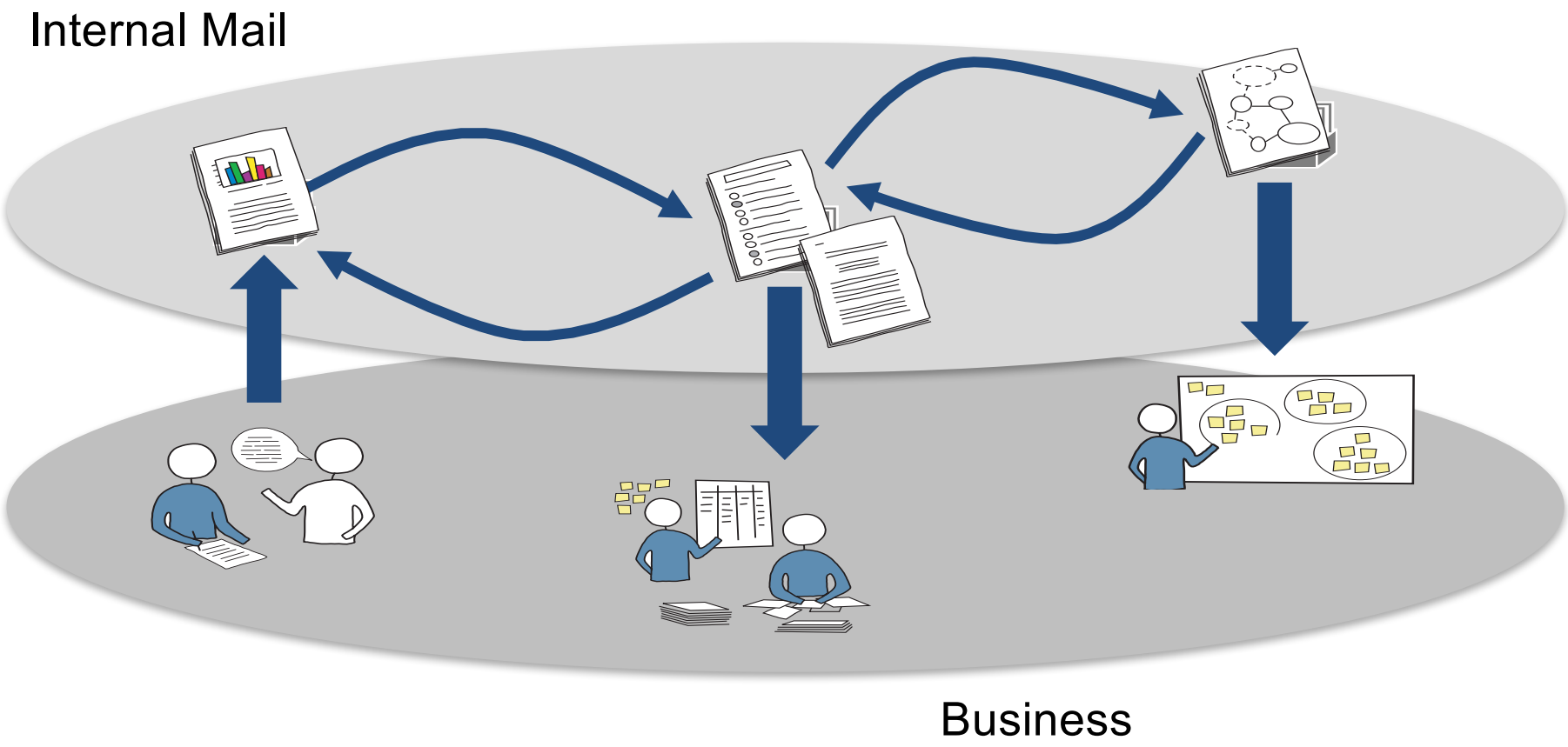


Two Questions

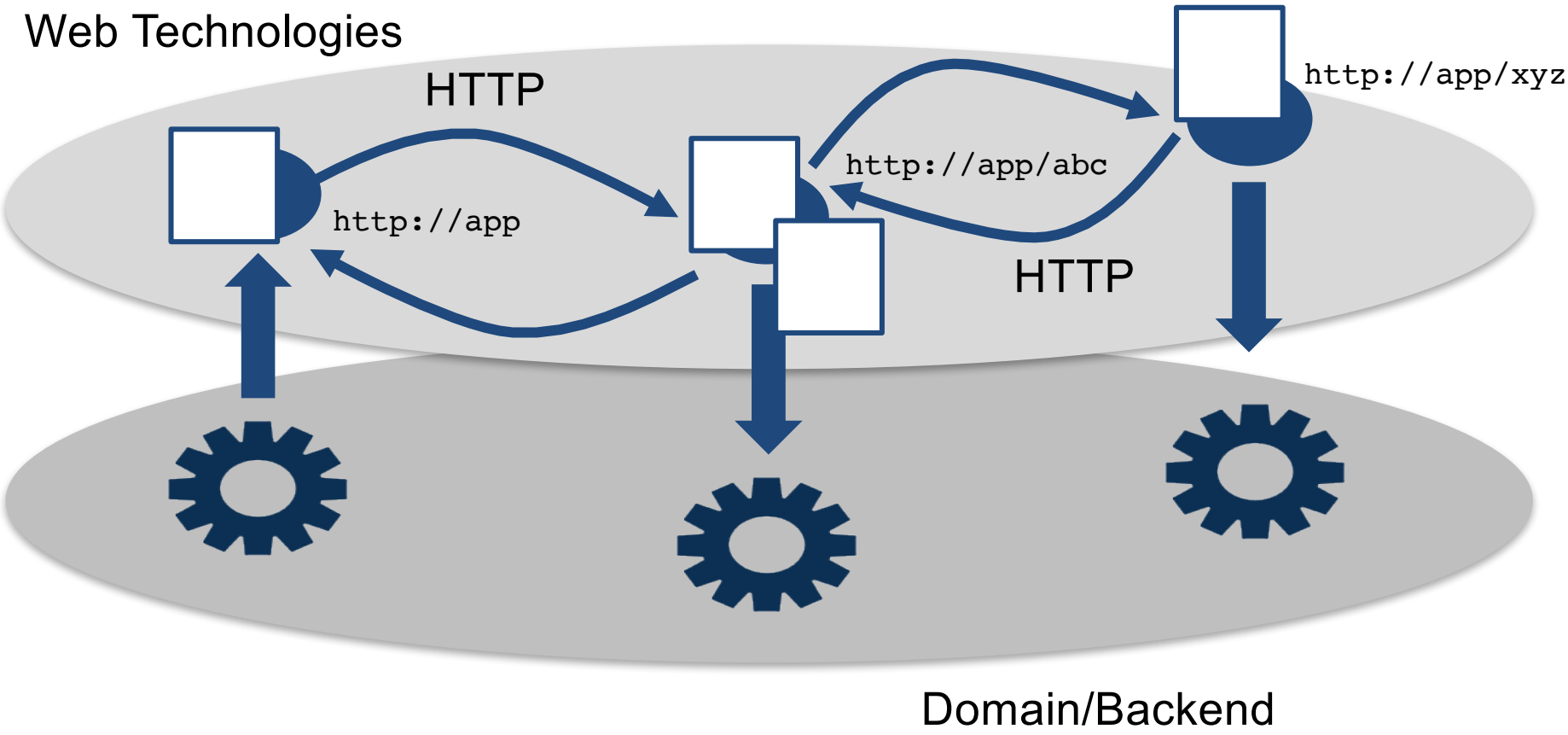
- What are the implications for API design if we say that "resources are manipulated through the exchange of representations" ?
  - What is the ***role*** of a resource?
- What are the constituent parts of an API?
  - What are the ***benefits*** of a Web API over, for example, a Java API?

The 1950s Office

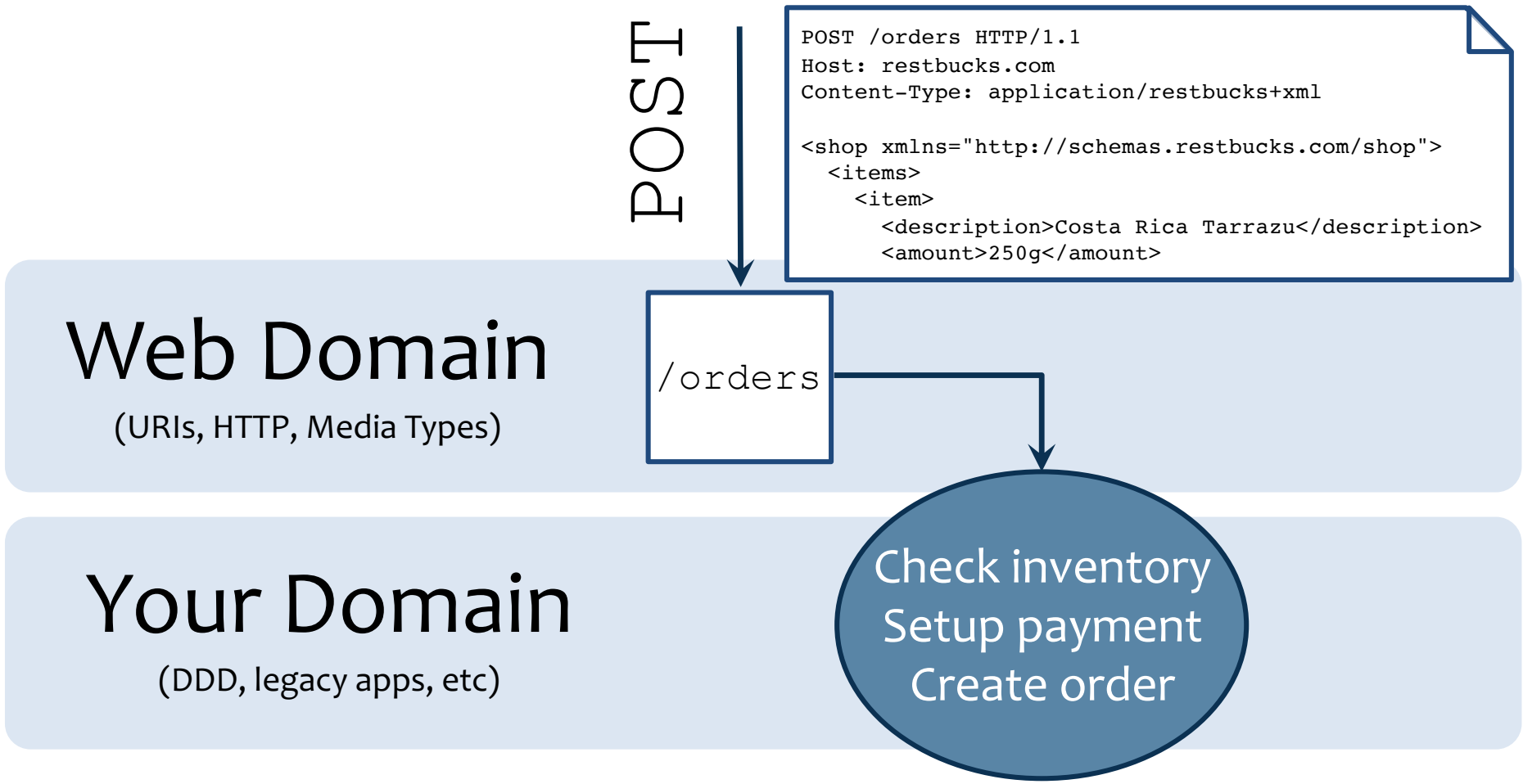
Getting Work Done Through the Exchange of Documents



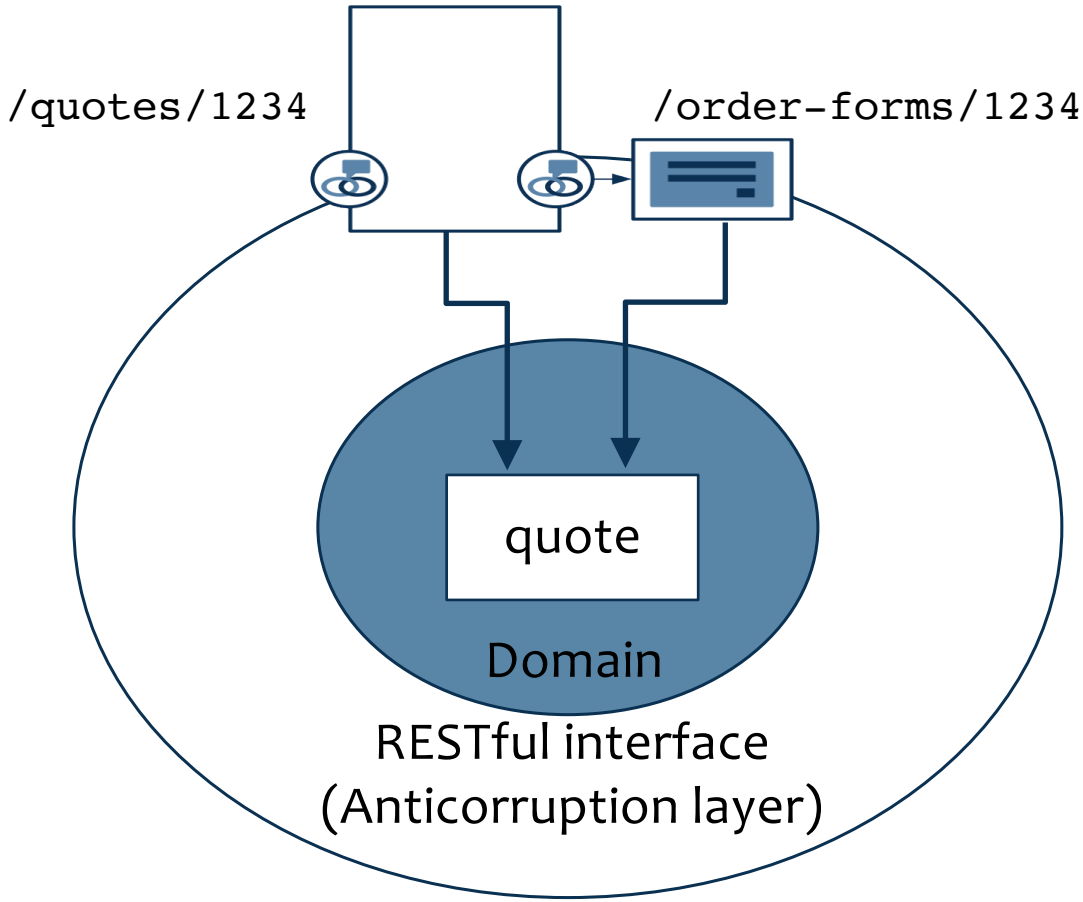
Manipulating Resources Through the Exchange of Representations



Work Transacted as a *Side-Effect* of Transferring Documents



Resources *Adapt* the Domain for Hypermedia Clients

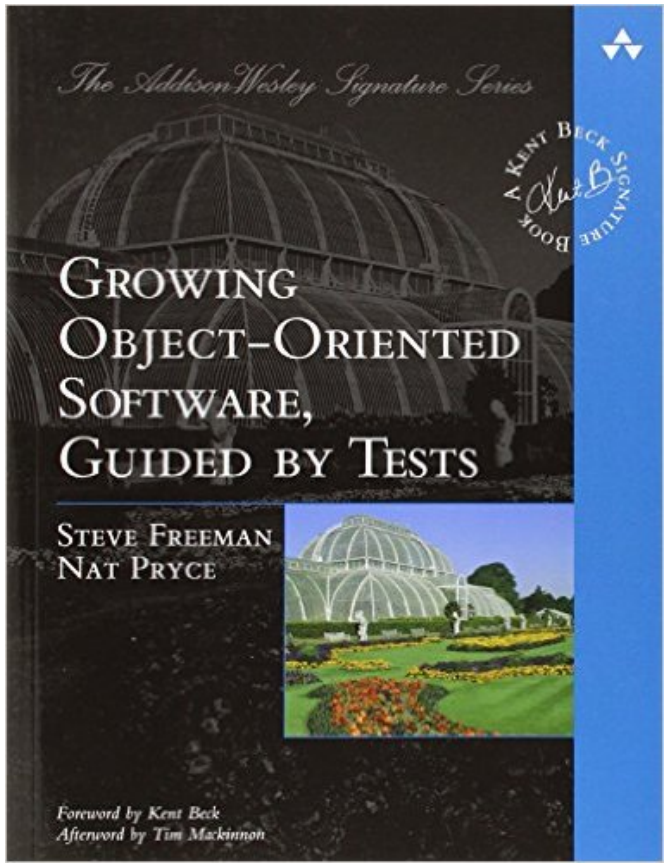


Resource Design Guidelines

- Resources don't necessarily map one-to-one onto domain entities
- Think in terms of ***use cases*** or ***capabilities***
  - Resources offer capabilities to the client
  - A small number of document exchanges serve to satisfy a use case
- Some representations *may* look like domain entities (e.g. *users*, *products*)
  - But some will look more like document-based forms (e.g. *mortgage application form*)

APIs: Interfaces + Communication Protocols

- **Interfaces** Define available methods and parameters (endpoints and messages)
  - Will 2 components ***fit*** together?
- **Communication Protocols** Describe relationships between interfaces
  - How do 2 components ***work*** together?
- Java has interfaces
  - But no notion of communication protocols
  - Supplement with TDD with mock objects to ***discover*** and ***document*** communication protocols
  - Fluent APIs



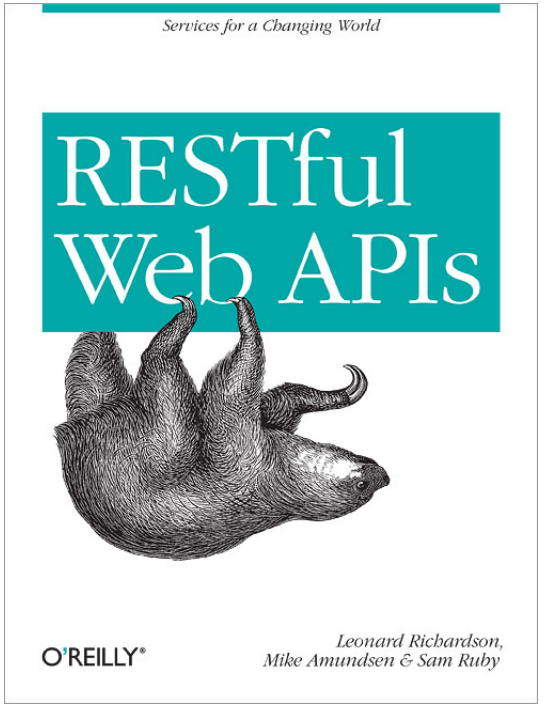
The Problem

- A good API answers two questions:
  - ***What*** components are available to the client (interfaces or endpoints and messages)?
  - ***How*** do these components work together to achieve an end-user goal?
- But descriptions of ***how*** risk breaking encapsulation and leaking implementation details
  - Clients end up tightly coupled to a specific implementation
- How do we resolve this tension?

Web APIs

A Web API describes:

- How to format resource representations
- How to use HTTP to transfer resource representations
- How to discover and interact with other resources



Web APIs

A Web API describes:

- How to format resource representations
- How to use HTTP to transfer resource representations
- How to discover and interact with other resources

Interfaces

Communication Protocols

Web APIs

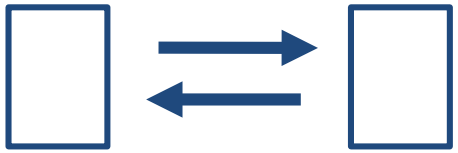
A Web API describes:

- How to format resource representations
- How to use HTTP to transfer resource representations
- How to discover and interact with other resources

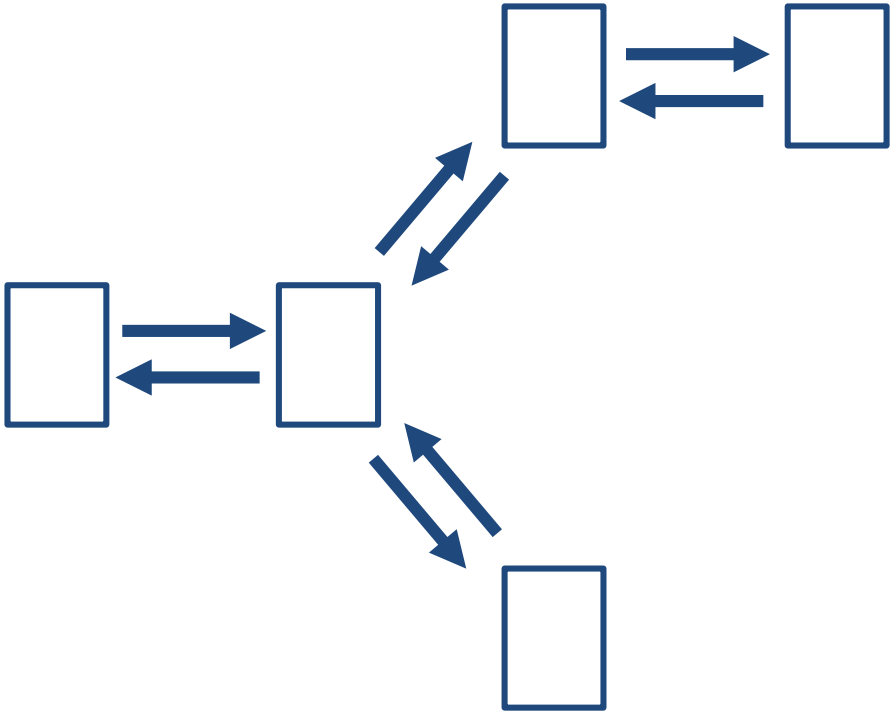
Interfaces	Media Types
	HTTP
Communication Protocols	Hypermedia (or documentation)

Protocols Coordinate Processes

HTTP

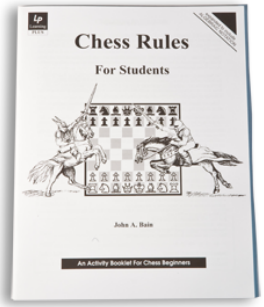


Communication Protocol (Domain Application Protocol)

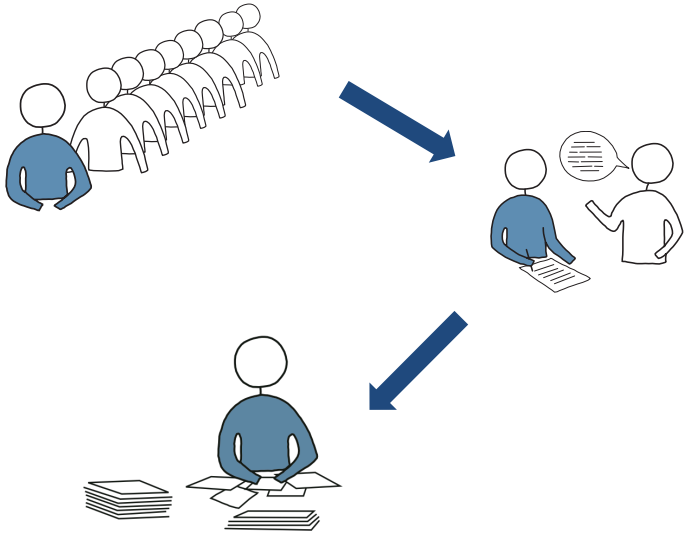


Two Approaches to Discovering and Executing Protocols

Specification



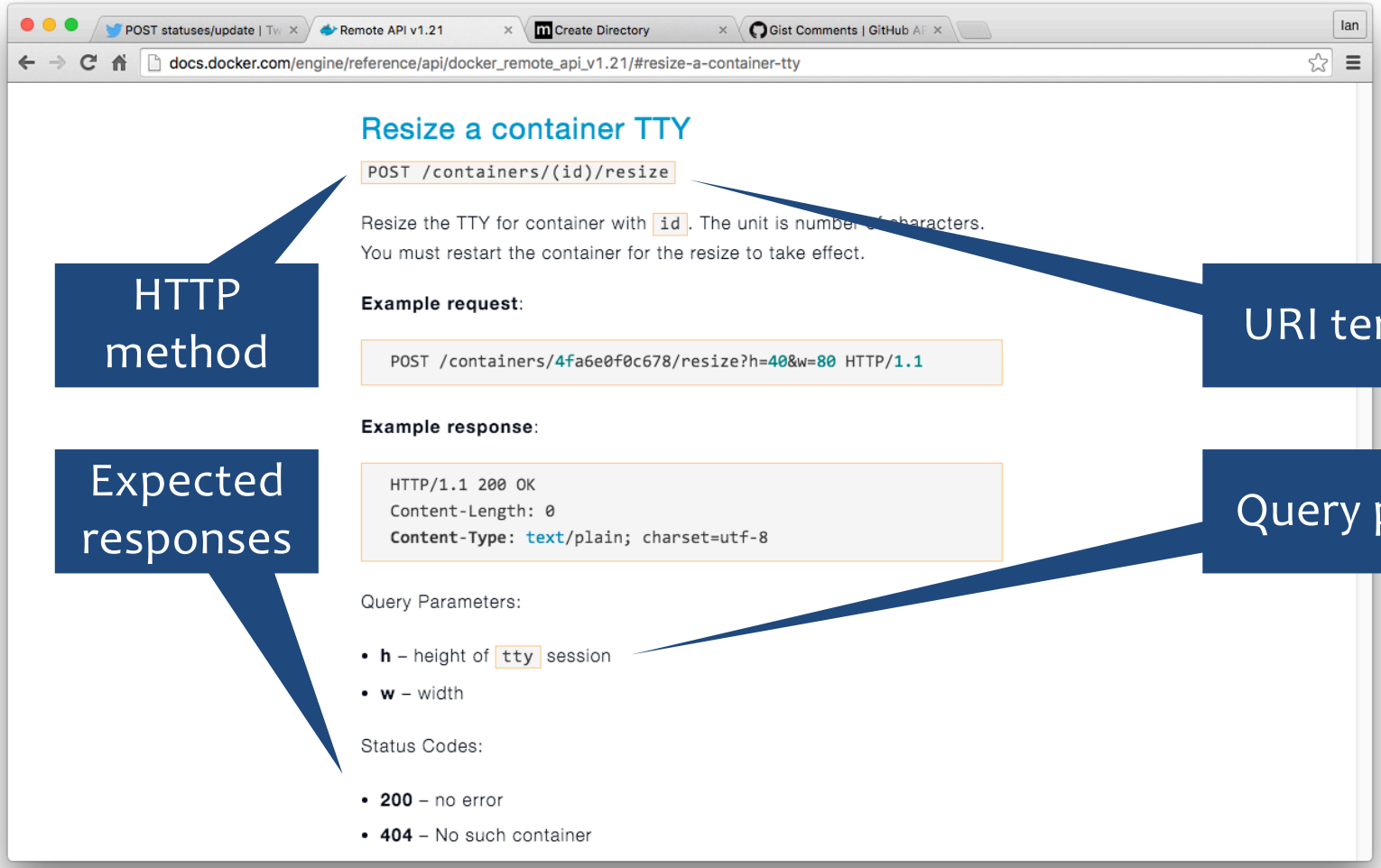
Discovery



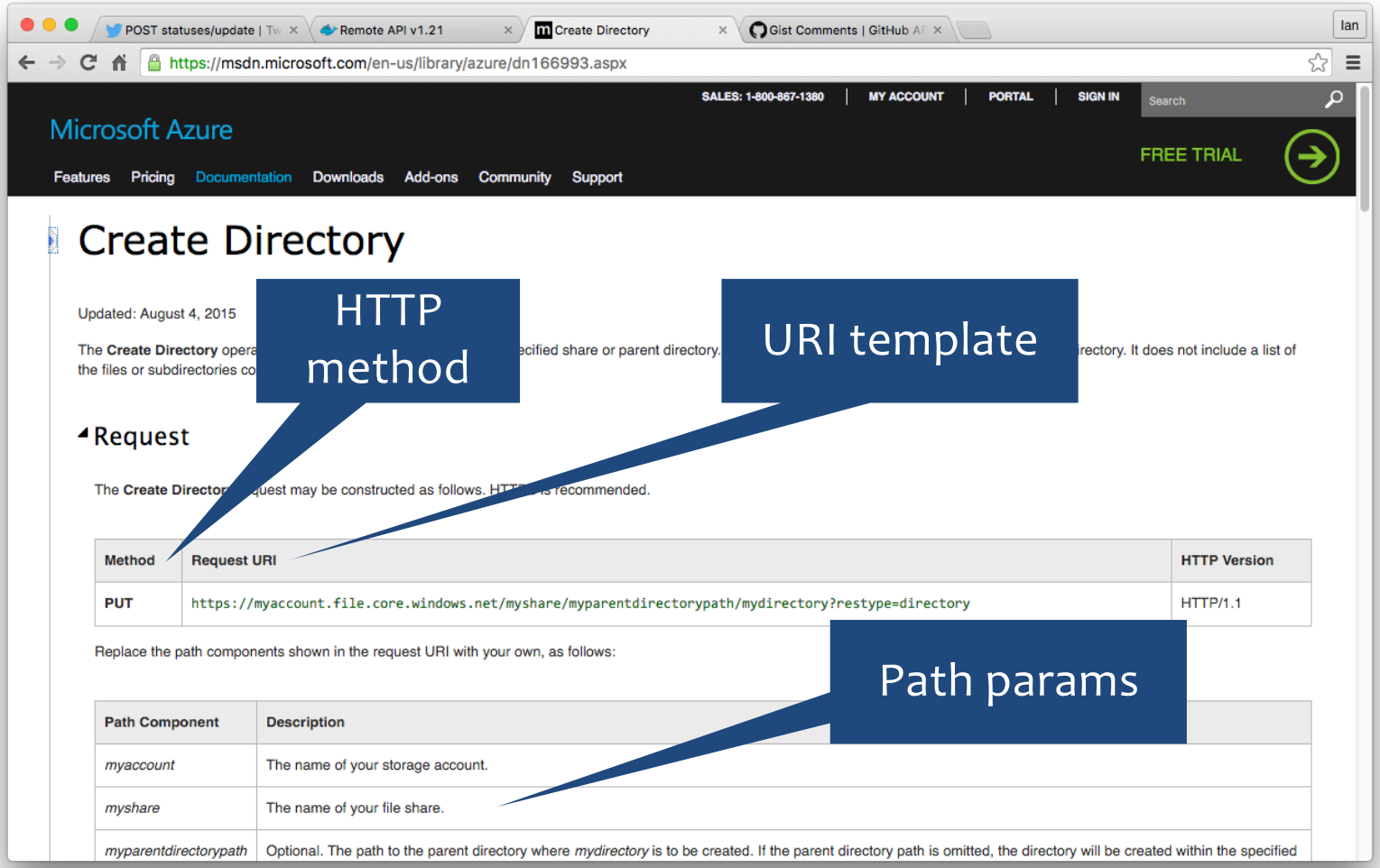
Protocol Specification

- Uses online documentation
- With examples
  - Request
  - Response
- Hackable URIs
  - URI templates
- Description languages
  - WADL, RAML, Swagger
  - Tool support

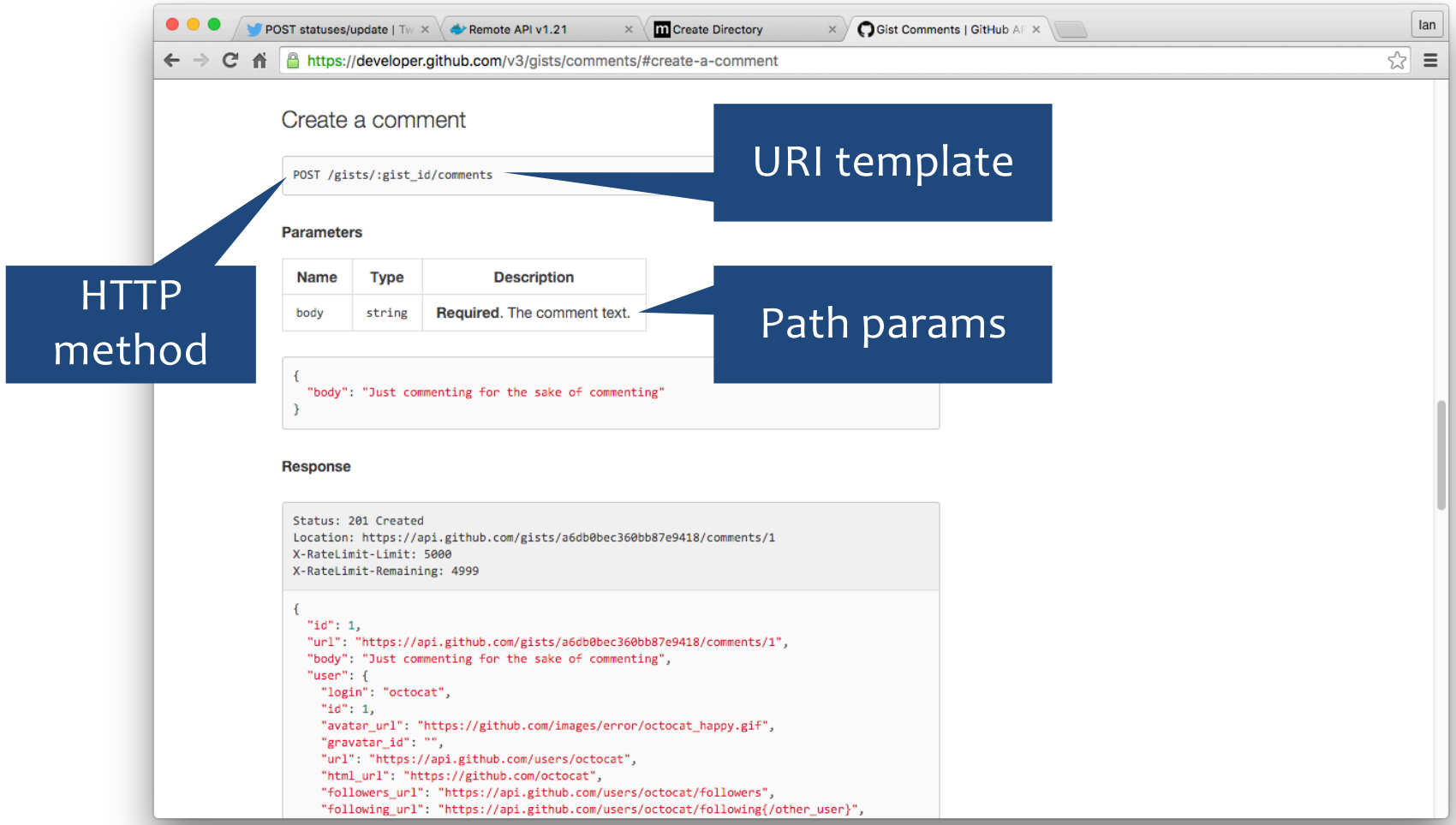
Docker Remote API



Microsoft Azure API



GitHub API

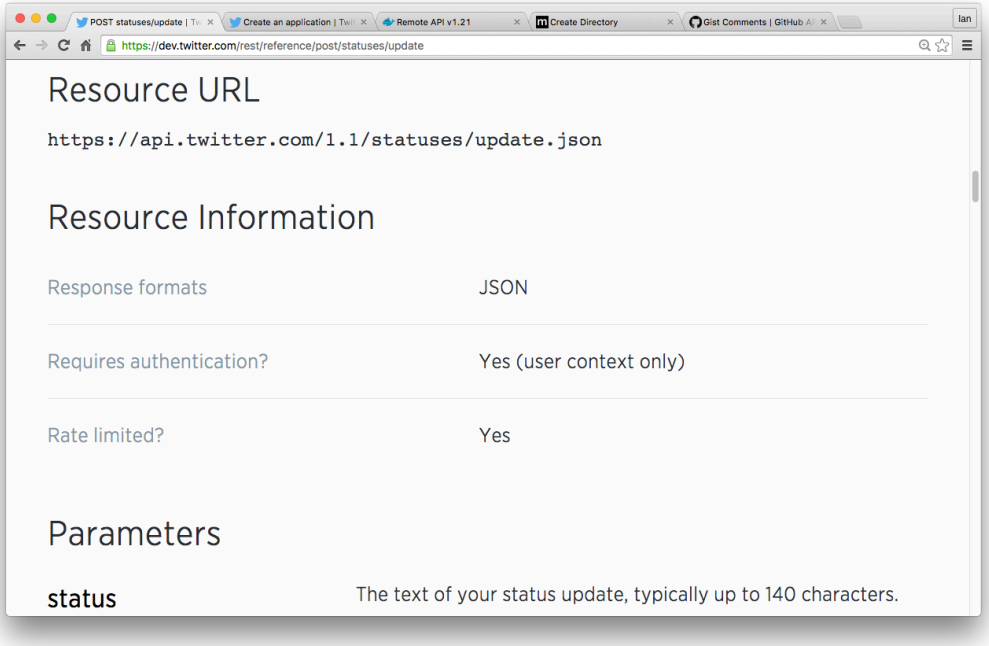

POST statuses/update | Tw
Remote API v1.21
Create Directory
Gist Comments | GitHub AP
Ian
https://developer.github.com/v3/gists/comments/#create-a-comment
Create a comment
POST /gists/:gist_id/comments
URI template
HTTP method
Parameters
Name
Type
Description
body
string
Required. The comment text.
Path params
{
"body": "Just commenting for the sake of commenting"
}
Response
Status: 201 Created
Location: https://api.github.com/gists/a6db0bec360bb87e9418/comments/1
X-RateLimit-Limit: 5000
X-RateLimit-Remaining: 4999
{
"id": 1,
"url": "https://api.github.com/gists/a6db0bec360bb87e9418/comments/1",
"body": "Just commenting for the sake of commenting",
"user": {
"login": "octocat",
"id": 1,
"avatar_url": "https://github.com/images/error/octocat_happy.gif",
"gravatar_id": "",
"url": "https://api.github.com/users/octocat",
"html_url": "https://github.com/octocat",
"followers_url": "https://api.github.com/users/octocat/followers",
"following_url": "https://api.github.com/users/octocat/following{/other_user}",

Pros and Cons

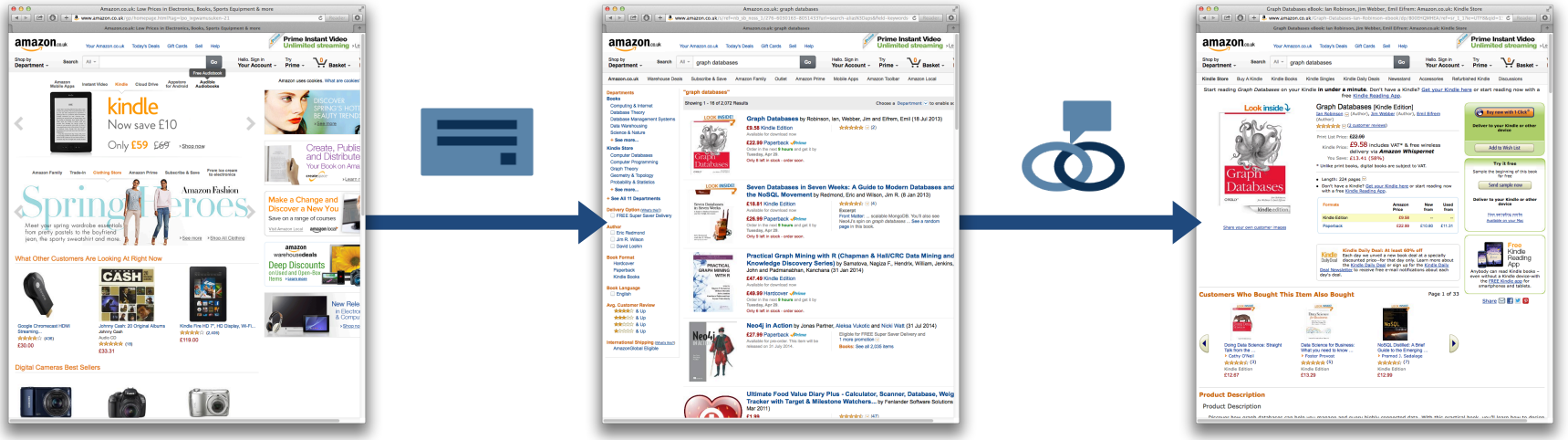
- Pros
  - Easy to understand
  - Description languages + tooling provide for auto-generated resource skeletons and client proxies
- Cons
  - Out-of-band communication
    - Potential for implementation-documentation drift
  - Clients tightly coupled to URI structure
    - Weak service encapsulation

Variant: RPC Idioms

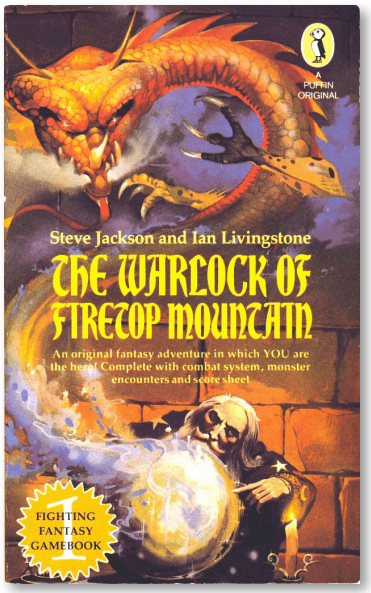
- "Network static functions"
- Single URI, many logical resources
  - Identity in header and/or payload
  - E.g Twitter
- **Reduces opportunities for caching, load-balancing, etc.**

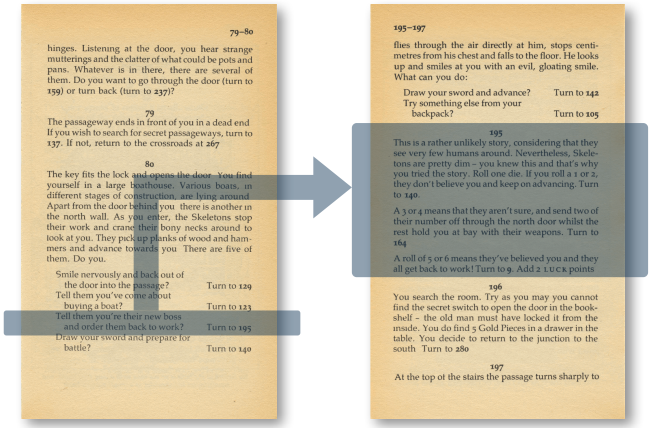


Protocol Discovery



Hypermedia Constraint





79–80

hinges. Listening at the door, you hear strange mutterings and the clatter of what could be pots and pans. Whatever is in there, there are several of them. Do you want to go through the door (turn to 159) or turn back (turn to 237)?

79

The passageway ends in front of you in a dead end If you wish to search for secret passageways, turn to 137. If not, return to the crossroads at 267

80

The key fits the lock and opens the door. You find yourself in a large boathouse. Various boats, in different stages of construction, are lying around Apart from the door behind you there is another in the north wall. As you enter, the Skeletons stop their work and crane their bony necks around to look at you. They pick up planks of wood and hammers and advance towards you There are five of them. Do you.

Smile nervously and back out of the door into the passage? Turn to 129
Tell them you've come about buying a boat? Turn to 123
Tell them you're their new boss and order them back to work? Turn to 195
Draw your sword and prepare for battle? Turn to 140

195–197

flies through the air directly at him, stops centimetres from his chest and falls to the floor. He looks up and smiles at you with an evil, gloating smile. What can you do:

Draw your sword and advance? Turn to 142
Try something else from your backpack? Turn to 105

195

This is a rather unlikely story, considering that they see very few humans around. Nevertheless, Skeletons are pretty dim – you knew this and that's why you tried the story. Roll one die. If you roll a 1 or 2, they don't believe you and keep on advancing. Turn to 140.

A 3 or 4 means that they aren't sure, and send two of their number off through the north door whilst the rest hold you at bay with their weapons. Turn to 164

A roll of 5 or 6 means they've believed you and they all get back to work! Turn to 9. Add 2 LUCK points

196

You search the room. Try as you may you cannot find the secret switch to open the door in the bookshelf – the old man must have locked it from the inside. You do find 5 Gold Pieces in a drawer in the table. You decide to return to the junction to the south Turn to 280

197

At the top of the stairs the passage turns sharply to

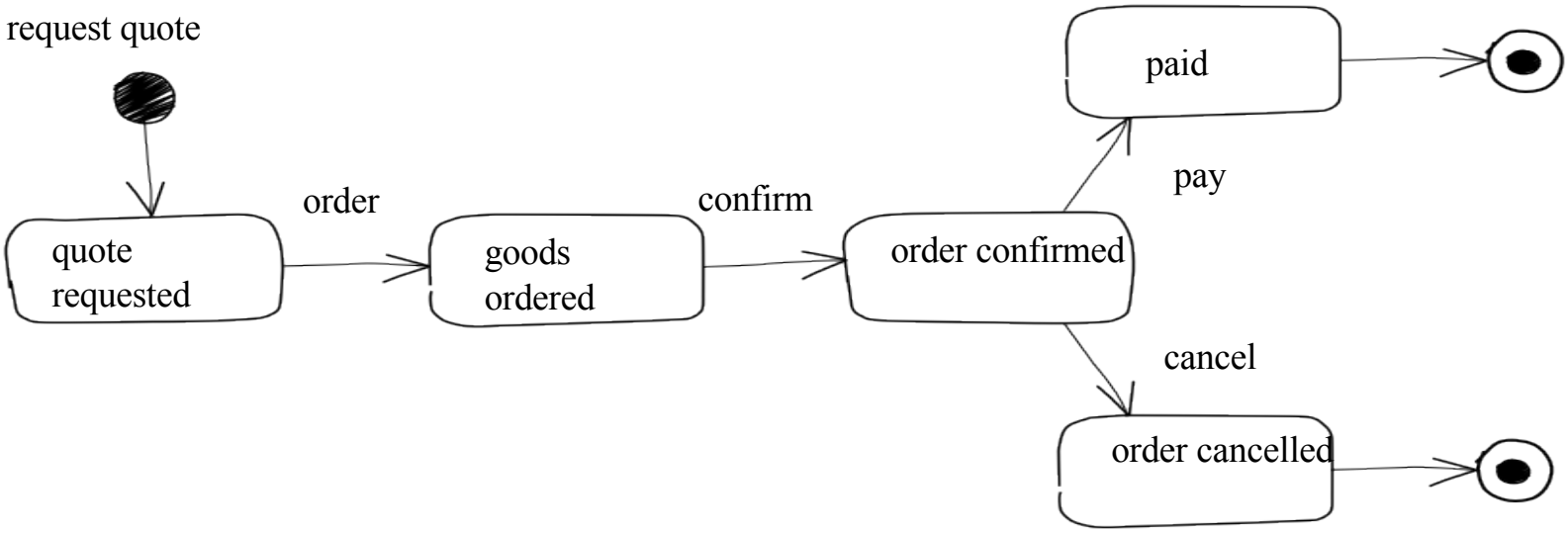
Hypermedia As The Engine Of Application State

State

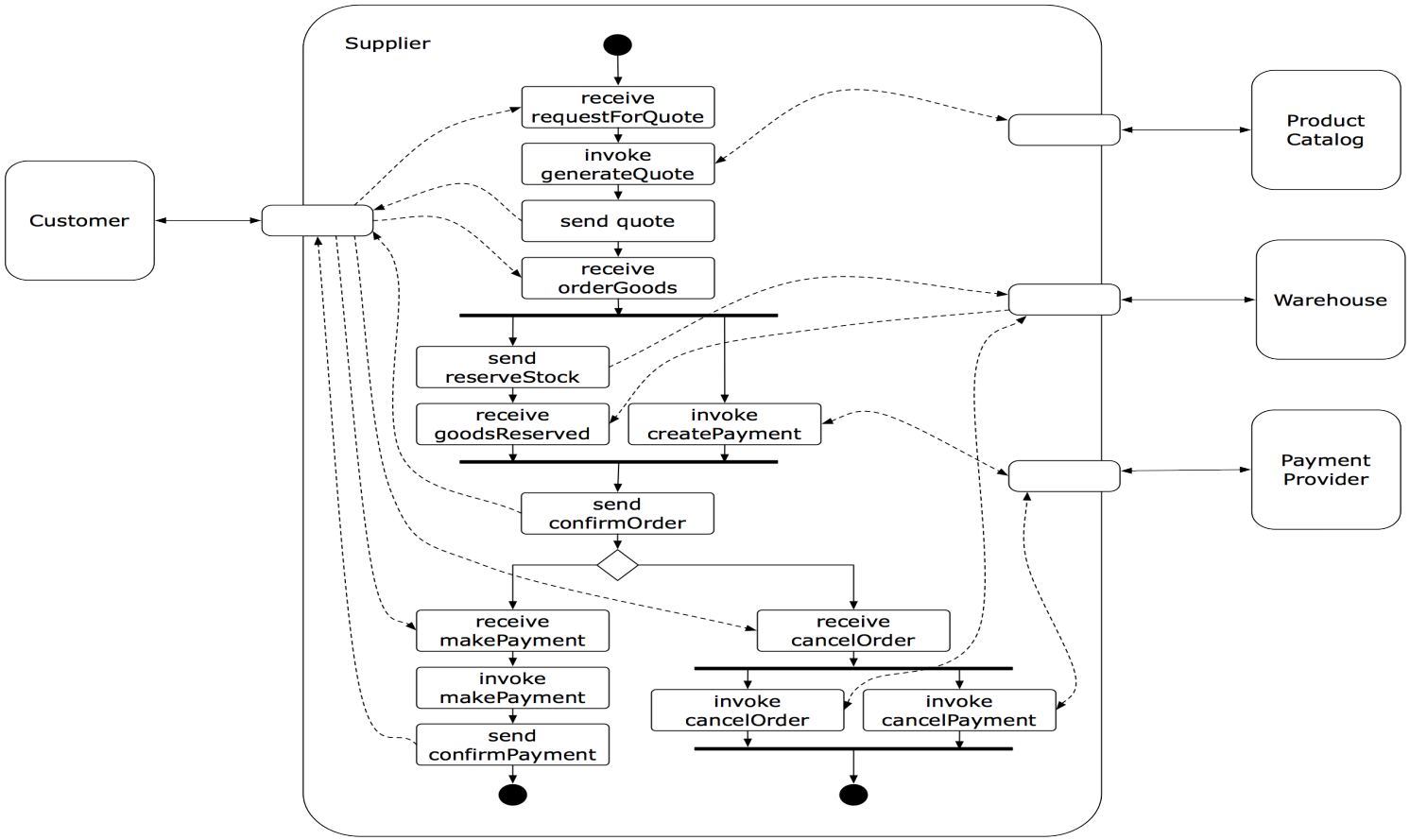
- ***Application state*** is a client-side concern
  - Progress towards an application goal
  - Lends integrity to sequence of actions
- ***Resource state*** is a server-side concern
  - Entity/domain state
  - Server governs resource lifecycle
  - May also be a function of state of associated resources because servers are also usually clients
    - Duncan Cragg, *Functional Observer REST*
    - *http://duncan-cragg.org/blog/post/forest-functional-observer-rest/*

Application State

Client applies a sequence of interactions to achieve an application goal

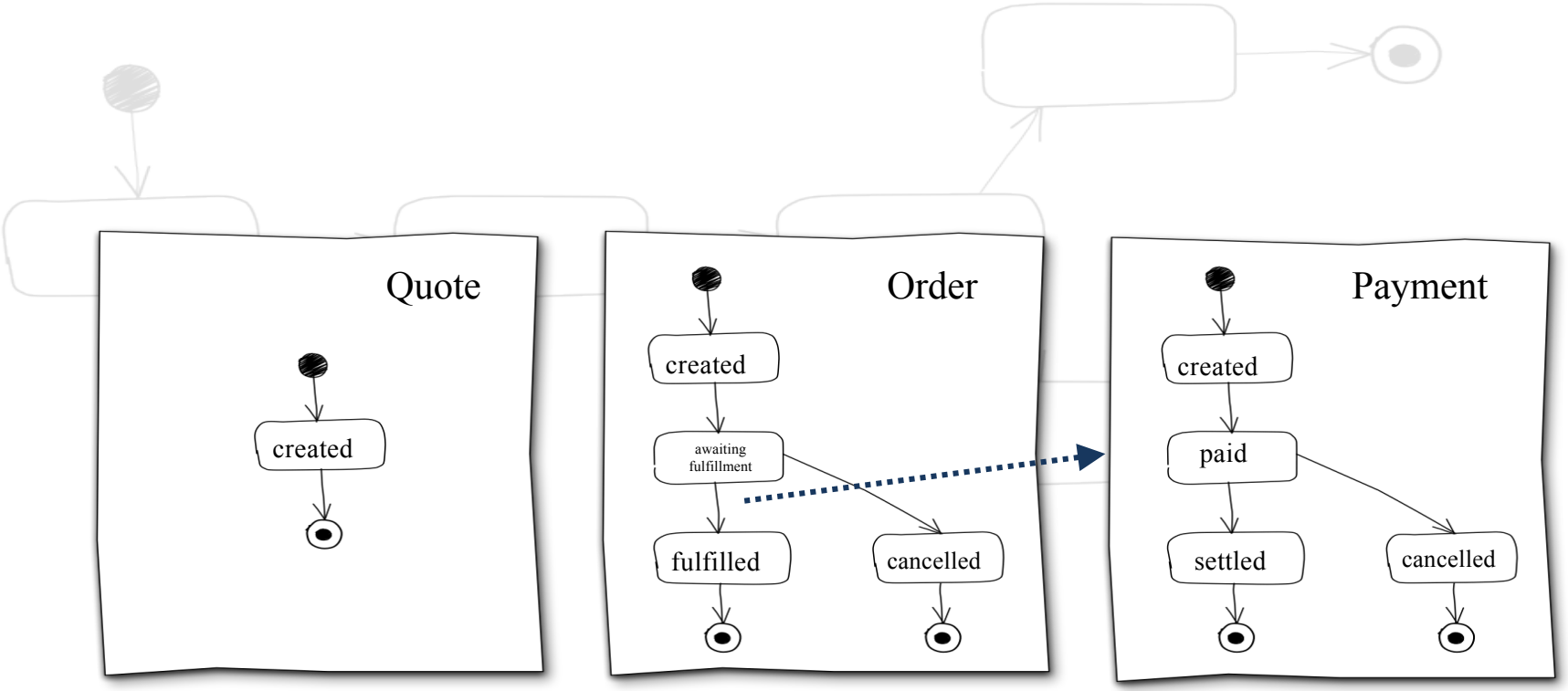


Application State Done Wrong



Resource State

Server governs access to resources, and controls their lifecycles



Use Status Codes to Coordinate Resource Lifecycle

- ***404 Not Found***
  - *No mapping from URI to resource state available*
- ***405 Method Not Allowed***
  - *Method not supported at this point in the resource's lifecycle*
- ***409 Conflict***
  - *Unable to accept the content of the request because of the current state of the resource*
  - *Method allowed, but state declared in request disallowed*
- ***503 Service Unavailable***
  - *Introduce backpressure*

Links and Forms Disclose State Transitions

- Links
  - Advertise other resources
    - Address
    - Semantics
      - For ***safe*** operations e.g. `GET`, `HEAD`, `OPTIONS`
- Forms
  - Advertise other resources
    - *And* how to manipulate them
  - For ***unsafe*** operations e.g. `PUT`, `POST`, `DELETE`

Anatomy of a Link

```
<link
  rel="rb:order"
  type="application/atom+xml"
  href="http://restbucks.com/orders/9876"/>
```

Anatomy of a Link

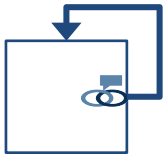
Semantic context
WHY access the linked resource?

Interpretation
HOW ought the response be interpreted/processed?

```
<link
  rel="rb:order"
  type="application/atom+xml"
  href="http://restbucks.com/orders/9876"/>
```

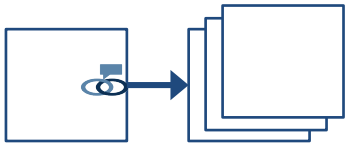
Address
WHERE does the linked resource reside?

Links Express a Resource's Knowledge of Other Resources



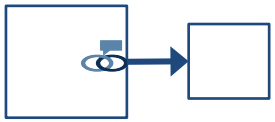
Self

An *order* links to *itself*



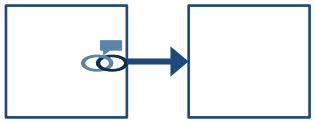
"Collections" of things

An *order* links to a collection of *orders*



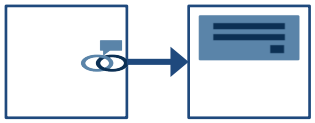
"Subordinate" resource

A *customer* links to an *address*



Associated resources

An *order* links to an associated *payment*



A form that provides some capability

Home page links to a *request-for-quote* form

Anatomy of a Form

```
<form name="input"
  method="post"
  action="/registration"
  enctype="application/x-www-form-urlencoded">
  <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

Anatomy of a Form

HTTP
HOW should the resource be manipulated?

URI
WHERE does the linked resource reside?

```
<form name="input"
  method="post"
  action="/registration"
  enctype="application/x-www-form-urlencoded">
  <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

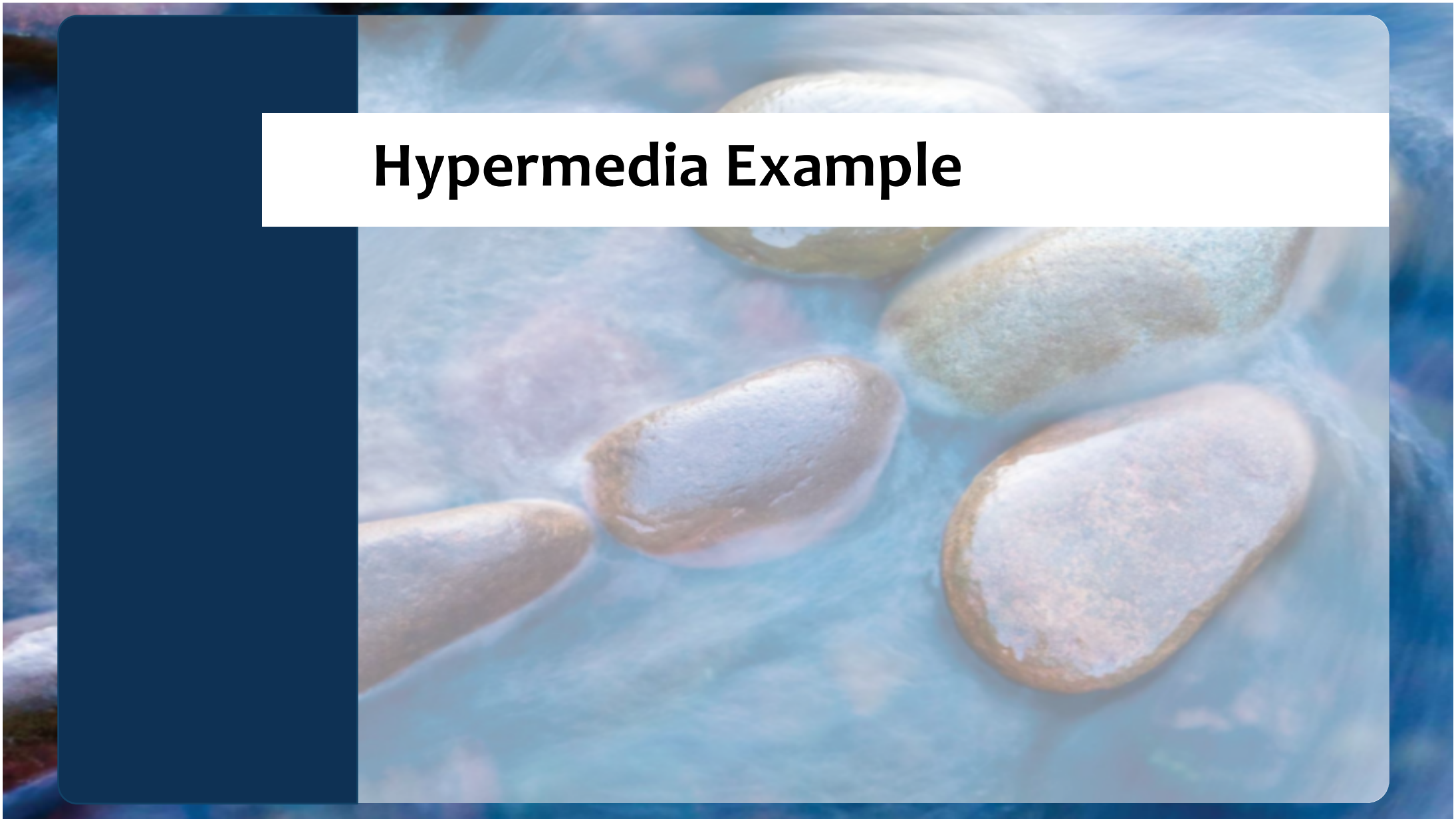
Media Type
WHAT format does a submitted representation take?

Forms

2 functions:

- Describe document to be submitted to resource
  - Form fields and/or schema
  - Prefilled forms
- ***Program*** the client
  - Target URI
  - HTTP method
  - Content-less `DELETE`

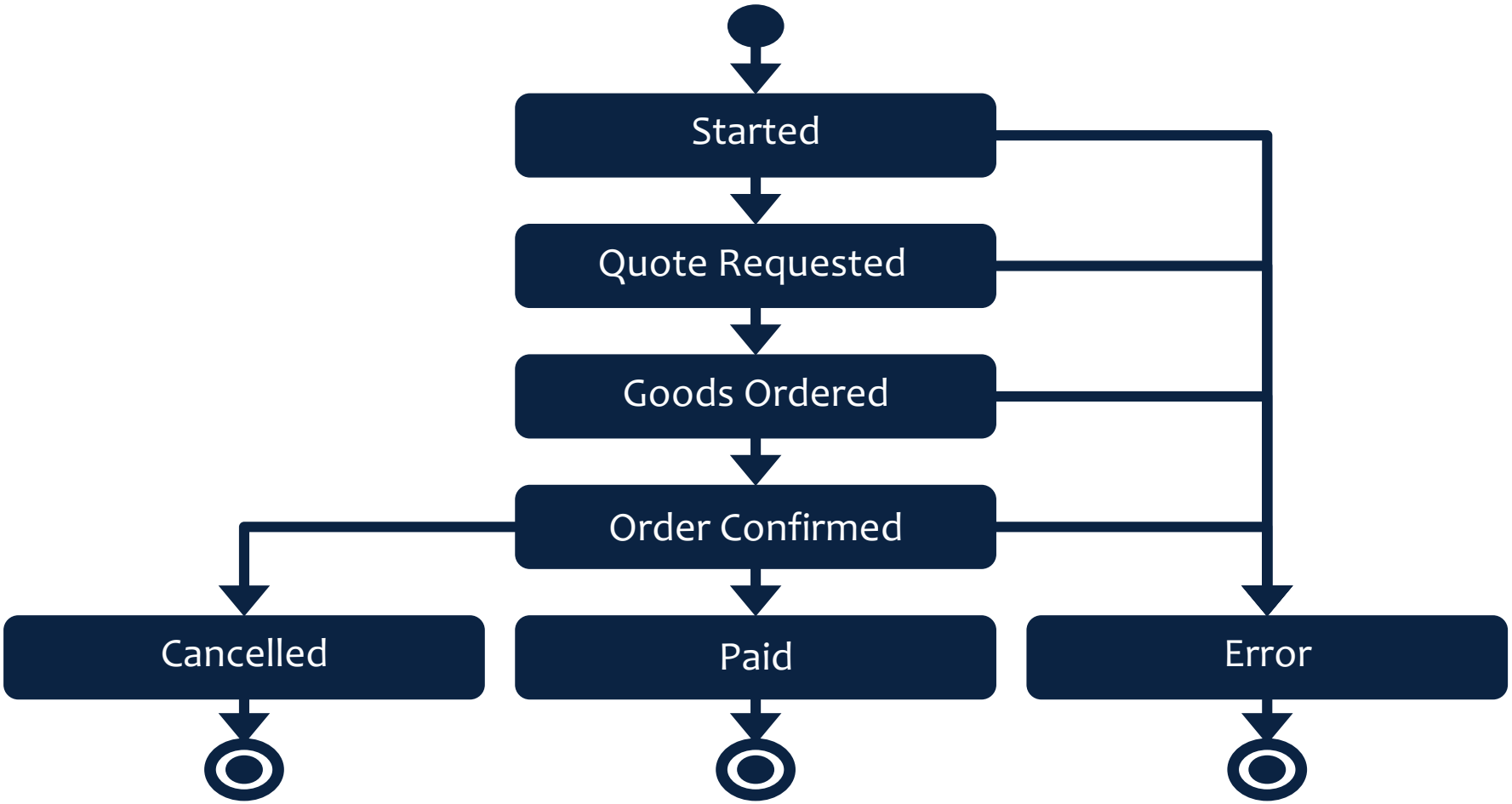
Hypermedia Example



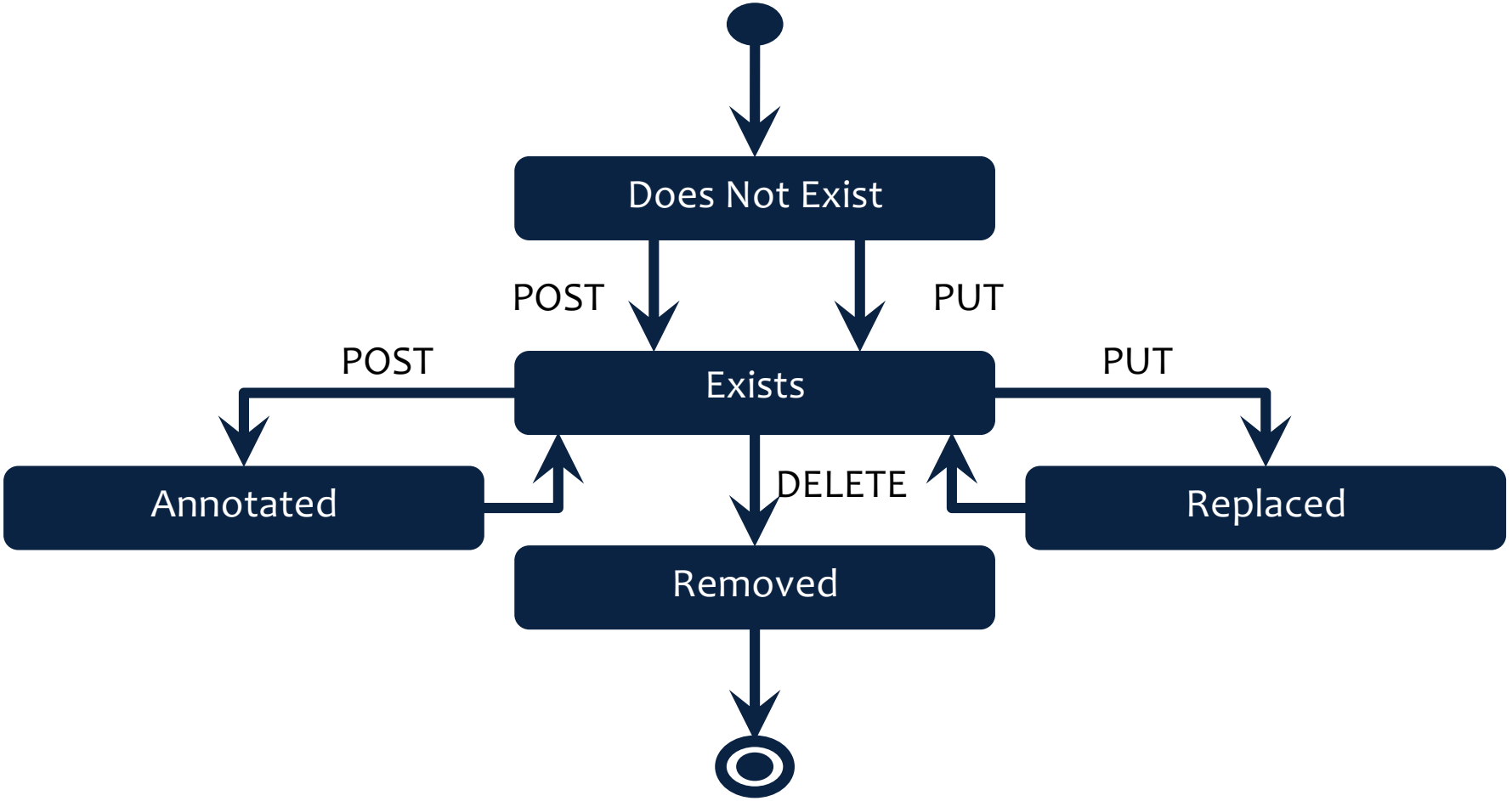
Clean APIs with Hypermedia

- A clean API scales not only with users but also over time
- This means it has clean affordances that allow third parties to evolve at their own pace as the API evolves
- This is very important when you consider the web of microservices that will make up our system
  - Services change, evolve, get re-written, retired
  - Clients don't (have to) change at the same rate

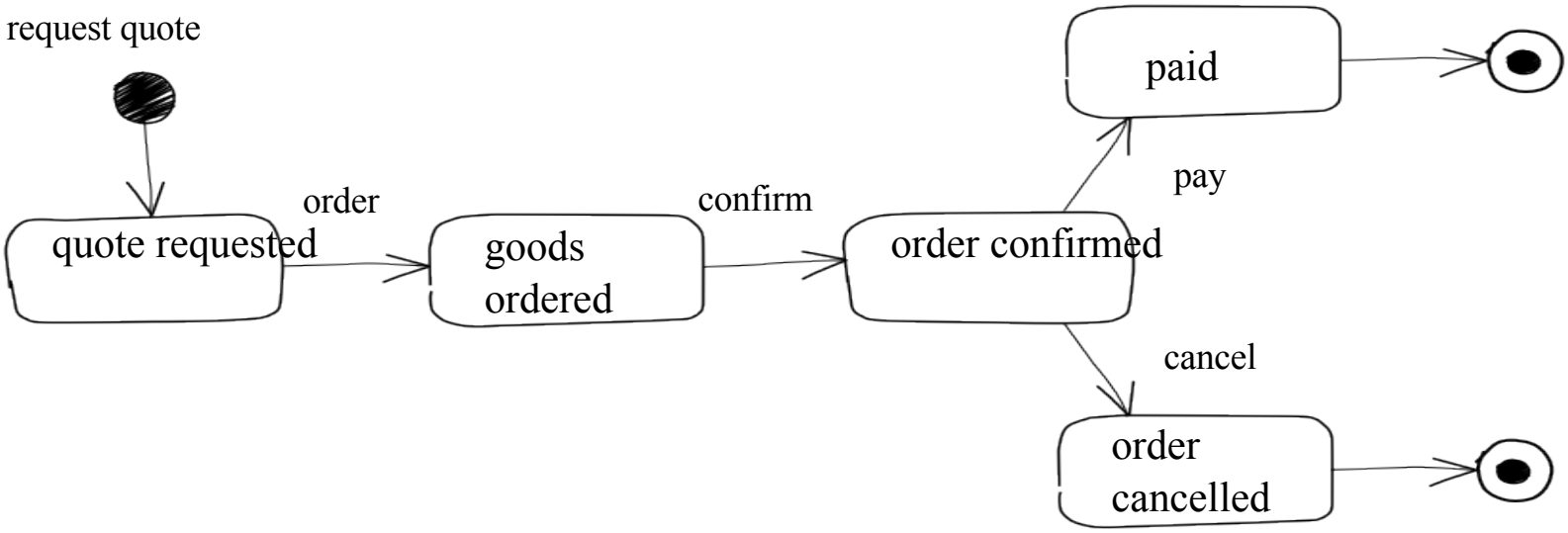
This is What We Want to Achieve



This is how the Web works



Procurement Application



System Design

- Three capabilities ➡ three services:
  - Quoting
  - Fulfilment
  - Payment
- Several media types:
  - Collection+JSON
  - Atom
  - XHTML
- Hypermedia to disclose application protocol

Application Start

Request

```
GET /shop HTTP/1.1
Host: quoting.restbucks.com
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 26 Jul 2015 10:00:00 GMT
Cache-Control: public, max-age=86400
Content-Type: application/vnd.collection+json

{
    "collection": {
        "href": "http://quoting.restbucks.com/shop",
        "links": [
            { "href": "/rfq", "rel": "request-for-quote" }
        ]
    }
}
```

Design strategies

Entry point

Advertise capabilities

Caching

Store infrequently changing representations closer to client

Get Request-For-Quote Form

Request

```
GET /rfq HTTP/1.1
Host: quoting.restbucks.com
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 26 Jul 2015 10:00:01 GMT
Cache-Control: public, max-age=86400
Content-Type: application/vnd.collection+json

{
    "collection": {
        "href": "http://quoting.restbucks.com/rfq",
        "template": {
            "data": [
                { "name": "description" },
                { "name": "amount" }
            ]
        }
    }
}
```

Submit Request-For-Quote Form

Request

```
POST /rfq HTTP/1.1
Host: quoting.restbucks.com
Content-Type: application/vnd.collection+json

{
    "template": {
        "data": [
            { "name": "description", "value": "Costa Rica Tarrazu" },
            { "name": "amount", "value": "250g" }
        ]
    }
}
```

Request-For-Quote Created

Response

```
HTTP/1.1 201 Created
Date: Mon, 26 Jul 2015 10:00:02 GMT
Location: http://restbucks.com/rfq/1234
```

Request

```
GET /rfq/1234 HTTP/1.1
Host: quoting.restbucks.com
```

Request-For-Quote

Response

```
HTTP/1.1 200 OK
Date: Mon, 26 Jul 2015 10:00:03 GMT
Content-Type: application/vnd.collection+json

{
    "collection": {
        "href": "http://quoting.restbucks.com/rfq/1234",
        "links": [ { "href": "/quote/1234", "rel": "quote" } ],
        "items": [
            {
                "href": "/rfq/1234/1",
                "data": [
                    { "name": "description", "value": "Costa Rica Tarrazu" },
                    { "name": "amount", "value": "250g" }
                ]
            }
        ],
        "template": {
            "data": [
                { "name": "description" },
                { "name": "amount" }
            ]
        }
    }
}
```

Design Strategies

Typed link to form

Link relation describes meaning of form in context of current representation

Inline control data

Form provides inline control data – HTTP idioms

Collection

POST form to collection resource

Navigate to Quote

Request

```
GET /quote/1234
Host: quoting.restbucks.com
```

Return Quote

Response

```
HTTP/1.1 200 OK
Cache-Control: public
Date: Mon, 26 Jul 2015 10:00:06 GMT
Expires: Mon, 02 Aug 2015 10:00:06 GMT
Content-Type: application/vnd.collection+json

{
    "collection": {
        "href": "http://quoting.restbucks.com/quote/1234",
        "items": [
            {
                "href": "/quote/1234/1",
                "data": [ { "name": "description", "value": "Costa Rica Tarrazu" },
                          { "name": "amount", "value": "250g" },
                          { "name": "cost", "value": "3.00" } ]
            } ],
        "commands": [
            {
                "href": "http://fulfillment.restbucks.com/order",
                "rel": "order",
                "data": [ { "name": "quote", "value": "http://quoting.restbucks.com/quote/1234" },
                          { "name": "checksum", "value": "c=99fe97e1&s=k2awEpcNmgDg8AyUo%3D" } ]
            } ]
    }
}
```

Submit Order

Request

```
POST /order HTTP/1.1
Host: fulfillment.restbucks.com
Content-Type: application/json
Content-Length: ...

{
    "quote": "http://quoting.restbucks.com/quote/1234",
    "checksum": "c=99fe97e1&s=k2awEpciJkd2X8rt3NmgDg8AyUo%3D"
}
```

Response

```
HTTP/1.1 202 Accepted
Cache-Control: no-store
Date: Mon, 26 Jul 2015 10:00:07 GMT
Location: http://fulfillment.restbucks.com/order/9876
```

Design Strategies

Prefilled form

Alternate representation of quote data, adapted for submitting an order

202 Accepted

Long-running operations
Successfully accepted request but result is not immediately available

Check State of Order

Request

```
GET /order/9876 HTTP/1.1
Host: fulfilment.restbucks.com
```

Response

```
HTTP/1.1 202 Accepted
Cache-Control: no-store
Date: Mon, 26 Jul 2015 10:00:08 GMT
Location: http://fulfilment.restbucks.com/order/9876


...
```

Request

```
GET /order/9876 HTTP/1.1
Host: fulfilment.restbucks.com
```

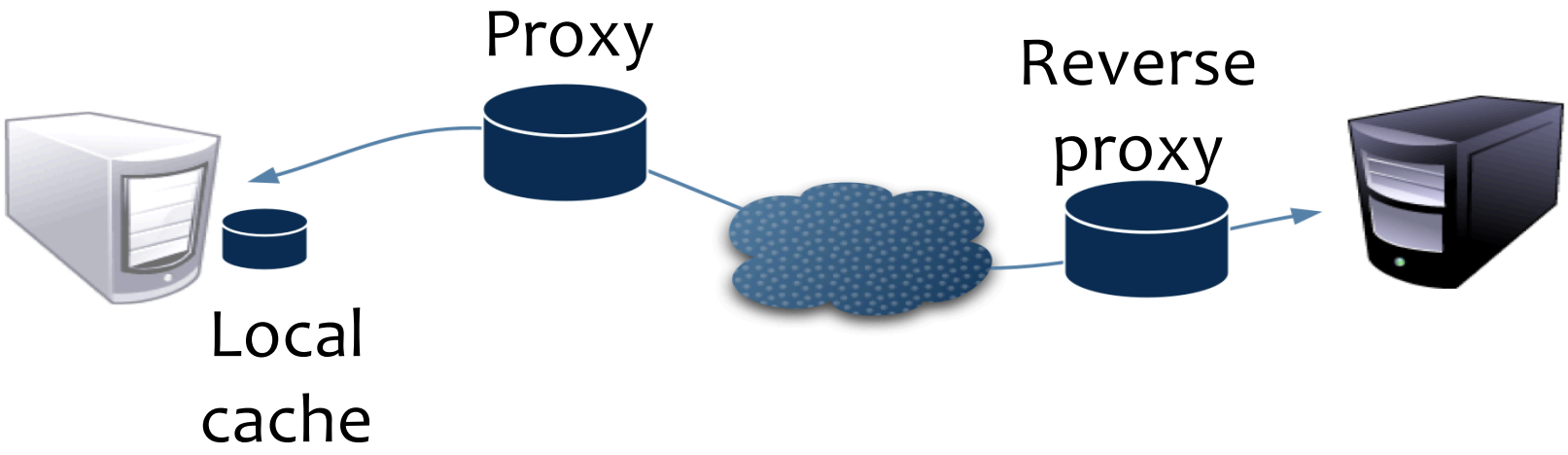
Return Order

Response

```
HTTP/1.1 200 OK
Cache-Control: public, max-age=0
Date: Mon, 26 Jul 2015 10:00:30 GMT
ETag: "4d3e88c9"
Content-Type: application/atom+xml

<entry>
  <title>Order 9876</title>
  <id>urn:uuid:f79300cc-c5c4-4d3d-9282-7a0f825d590d</id>
  <link rel="self"
        href="http://fulfilment.restbucks.com/order/9876"/>
  <link rel="payment" type="application/xhtml+xml"
        href="https://payments.restbucks.com/payment/1010"/>
  <link rel="quote" type="application/vnd.collection+json"
        href="http://quoting.restbucks.com/quote/1234"/>
  <updated>2015-07-26T10:00:15Z</updated>
  <category term="AwaitingPayment"/>
  <content type="application/xml" xmlns="http://schemas.restbucks.com/order">
    <order>
      <description>Costa Rica Tarrazu</description>
      <amount>250g</amount>
      <price>3.00</price>
    </order>
  </content>
</entry>
```

Polling Warms Caches

Client is responsible for guaranteed delivery



ETag Header

ETag

Opaque “checksum” of resource state

When resource changes, entity tag changes

Conditional GET

`If-None-Match: <etag value>`

“Return a full response only if the resource *has* changed”

`304 Not Modified`

Conditional POST/PUT

`If-Match: <etag value>`

“Process this request only if the resource *hasn’t* changed”

`412 Precondition Failed`

Design Strategies

Polling

Client guarantees delivery of notification

ETag

Efficient re-querying

Cache but revalidate strategy

Tradeoff between bandwidth usage and consistency

Navigate to Payment Form

Request

```
GET https://payments.restbucks.com/payment/1010 HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
Cache-Control: no-store
Date: Mon, 26 Jul 2015 10:00:31 GMT
Content-Type: application/xhtml+xml

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Payment</title>
  </head>
  <body>
    <form method="post"
      action="https://payments.restbucks.com/payment/1010"
      enctype="application/x-www-form-urlencoded">
      <input type="text" name="card-type"/>
      <input type="text" name="name"/>
      <input type="text" name="card-number"/>
      <input type="text" name="security-code"/>
      <input type="submit" value="Submit"/>
    </form>
  </body>
</html>
```

Submit Payment Details

Request

```
POST https://payments.restbucks.com/payment/1010 HTTP/1.1
Content-Type: application/x-www-form-urlencoded

card-type=Visa+Debit&name=MR+JOHN+SMITH
➥ &card-number=4876512418675010&security-code=212
```

Response

```
HTTP/1.1 200 OK
Cache-Control: no-store
Date: Mon, 26 Jul 2015 10:00:32 GMT
Content-Type: application/xhtml+xml

<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>Payment Confirmation</title></head>
  <body>
    <form method="post"
      action="http://fulfilment.restbucks.com/order/9876/confirmation"
      enctype="application/x-www-form-urlencoded">
      <input type="hidden" name="confirmation-id">6a0806ca</input>
      <input type="continue" value="Continue"/>
    </form>
  </body>
</html>
```

Notify Restbucks

Request

```
POST /order/9876/confirmation HTTP/1.1
Host: fulfilment.restbucks.com
Content-Type: application/x-www-form-urlencoded
Content-Length: ...

confirmation-id=6a0806ca
```

Response

```
HTTP/1.1 303 See Other
Date: Mon, 26 Jul 2015 10:00:33 GMT
Location: http://fulfilment.restbucks.com/order/9876
```

Get Order

Request

```
GET /order/9876 HTTP/1.1
Host: fulfilment.restbucks.com
If-None-Match: "4d3e88c9"
```

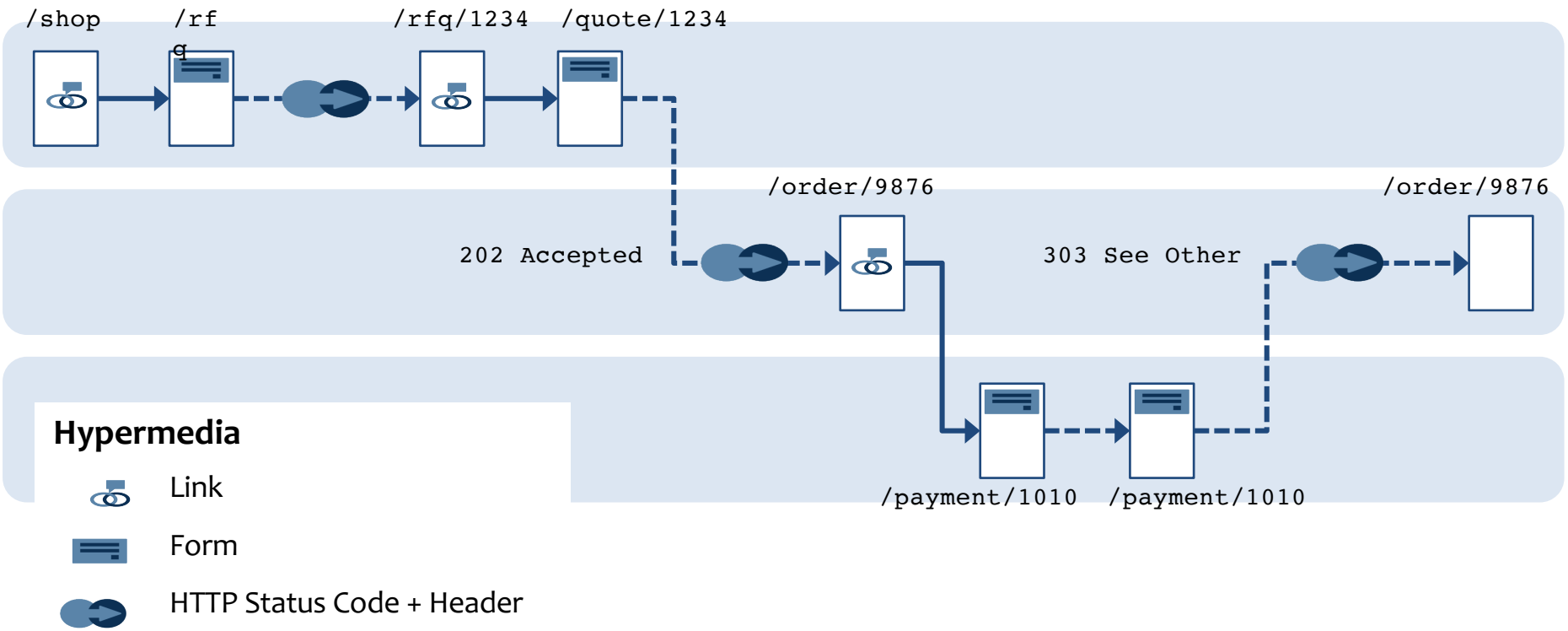
Return Paid Order

Response

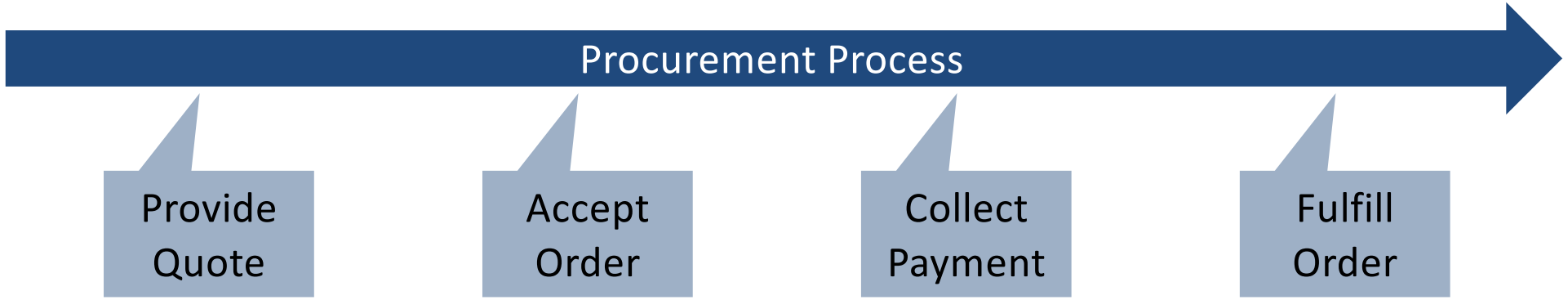
```
HTTP/1.1 200 OK
Cache-Control: public, max-age=0
Date: Mon, 26 Jul 2015 10:00:34 GMT
ETag: "5612cfaa"
Content-Type: application/atom+xml

<entry>
  <title>Order 9876</title>
  <id>urn:uuid:f79300cc-c5c4-4d3d-9282-7a0f825d590d</id>
  <link rel="self"
        href="http://fulfilment.restbucks.com/order/9876"/>
  <link rel="quote" type="application/vnd.collection+json"
        href="http://quoting.restbucks.com/quote/1234"/>
  <updated>2015-07-26T10:00:31Z</updated>
  <category term="Paid"/>
  <content type="application/xml" xmlns="http://schemas.restbucks.com/order">
    <order>
      <description>Costa Rica Tarrazu</description>
      <amount>250g</amount>
      <price>3.00</price>
    </order>
  </content>
</entry>
```

Use Hypermedia to Guide the Client

Clients *apply* resources to achieve an *application* goal



Protocol Layers



Business process orchestration/choreography	Application
Resource communication protocol	Hypermedia
Document transfer protocol	HTTP

Resources

- Resources act like an anticorruption layer
  - They adapt your domain on behalf of clients
- Clients manipulate resources through exchanging resource representations
  - Much like work in a 1950s office is triggered by exchanging documents

Documenting Your API

- Media Types
  - Schemas
  - Processing model
- Semantics
  - Link relations
  - Form annotations
- HTTP idioms
  - Headers
  - Methods
  - Status codes
  - Remember, inline forms help specify HTTP idioms at runtime

Documenting Your API

- Media Types
  - Schemas
  - Processing model
- Semantics
  - Link relations
  - Form annotations
- HTTP idioms
  - Headers
  - Methods
  - Status codes

Example Documentation: Atom Publication Protocol

- Media Types
  - Refers to Atom Syndication Format specification for details of Atom feed and entries
  - Defines 2 additional protocol documents:
    - Atom service document
    - Atom category document
- Semantics
  - Defines the `edit` and `edit-media` link relations
- HTTP idioms
  - `POST` to collection to create new entry, returns `201 Created`
  - `PUT` to existing entry to edit, returns `200 OK`
  - Conditional `PUT` to avoid lost updates

APIs

- A good API tells you:
  - What components are available to the client (interface)
  - How to manipulate these components to achieve an application goal (protocol)
- Protocols can be described in two ways:
  - Upfront, static specification
  - Dynamic, just-in-time disclosure

Web APIs

- Correspondingly, there are two ways of describing a Web API:
  - Documenting URI templates, document formats, and the HTTP idioms necessary to coordinate document exchanges
  - Using hypermedia to progressively disclose a protocol, advertising next steps to the client as it applies the protocol

Hypermedia

- Links
  - Advertise ***safe*** operations
  - Use ***link relations*** to provide semantic context
- Forms
  - Advertise ***unsafe*** operations
  - Program the client with the HTP idioms necessary to execute the operation
  - Describe the document to be submitted (fields and/or schema)
- Some HTTP status code and header combinations act as simple hypermedia

Design Strategies

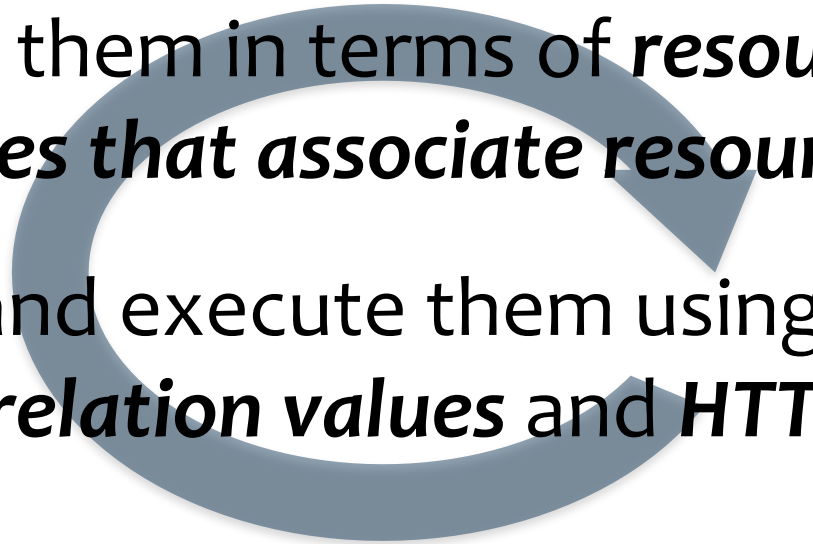
- Provide a homepage or entry point at a well-known, stable URI
- Cache data in the network, to improve availability and reduce latency
- Use pre-filled forms to round-trip data in self-descriptive messages
- Use `202 Accepted` for long-running operation
- Use conditional `GET` to enable efficient polling
- Use conditional `POST/PUT` to avoid the lost update problem

If In Doubt...

Design your API ***as if*** you're building an old-fashioned (no JavaScript) HTML app

Distributed application design and implementation guidelines

1. Model applications in terms of ***application state machines***
2. Implement them in terms of ***resource lifecycles*** and the ***rules that associate resources***
3. Advertise and execute them using ***media types***, ***link relation values*** and ***HTTP idioms***




Test-Driven Resources

Test-Driven Resource Development

- Goal:
  - Ensure that each resource:
    - Adopts HTTP uniform interface
    - Interacts with underlying domain
    - Produces correct representations
    - Exposes domain protocol through hypermedia

Quote Resource

```
[ServiceContract]
public class Quote
{
  private readonly IQuotationEngine quoteE

  public Quote(IQuotationEngine quoteEngine)
  {
    this.quote
  }

  [WebGet(UriTemplate = "{id}")]
  public HttpResponseMessage<EntityBody> Get(string id,
                                             HttpRequestMessage request)
  {
    //Get quotation from quotation engine
    //Add HTTP headers to response
    //Return response
  }
}
```

Inject domain/resource state

Dispatch on HTTP method

Microsoft ASP.NET Web API

http://www.asp.net/web-api

HTTP Uniform Interface – Status Codes

```
[Test]
public void ShouldReturn404NotFoundWhenGettin
{
  var quote = new Quote(EmptyQuotationEngine.Instance);

  try
  {
    quote.Get(Guid.NewGuid().ToString("N"), new HttpRequestMessage());
    Assert.Fail();
  }
  catch (HttpResponseException ex)
  {
    Assert.AreEqual(HttpStatusCode.NotFound, ex.Response.StatusCode);
  }
}
```

Always throws KeyNotFoundException

Return 404 When Quotation Doesn't Exist

```
public class Quote
{
  private readonly IQuotationEngine quoteEngine;

  public Quote(IQuotationEngine quoteEngine)
  {
      this.quoteEngine = quoteEngine;
  }

  [WebGet(UriTemplate = "{id}")]
  public HttpResponseMessage<EntityBody> Get(string id,
                                             HttpRequestMessage request)
  {
    Quotation quote;
    try
    {
      quote = quoteEngine.GetQuote(new Guid(id));
    }
    catch (KeyNotFoundException)
    {
      throw new HttpResponseException(HttpStatusCode.NotFound);;
    }
    return null;
  }
}
```

HTTP Uniform Interface – Headers

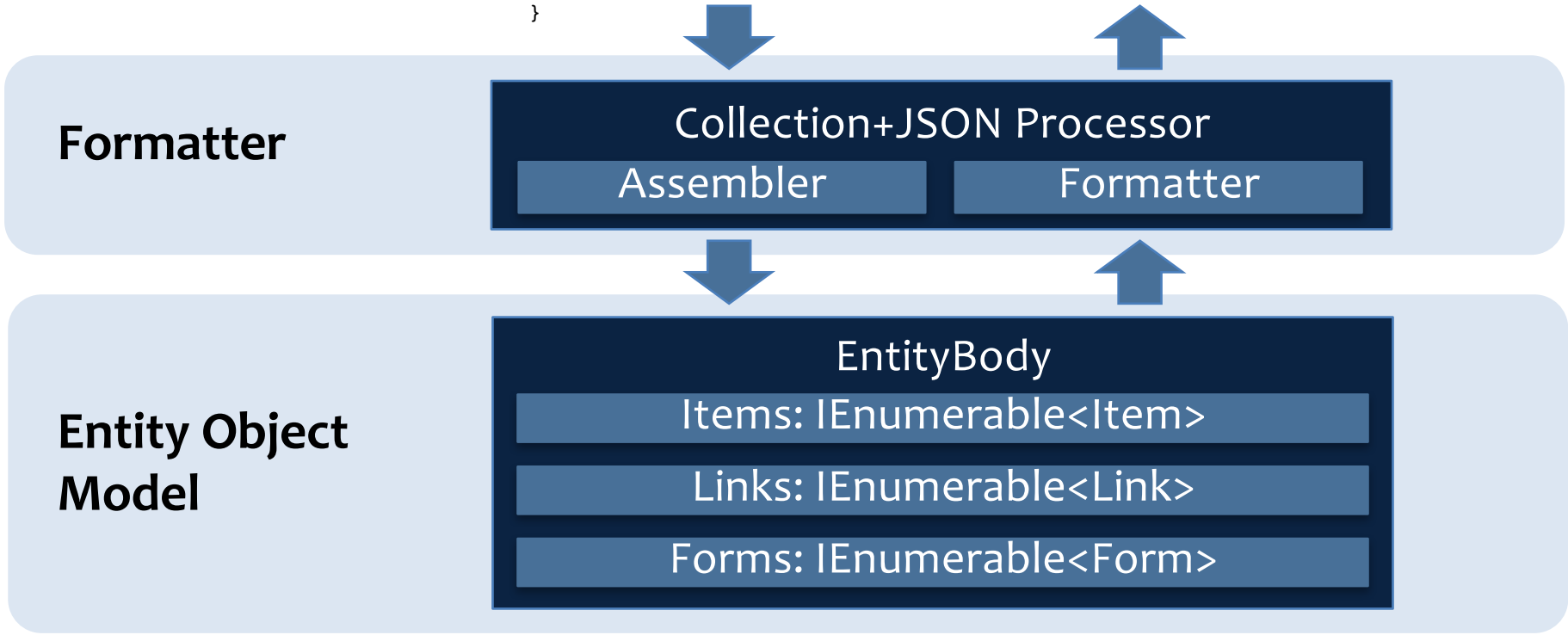
```
[Test]
public void ResponseShouldExpire7DaysFromDateTimeQuoteWasCreated()
{
  var resource = new Quote(DummyQuotationEngine.Instance);
  var response = resource.Get(Guid.NewGuid().ToString("N"),
                              new HttpRequestMessage());

  Assert.AreEqual("public", response.Headers.CacheControl.ToString());
  Assert.AreEqual(
    DummyQuotationEngine.Quotation.CreatedDateTime.AddDays(7.00),
    response.Content.Headers.Expires);
}
```

Add Caching Headers

```
public class Quote
{
  ...

  [WebGet(UriTemplate = "{id}")]
  public HttpResponseMessage<EntityBody> Get(string id,
                                             HttpRequestMessage request)
  {
    Quotation quote;

    //Retrieve quote
    ...

    var response = new HttpResponseMessage<EntityBody>(null) {
      StatusCode = HttpStatusCode.OK};

    response.Headers.CacheControl =
      new CacheControlHeaderValue {Public = true};
    response.Content.Headers.Expires =
      quote.CreatedDateTime.AddDays(7.0);

    return response;
  }
}
```

Separate Logical Entity Object Model from Representation Formatters

```
{
    "collection": {
        "href": "http://quoting.restbucks.com/shop",
        "links": [
            { "href": "/rfq", "rel": "request-for-quote" }
        ]
    }
}
```



Entity Body Object Model

```
public HttpResponseMessage<EntityBody> Get(string id,
                                           HttpRequestMessage request)
{
  //Retrieve quotation
  ...

  var body = new EntityBodyBuilder(new Uri("http://restbucks.com/"))
     .AddItem(new Item("coffee beans", new Amount("g", 250)))
     .AddLink(new Link(
       new Uri("quote/" + quoteId, UriKind.Relative),
       RestbucksMediaType.Value, LinkRelations.Self))
     .AddLink(new Link(
       new Uri("order-form/" + quoteId, UriKind.Relative),
       RestbucksMediaType.Value, LinkRelations.OrderForm))
     .Build();

  var response = new HttpResponseMessage<EntityBody>(body) {
     StatusCode = HttpStatusCode.OK};

  //Add headers
  ...

  return response;
}
```

Problem: URI redundancy

Routes

```
RouteTable.Routes.AddServiceRoute<Quote>("quote", configuration);
RouteTable.Routes.AddServiceRoute<OrderForm>("order-form", configuration);
```

Methods

```
[WebGet(UriTemplate = "{id}")]
public HttpResponseMessage<EntityBody> Get(string id,
                                           HttpRequestMessage request)
{
  ...
```

Links

```
var body = new EntityBodyBuilder(new Uri("http://restbucks.com/"))
  .AddItem(new Item("coffee beans", new Amount("g", 250)))
  .AddLink(new Link(
    new Uri("quote/" + quoteId, UriKind.Relative),
    RestbucksMediaType.Value, LinkRelations.Self))
  .AddLink(new Link(
    new Uri("order-form/" + quoteId, UriKind.Relative),
    RestbucksMediaType.Value, LinkRelations.OrderForm))
  .Build();
```

Solution: UriFactory

```
[UriTemplate("quote", "{id}")]
public class Quote
{
}

[Test]
public void UriFactoryExample()
{
  var uriFactory = new UriFactory();
  uriFactory.Register<Quote>();

  Assert.AreEqual(
    new Uri("quote/1234", UriKind.Relative),
    uriFactory.CreateRelativeUri<Quote>(1234));

  Assert.AreEqual(
    new Uri("http://restbucks.com/quote/1234"),
    uriFactory.CreateAbsoluteUri<Quote>(
      new Uri("http://restbucks.com"), 1234));

  Assert.AreEqual(
    new Uri("http://restbucks.com/"),
    uriFactory.CreateBaseUri<Quote>(
      new Uri("http://restbucks.com/quote/1234")));
}
```

DRY URIs

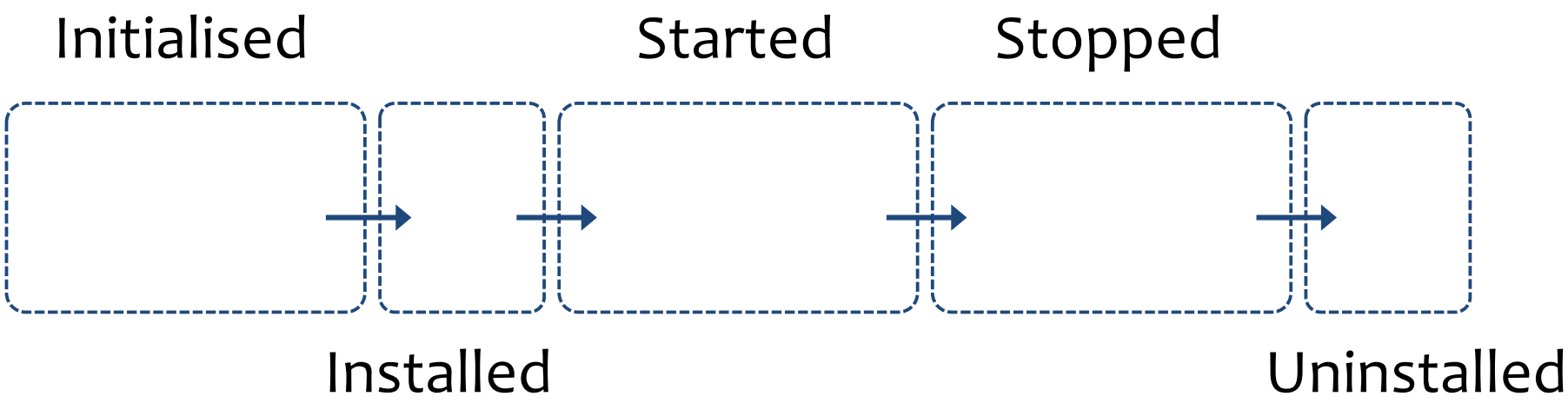
```
[ServiceContract]
[UriTemplate("quote", "{id}")]
public class Quote
{
  private readonly UriFactory uriFactory;

  public Quote(UriFactory uriFactory, IQuotationEngine quoteEngine)
  {...}

  [WebGet]
  public HttpResponseMessage<EntityBody> Get(string id,
                                             HttpRequestMessage request)
  {
    ...

    var baseUri = uriFactory.CreateBaseUri<Quote>(request.Uri);
    var body = new EntityBodyBuilder(baseUri, quote.LineItems.Select(
        li => new LineItemToItem(li).Adapt()))
      .AddLink(new Link(uriFactory.CreateRelativeUri<Quote>(quote.Id),
        RestbucksMediaType.Value, LinkRelations.Self))
      .AddLink(new Link(
        uriFactory.CreateRelativeUri<OrderForm>(quote.Id),
        RestbucksMediaType.Value, LinkRelations.OrderForm)).Build();

    ...
```

TDD Considerations

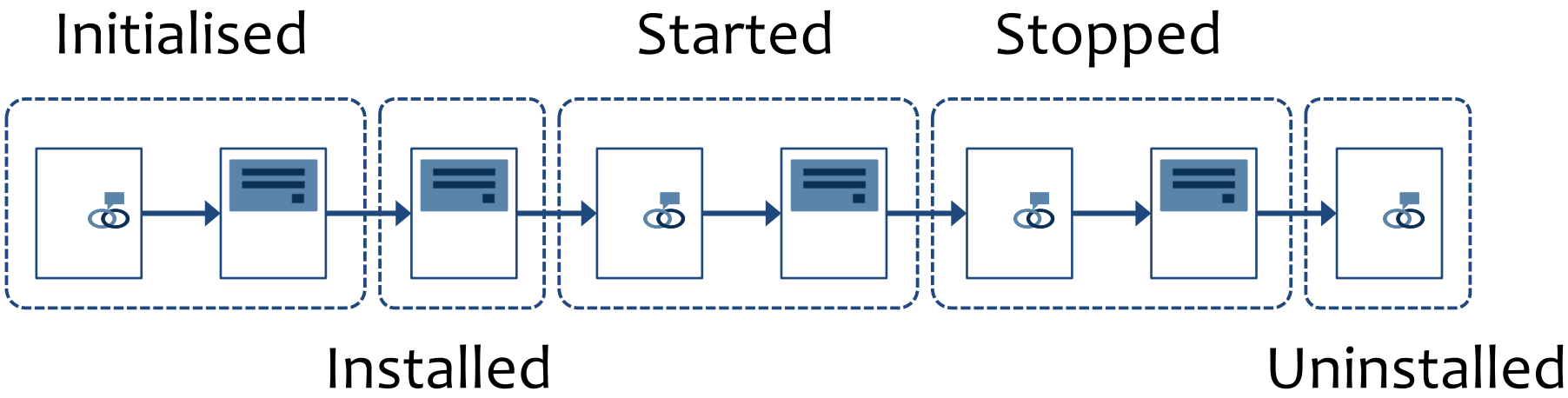
- Choose a framework that allows you to test HTTP interactions without having to spin up a service
- Write tests that assert correct use of HTTP idioms
  - Methods
  - Headers
  - Status codes
- Separate a logical entity body from its representations
  - Logic in each method to format for the wire
  - Or resource formatters as chosen by the framework's content negotiation mechanism
- DRY URIs
  - Use URI templates ***internally*** to document and implement URI construction
  - Provide a mapping from resource class to URI template


Client-Side Development

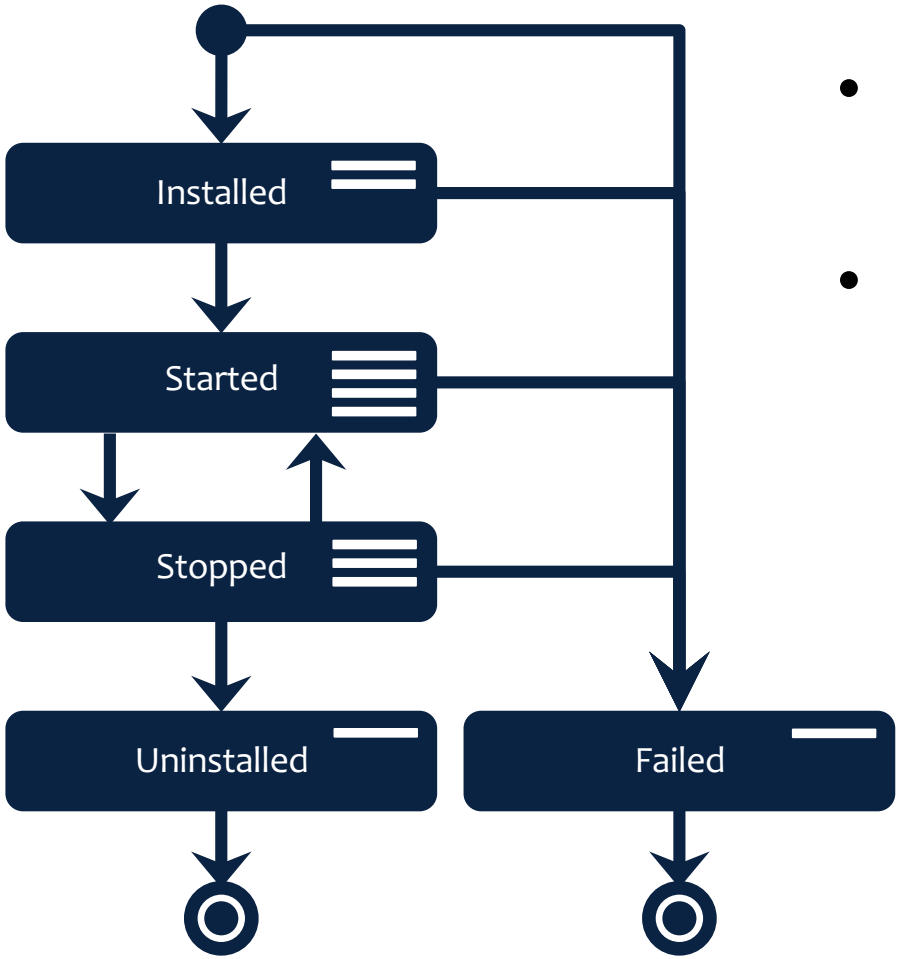
Changes in Application State are Client-Side Concerns



Hypermedia Systems Transform Application State



Client-Side State Machine



- Each state is a bucket of rules
- The rules available to the client change as the application changes

State Implementation

```
public class Initialised : IState
{
  ...

  public IState NextState(IClientCapabilities httpClient)
  {
    var rules = new Rules(
      When
        .IsTrue(response => prevResponse.ContainsForm(Forms.INSTALL))
        .Invoke(actions  => actions.SubmitForm(Forms.INSTALL))
        .Return(new[]{
           On.Status(HttpStatusCode.Created)
             .Do((response, vars) => new DatabaseInstalled(response, vars))}),
      When
        .IsTrue(response => prevResponse.ContainsLink(Links.INSTALLATION))
        .Invoke(actions  => actions.ClickLink(Links.INSTALLATION))
        .Return(new[]{
           On.Status(HttpStatusCode.OK)
             .Do((response, vars) => new Initialised(response, vars))}));

    return rules.Evaluate(previousResponse, stateVariables, httpClient);
  }

  public bool IsTerminalState { get { return false; } }
}
```

Get Homepage

Request

```
GET /database-agent HTTP/1.1
Host: localhost:38000
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <body>
    <a href="http://localhost:38000/database-agent/install" rel="installation">Install</a>
    ...
  </body>
</html>
```

Second Rule Fires

```
public class Initialised : IState
{
  ...

  public IState NextState(IClientCapabilities httpClient)
  {
    var rules = new Rules(
      When
        .IsTrue(response => prevResponse.ContainsForm(Forms.INSTALL))
        .Invoke(actions  => actions.SubmitForm(Forms.INSTALL))
        .Return(new[]{
           On.Status(HttpStatusCode.Created)
             .Do((response, vars) => new DatabaseInstalled(response, vars))}),
      When
        .IsTrue(response => prevResponse.ContainsLink(Links.INSTALLATION))
        .Invoke(actions  => actions.ClickLink(Links.INSTALLATION))
        .Return(new[]{
           On.Status(HttpStatusCode.OK)
             .Do((response, vars) => new Initialised(response, vars))}));

    return rules.Evaluate(previousResponse, stateVariables, httpClient);
  }

  public bool IsTerminalState { get { return false; } }
}
```

Get Installation Page

Request

```
GET /database-agent/install HTTP/1.1
Host: localhost:38000
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <body>
    <form class="install-database"
          method="POST"
          action="/database-agent/install"
          enctype="application/x-www-form-urlencoded">
        <input type="text" name="installUri"></input>
        <button type="submit" id="submit-form-install-database">Install</button>
    </form>
    ...
  </body>
</html>
```

First Rule Fires

```
public class Initialised : IState
{
  ...

  public IState NextState(IClientCapabilities httpClient)
  {
    var rules = new Rules(
      When
        .IsTrue(response => prevResponse.ContainsForm(Forms.INSTALL))
        .Invoke(actions  => actions.SubmitForm(Forms.INSTALL))
        .Return(new[]{
           On.Status(HttpStatusCode.Created)
             .Do((response, vars) => new DatabaseInstalled(response, vars))}),
      When
        .IsTrue(response => prevResponse.ContainsLink(Links.INSTALLATION))
        .Invoke(actions  => actions.ClickLink(Links.INSTALLATION))
        .Return(new[]{
           On.Status(HttpStatusCode.OK)
             .Do((response, vars) => new Initialised(response, vars))}));

    return rules.Evaluate(previousResponse, stateVariables, httpClient);
  }

  public bool IsTerminalState { get { return false; } }
}
```

Alternate Homepage (Inlined Form)

Request

```
GET /database-agent HTTP/1.1
Host: localhost:38000
```

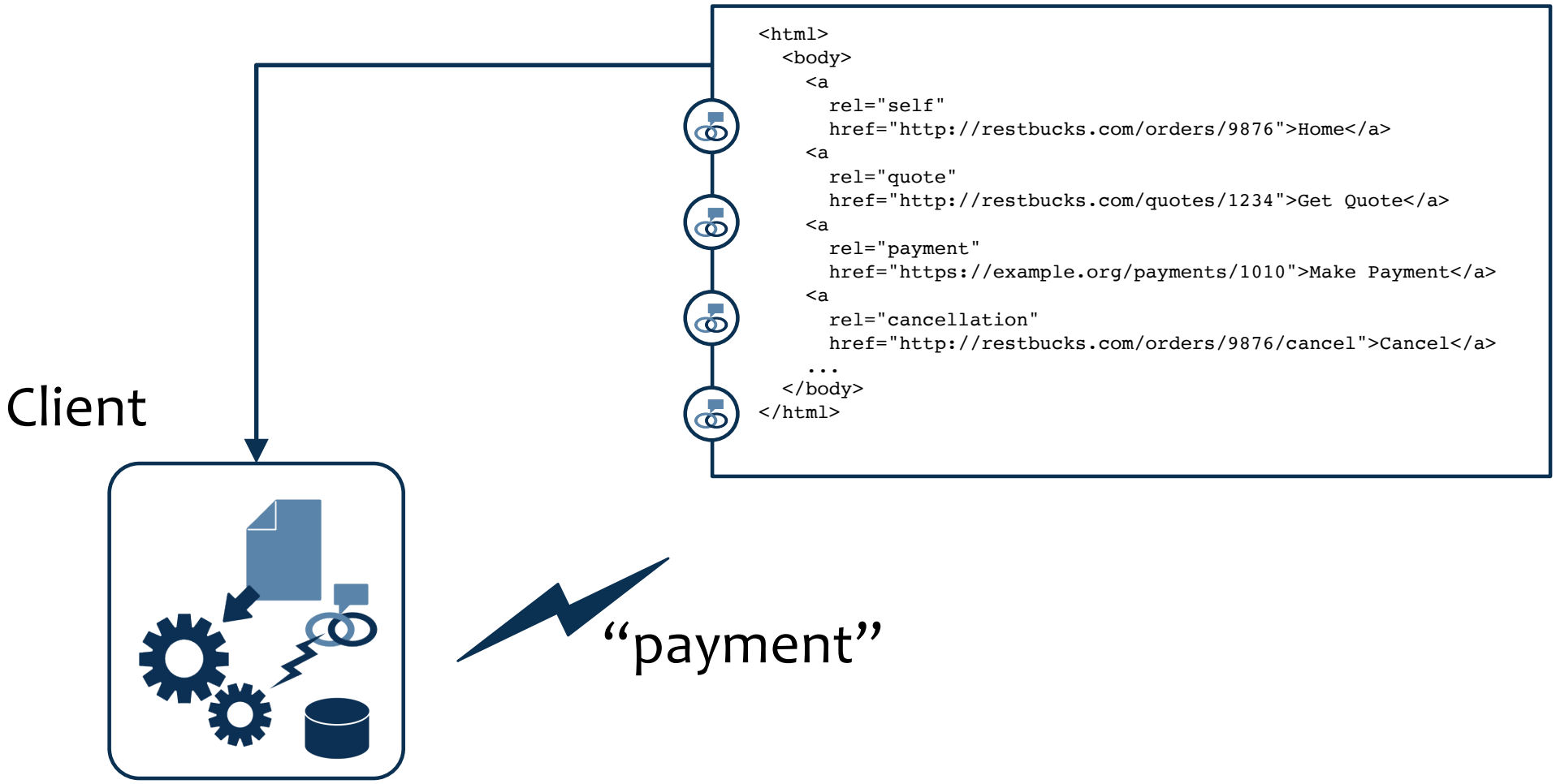
Response

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <body>
    <a href="/database-agent/install" rel="installation">Install</a>
    <form class="install-database"
          method="POST"
          action="/database-agent/install"
          enctype="application/x-www-form-urlencoded”>
        <input type="text" name="installUri"></input>
        <button type="submit" id="submit-form-install-database">Install</button>
    </form>
    ...
  </body>
</html>
```

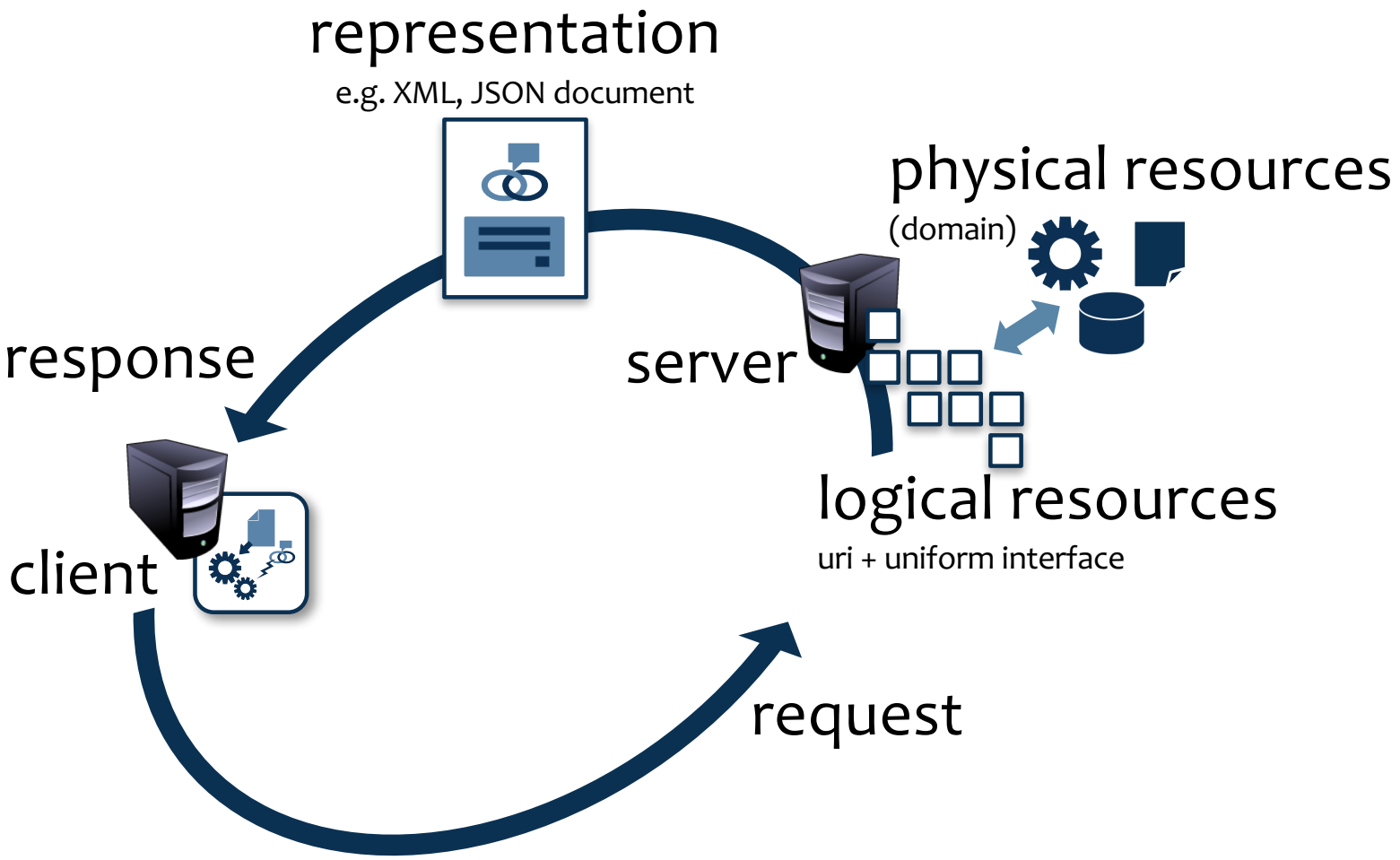
First Rule Fires

```
public class Initialised : IState
{
  ...

  public IState NextState(IClientCapabilities httpClient)
  {
    var rules = new Rules(
      When
        .IsTrue(response => prevResponse.ContainsForm(Forms.INSTALL))
        .Invoke(actions  => actions.SubmitForm(Forms.INSTALL))
        .Return(new[]{
           On.Status(HttpStatusCode.Created)
             .Do((response, vars) => new DatabaseInstalled(response, vars))}),
      When
        .IsTrue(response => prevResponse.ContainsLink(Links.INSTALLATION))
        .Invoke(actions  => actions.ClickLink(Links.INSTALLATION))
        .Return(new[]{
           On.Status(HttpStatusCode.OK)
             .Do((response, vars) => new Initialised(response, vars))}));

    return rules.Evaluate(previousResponse, stateVariables, httpClient);
  }

  public bool IsTerminalState { get { return false; } }
}
```

Agents, Events and Hypermedia Event Handlers



A Hypermedia-Driven Distributed Application



Summary

- Documents at a distance
- HATEOAS
- Hypermedia controls
- Testing is testing
- Client-side state machines and rule-based evaluation