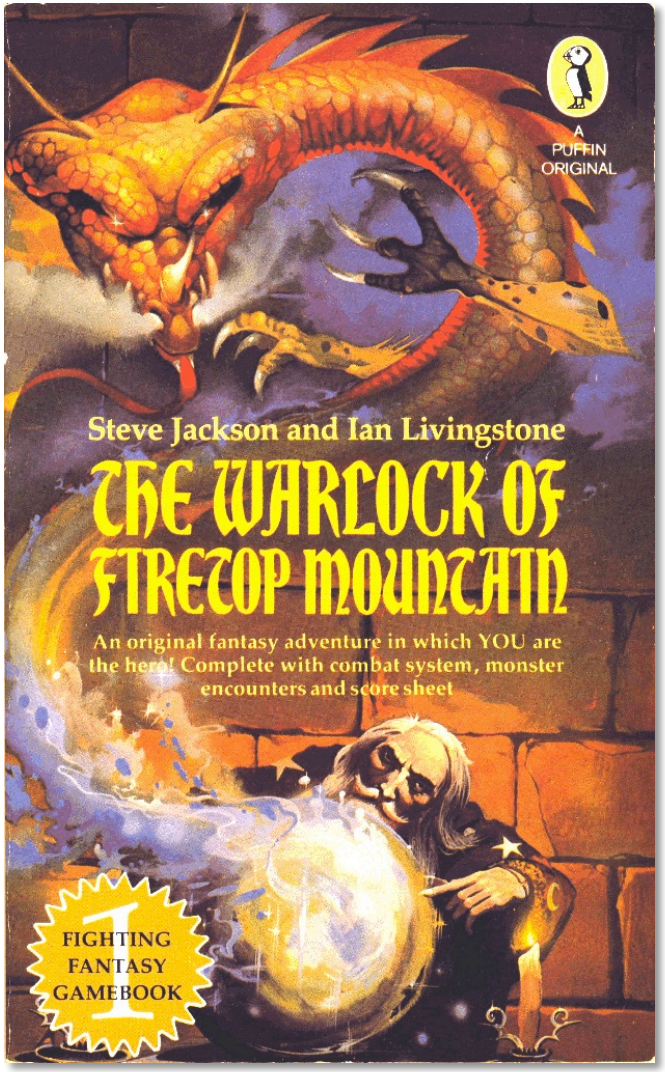
Test-Driven Web APIs

http://ian**S**robinson.com
@ian**S**robinson

Web is like 1950s office
Domain work as side-effect of shuffling docs around
HTTP is the Web's application protocol for shuffling documents around
Procurement app used for examples throughout
Example of POSTng doc to trigger work
Resources adapt domain for Web
Hypermedia guides client through domain protocol
Anatomy of a resource
What do we need to test when developing resources?
    - HTTP uniform interface
    - Representation formats
    - Hypermedia
    - Interaction with underlying domain

Pick your path to adventure



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

Executing a specialism in a generalized way

Warlock of Firetop Mountain Protocol

- Go north
- Defeat goblin
- Take key
- Unlock door
- Go east
- Solve riddle

Fighting Fantasy Transfer Protocol

- Numbered prose paragraphs
- Multiple choices keyed to numbered paragraphs

Architectural sympathy

HTTP is the Web's application protocol

Status codes

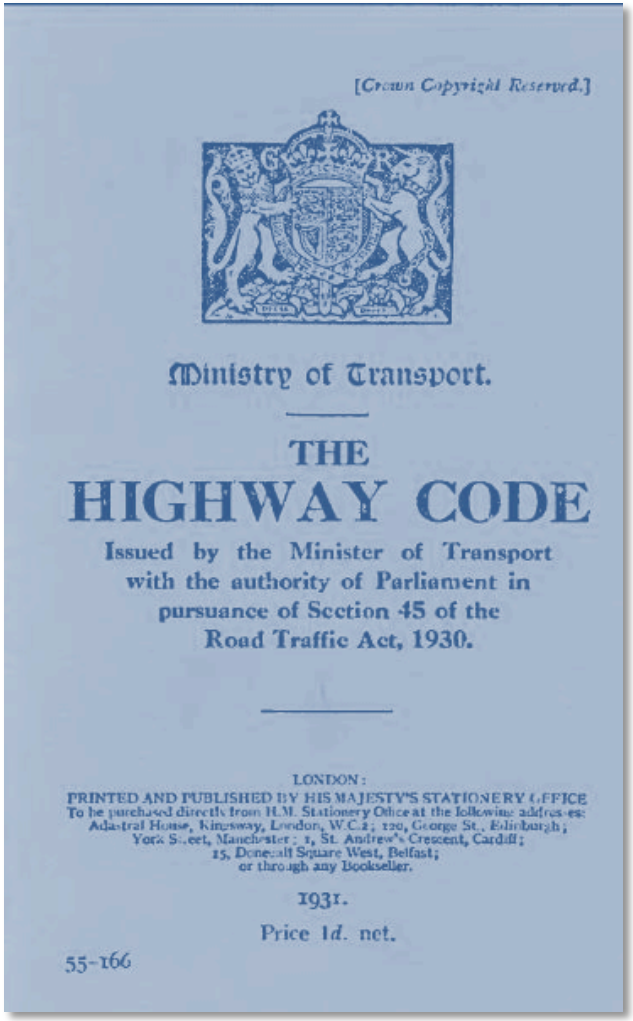
Coordination

Headers

Message processing context
Availability and consistency

Methods

Reliability



[Crown Copyright Reserved.]

Ministry of Transport.

THE
HIGHWAY CODE

Issued by the Minister of Transport
with the authority of Parliament in
pursuance of Section 45 of the
Road Traffic Act, 1930.

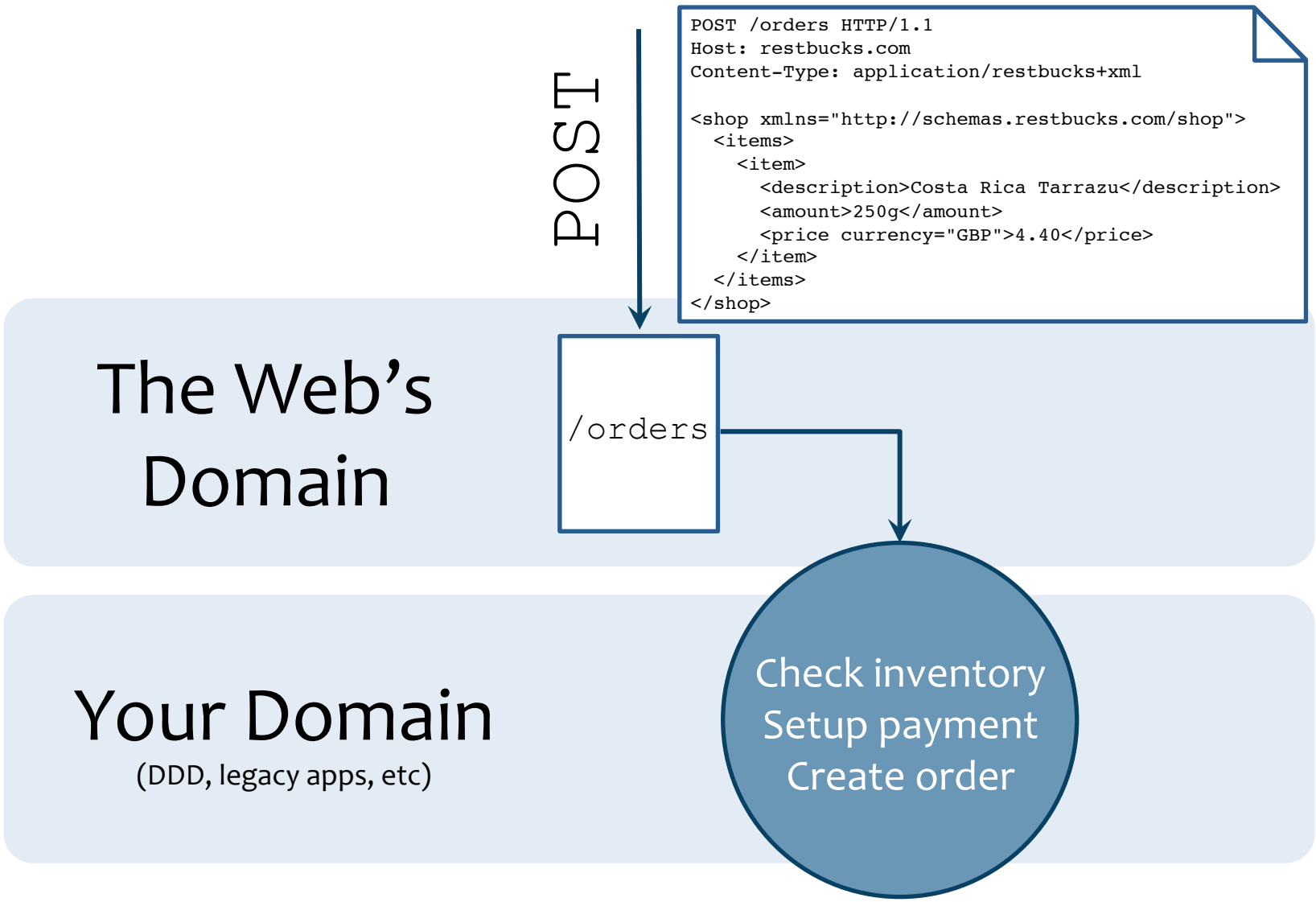
LONDON:
PRINTED AND PUBLISHED BY HIS MAJESTY'S STATIONERY OFFICE
To be purchased directly from H.M. Stationery Office at the following addresses:
Adastral House, Kingsway, London, W.C.2; 120, George St., Edinburgh;
York Street, Manchester; 1, St. Andrew's Crescent, Cardiff;
15, Donegall Square West, Belfast;
or through any Bookseller.

1931.

Price 1d. net.

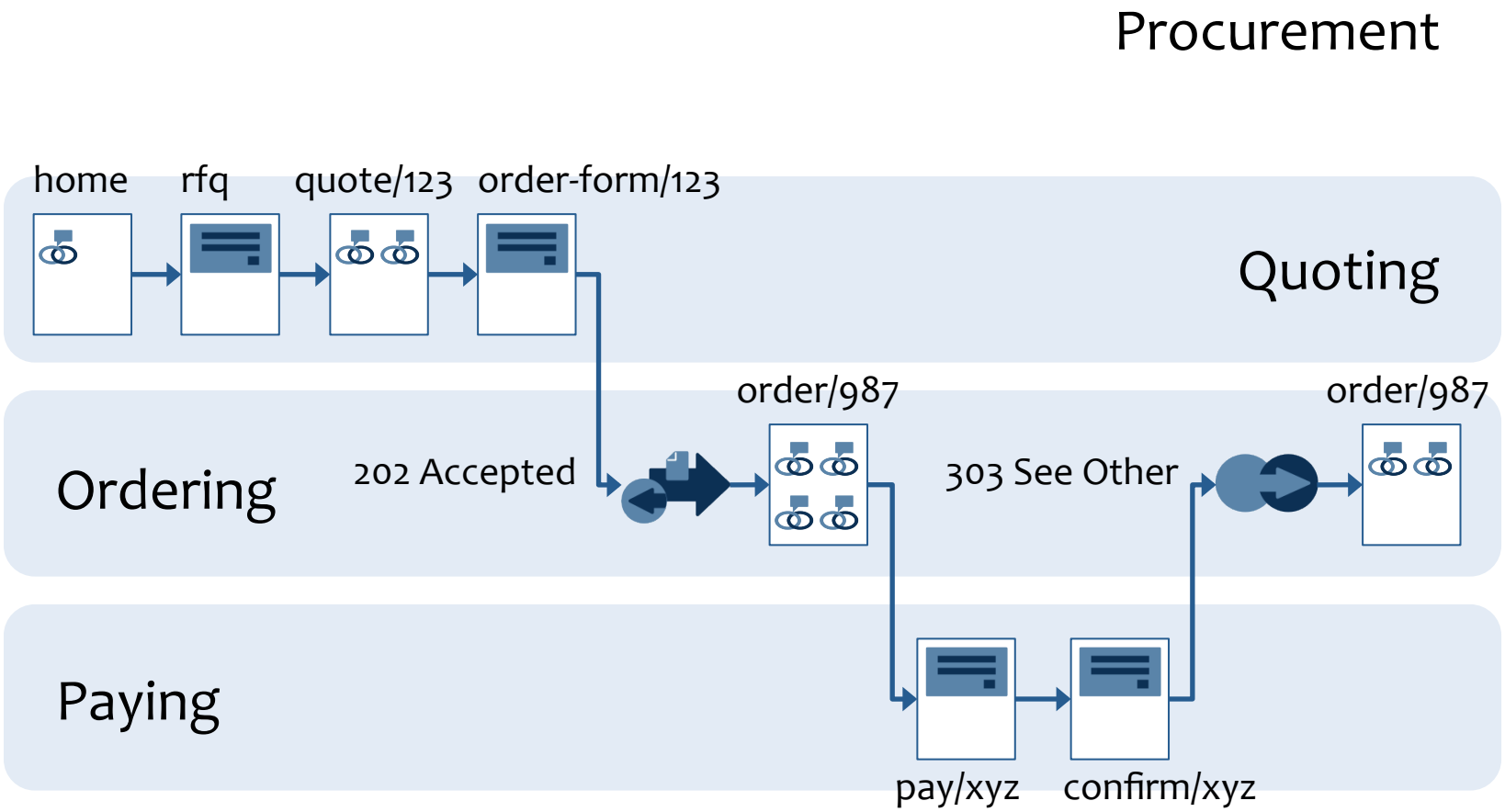
55-166

Work transacted as a side-effect of transferring documents

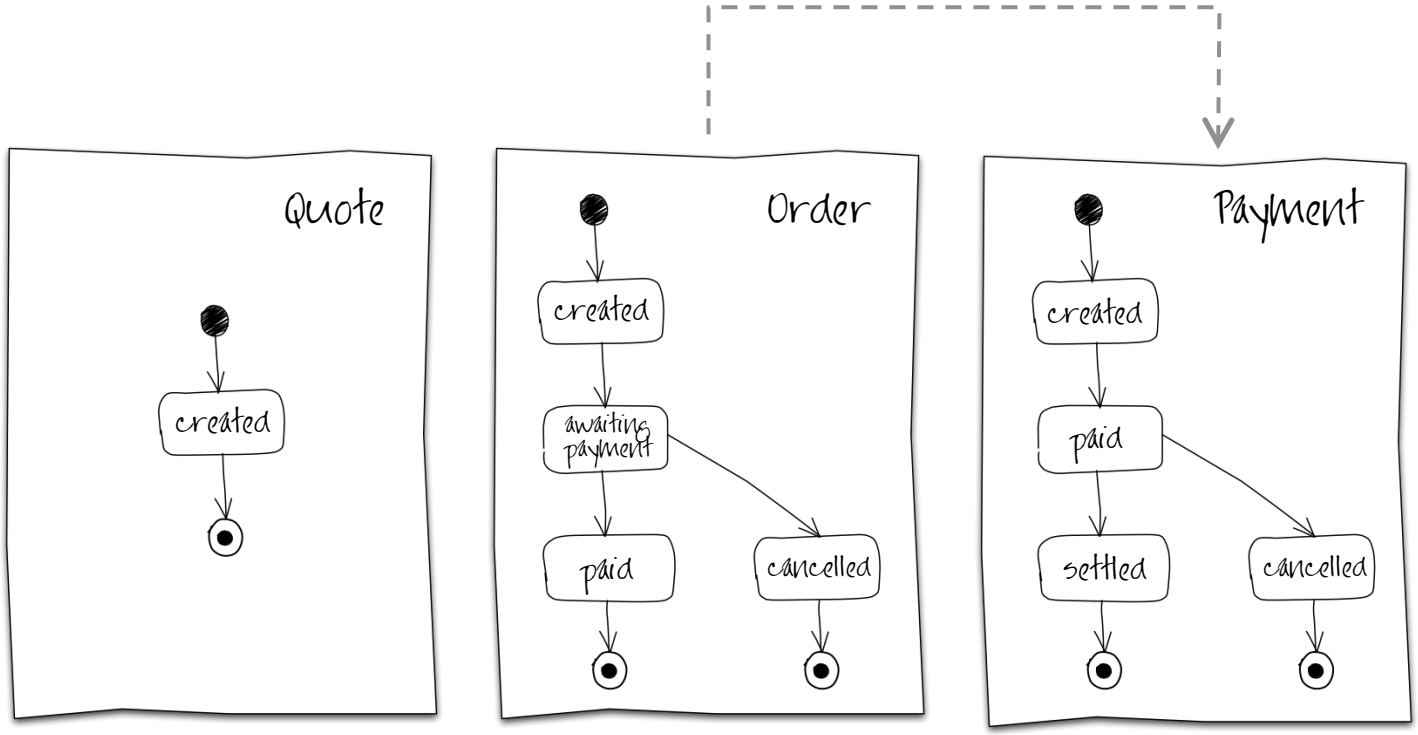


Resource Development

Links and forms guide the client through *your* business protocol



Autonomous resources



What is the role of a resource?

Resources adapt your domain for Web clients

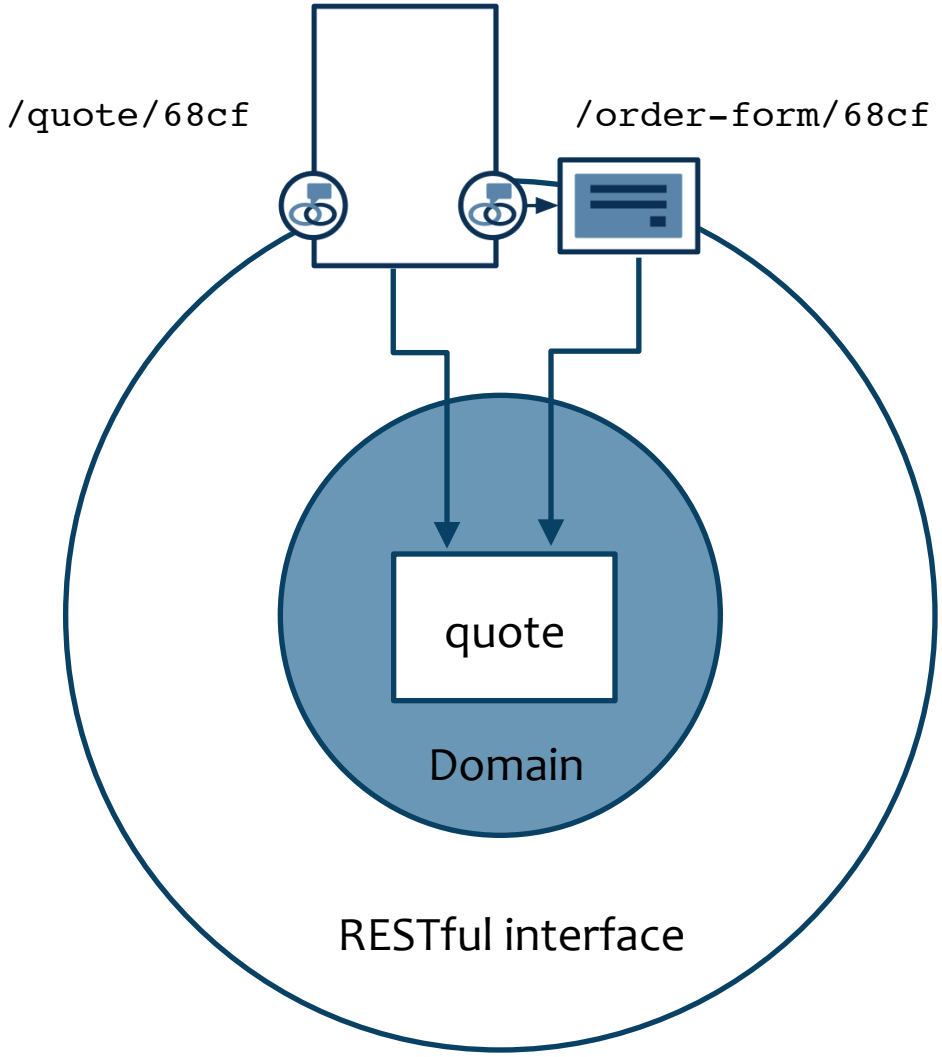
Quote

```
<shop xmlns:rb="http://relations.restbucks.com/"
      xml:base="http://restbucks.com/"
      xmlns="http://schemas.restbucks.com/shop">
  <items>
    <item>
      <description>coffee</description>
      <amount measure="g">125</amount>
      <price currency="GBP">1.25</price>
    </item>
  </items>
  <link rel="rb:order-form"
        type="application/vnd.restbucks+xml"
        href="order-form/68cff6e75a09474fa0098c9393aa6d4e" />
</shop>
```

Order form

```
<shop xml:base="http://restbucks.com/"
      xmlns="http://schemas.restbucks.com/shop">
  <model id="order" xmlns="http://www.w3.org/2002/xforms">
    <instance>
      <shop xml:base="http://restbucks.com/"
            xmlns="http://schemas.restbucks.com/shop">
        <items>
          <item>
            <description>coffee</description>
            <amount measure="g">125</amount>
            <price currency="GBP">1.25</price>
          </item>
        </items>
        <link rel="self"
              type="application/vnd.restbucks+xml"
              href="quote/68cff6e75a09474fa0098c9393aa6d4e" />
      </shop>
    </instance>
    <submission resource="/orders/?c=12345&amp;s=325"
                method="post"
                mediatype="application/vnd.restbucks+xml" />
  </model>
</shop>
```

Resources adapt your domain for Web clients

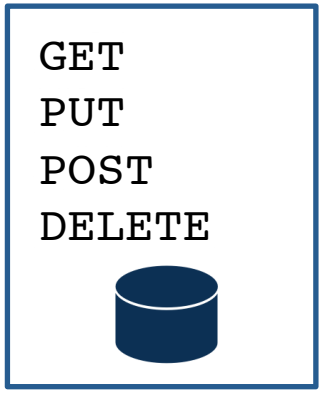


Anatomy of a resource

- Address + identity (URI)
- State (own or underlying domain)
- Representations (e.g. HTML, Atom, JSON dialect)
- Supports HTTP uniform interface

```
<shop xmlns="http://schemas.restbucks.com/shop"
  xmlns:rb="http://relations.restbucks.com/">
  <items>
    <item>
      <description>Costa Rica Tarrazu</description>
      <amount>250g</amount>
      <price currency="GBP">4.40</price>
    </item>
  </items>
  <link rel="rb:order-form"
    href="http://restbucks.com/order-forms/1234"/>
</shop>
```

```
http://restbucks.com/quotes/1234
```



Test-driven resources

Ensure that each resource:

- Adopts HTTP uniform interface
- Interacts with underlying domain/ resource state
- Produces correct representations
- Exposes domain protocol through hypermedia

Quote resource

```
[ServiceContract]
public class Quote
{
  private readonly IQuotationEngine quoteEngine;

  public Quote(I                    Engine)
  {
    this.quoteEn
  }

  [WebGet(UriTemplate = "{id}")]
  public HttpResponseMessage<Shop> 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 ShouldReturn404NotFoundWhenGet
{
  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 quotationEngine)
  {
     this.quotationEngine = quotationEngine;
  }

  [WebGet(UriTemplate = "{id}")]
  public HttpResponseMessage<Shop> 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(DummyQuotationEngine.QuoteId,
    new HttpRequestMessage());

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

public class DummyQuotationEngine : IQuotationEngine
{
  public static readonly IQuotationEngine Instance = new
    DummyQuotationEngine();

  public static readonly Quotation Quotation = new Quotation(...);
  public static readonly string QuoteId = Quotation.Id.ToString("N");
  ...
}
```

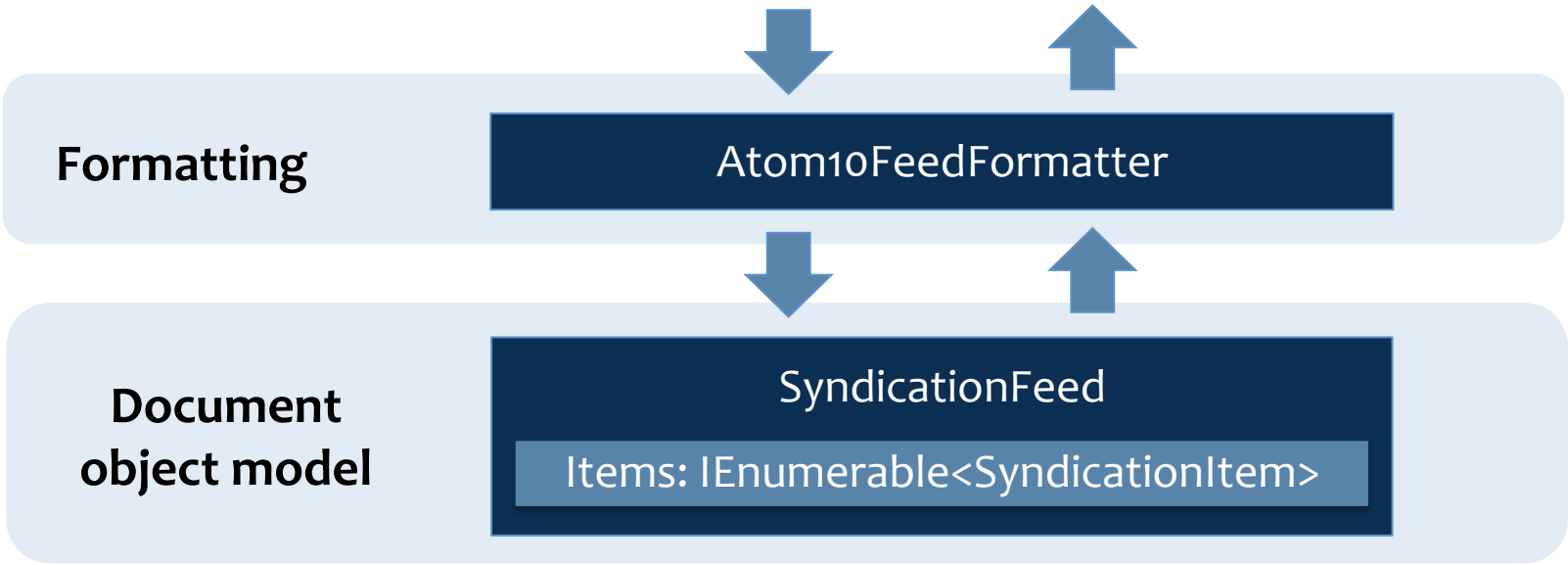
Static members provide test domain object

Add caching headers

```
public class Quote
{
  ...

  [WebGet(UriTemplate = "{id}")]
  public HttpResponseMessage<Shop> Get(string id,
                                       HttpRequestMessage request)
  {
    //Retrieve quote
    ...

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

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

    return response;
  }
}
```

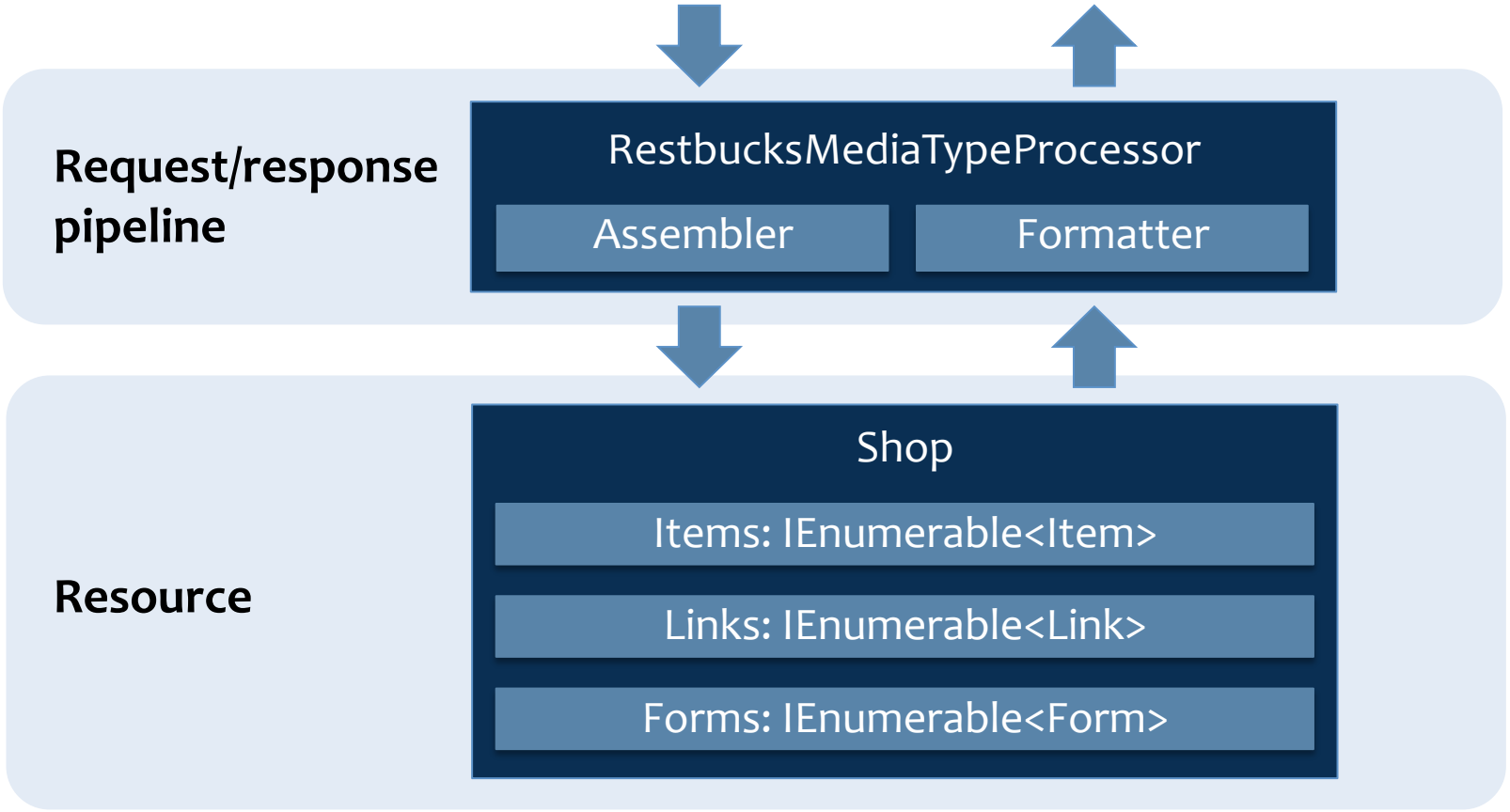
Document object model and formatter example

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Restbucks products and promotions</title>
  <id>urn:uuid:4aaf346c-1c9c-42cb-a006-5ff35d83c707</id>
  <updated>2010-11-29T04:38:09Z</updated>
  <author>
    <name>Product Catalog</name>
  </author>
  <generator>Product Catalog</generator>
  <link rel="self" href="http://localhost./updates" />
  <link rel="via" href="http://localhost./updates?p=3" />
  <link rel="prev-archive" href="http://localhost./updates?p=2" />
  <entry>
    ...
  </entry>
</feed>
```



Restbucks media type library

```
<?xml version="1.0" encoding="utf-8"?>
<shop xmlns:rb="http://relations.restbucks.com/"
  xml:base="http://win-cupsr6vr8g5/restbucks/"
  xmlns="http://schemas.restbucks.com/shop">
  <link rel="rb:rfq prefetch" type="application/vnd.restbucks+xml"
    href="request-for-quote/" />
</shop>
```



Entity body object model

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

  var body = new ShopBuilder(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<Shop>(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<Shop> Get(string id,
                                     HttpRequestMessage request)
{
   ...
```

Links

```
var body = new ShopBuilder(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")));
}
```

Registering resources at startup

Startup

```
var types = from t in assembly.GetTypes()
            where t.GetCustomAttributes(typeof (UriTemplateAttribute),
              false).Length > 0
            select t;
```

Register resources with `UriFactory` instance
Register resources with `RouteTable`

With each request

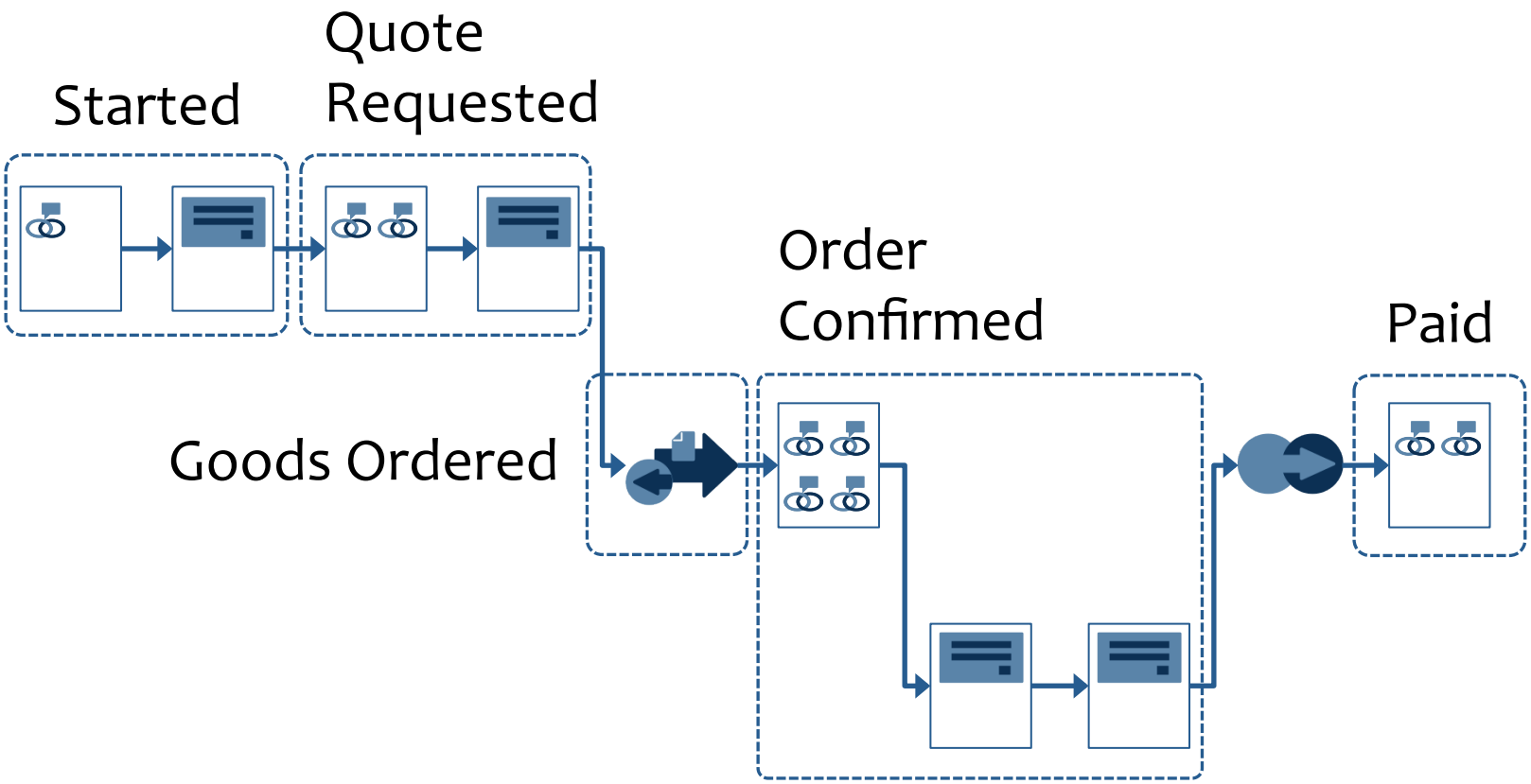
Populate `[WebGet]` and `[WebInvoke]` `UriTemplate` attributes

DRY URIs

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

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

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

    var baseUri = uriFactory.CreateBaseUri<Quote>(request.Uri);
    var body = ShopBuilder(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();

    ...
```

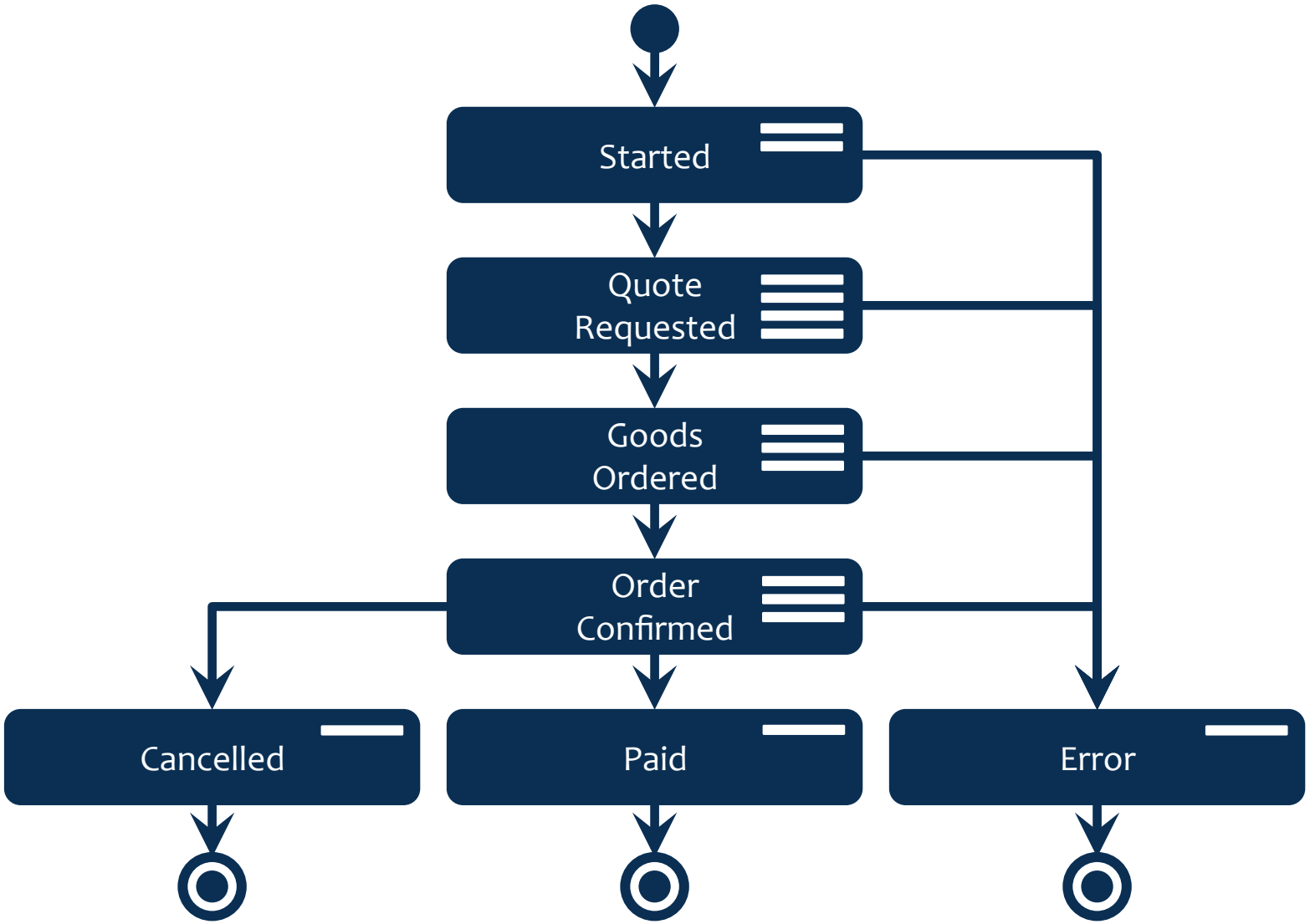
Client-side Development

The application is in the eye of the client



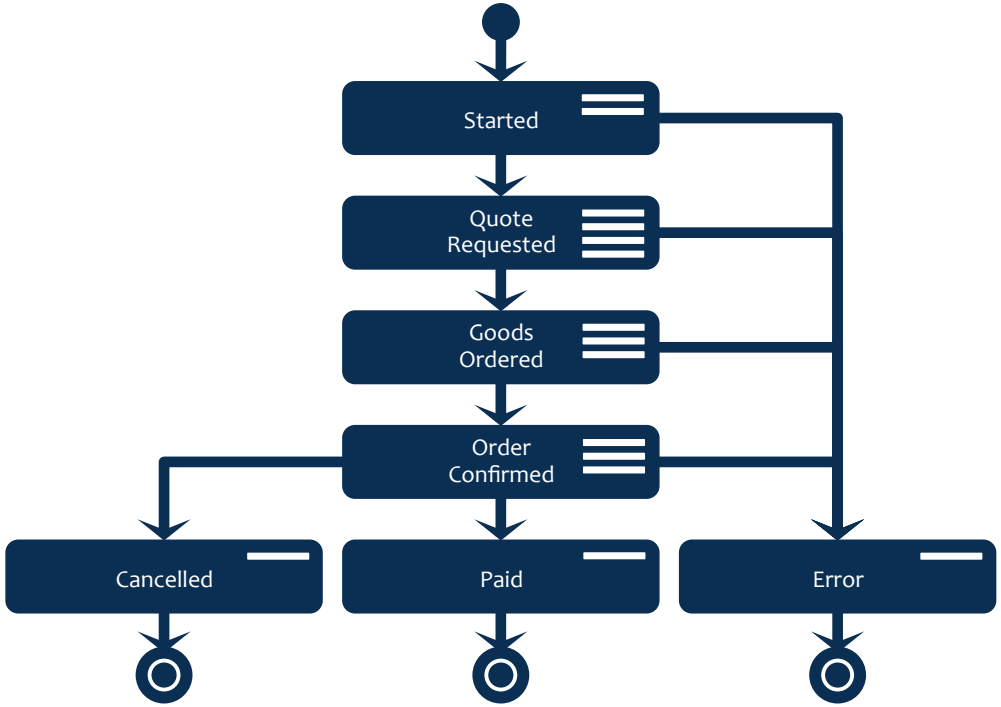
Clients *apply* resources to achieve an *application* goal

Client-side state machine: buckets of rules



Running the client

```
//Dictionary of state date includes entry point URI
var stateData = ...

var applicationState = new Uninitialized(stateData);

while (!applicationState.IsTerminalState)
{
  applicationState = applicationState.NextState(HttpClient.Instance);
}
```



A bucket of rules

```
public class Started : IState
{
  ...

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

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

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

Get homepage

Request

```
GET /shop/ HTTP/1.1
Accept: application/vnd.restbucks+xml
Host: restbucks.com
```

Response

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, public
Content-Length: 285
Content-Type: application/vnd.restbucks+xml
Date: Wed, 11 May 2011 17:27:22 GMT

<shop xmlns:rb="http://relations.restbucks.com/"
      xml:base="http://restbucks.com/"
      xmlns="http://schemas.restbucks.com/shop">
  <link rel="rb:rfq"
        type="application/vnd.restbucks+xml"
        href="request-for-quote/" />
</shop>
```

Second rule fires

```
public class Started : IState
{
  ...

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

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

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

Get request for quote form

Request

```
GET /request-for-quote/ HTTP/1.1
Accept: application/vnd.restbucks+xml
Host: restbucks.com
```

Response

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, public
Content-Length: 385
Content-Type: application/vnd.restbucks+xml
Date: Wed, 11 May 2011 22:12:52 GMT

<shop xml:base="http://restbucks.com/"
      xmlns="http://schemas.restbucks.com/shop">
  <model id="rb:rfq"
         schema="http://schemas.restbucks.com/shop"
         xmlns="http://www.w3.org/2002/xforms">
    <instance />
    <submission resource="quotes/"
                method="post"
                mediatype="application/vnd.restbucks+xml" />
  </model>
</shop>
```

First rule fires

```
public class Started : IState
{
  ...

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

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

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

Alternate homepage (inlined form)

Request

```
GET /shop/ HTTP/1.1
Accept: application/vnd.restbucks+xml
Host: restbucks.com
```

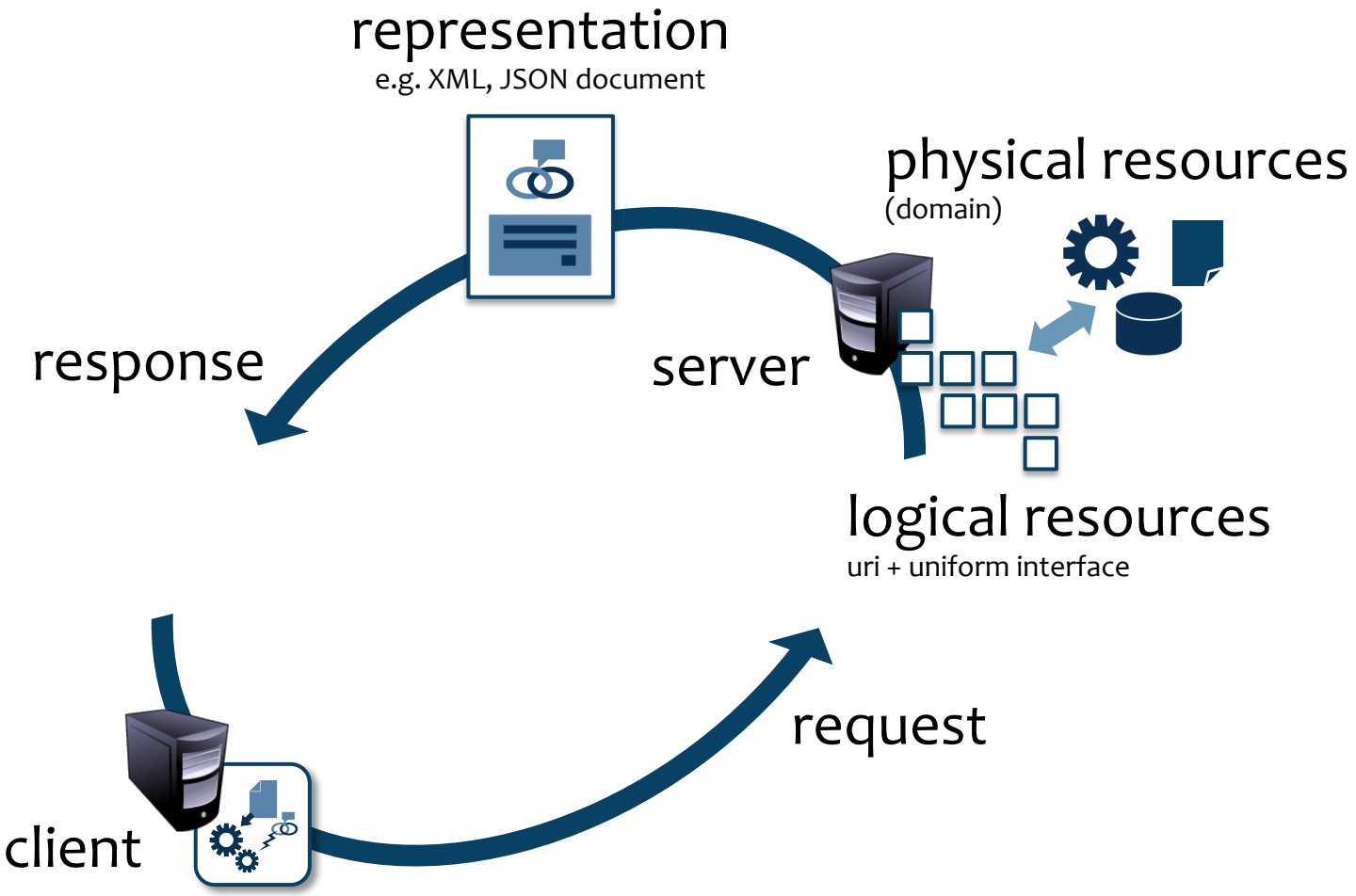
Response

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, public
Content-Length: 470
Content-Type: application/vnd.restbucks+xml
Date: Wed, 11 May 2011 17:25:31 GMT

<shop xml:base="http://restbucks.com/"
      xmlns="http://schemas.restbucks.com/shop">
  <link rel="rb:rfq"
        type="application/vnd.restbucks+xml"
        href="request-for-quote/" />
  <model id="rb:rfq"
         schema="http://schemas.restbucks.com/shop"
         xmlns="http://www.w3.org/2002/xforms">
    <instance />
    <submission resource="quotes/" method="post"
                mediatype="application/vnd.restbucks+xml" />
  </model>
</shop>
```

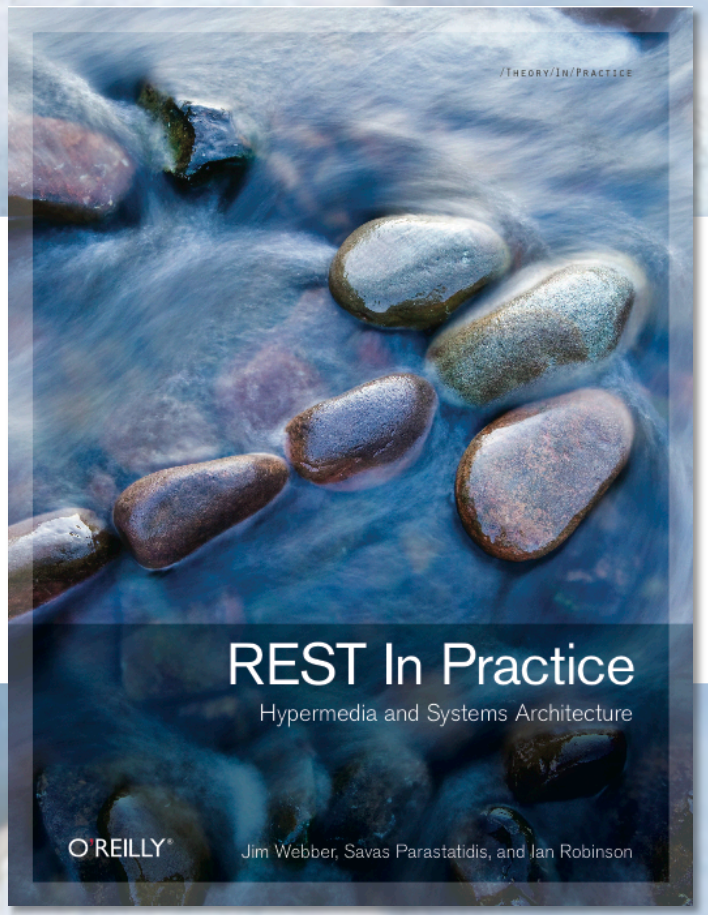
First rule fires

```
public class Started : IState
{
  ...

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

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

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

Result: a hypermedia Web API



Thank you



http://ian**S**robinson.com
@ian**S**robinson