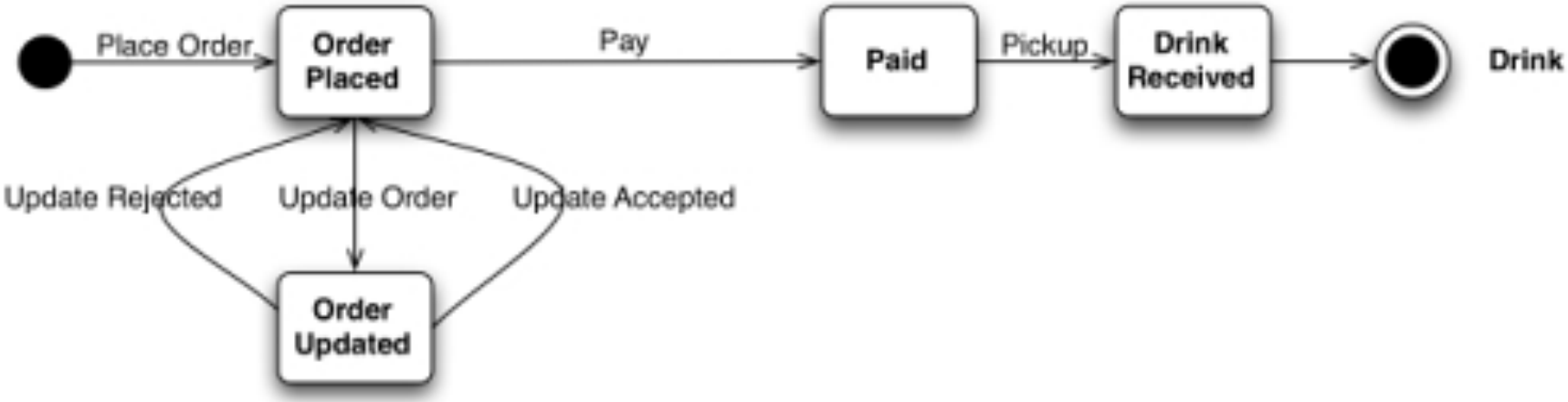
How to GET a cup of coffee

Based on Jim Webber's Article:

http://www.infoq.com/articles/webber-rest-workflow/

The customer state machine



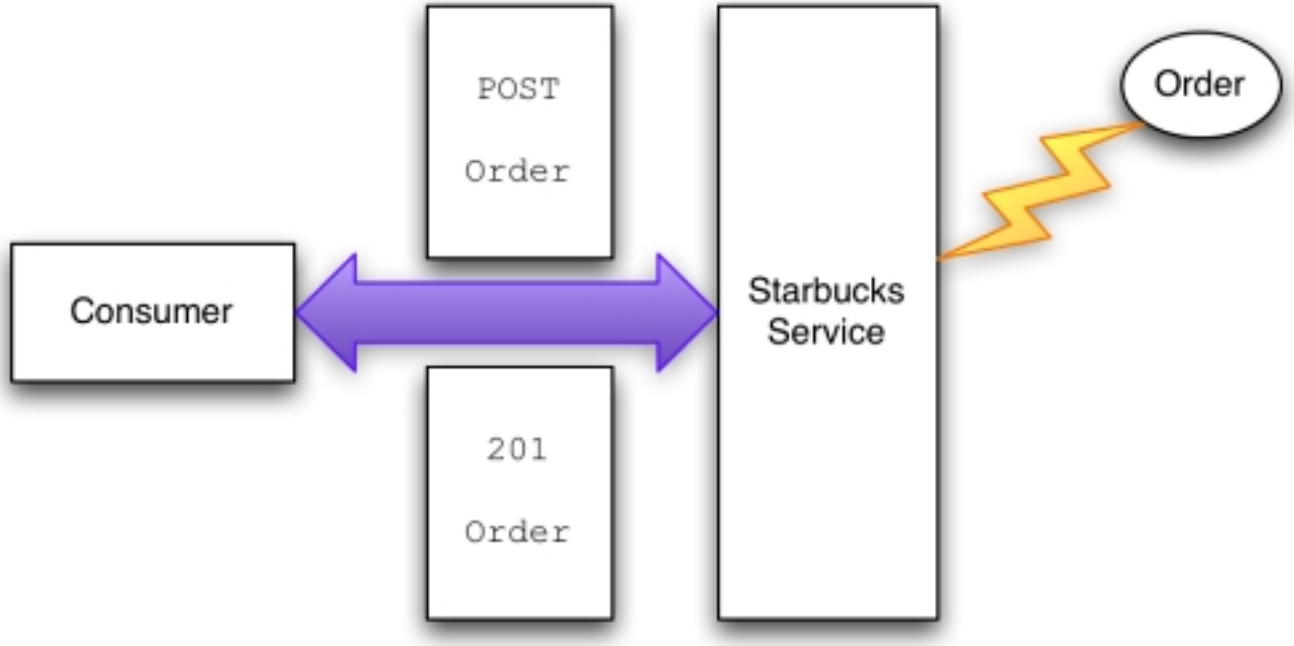
The barista's state machine

Story 1:

As a customer, I want to order a coffee

so that Starbucks can prepare my drink

Story 1. POST



POST

```
POST /order HTTP 1.1
Host: starbucks.example.org
Content-Type: application/xml
Content-Length: ...

<order xmlns="http://starbucks.example.org/">
  <drink>latte</drink>
</order>
```

```
201 Created
Location: http://starbucks.example.org/order/1234
Content-Type: application/xml
Content-Length: ...

<order xmlns="http://starbucks.example.org/">
  <drink>latte</drink>
  <cost>3.00</cost>
  <next xmlns="http://example.org/state-machine"
    rel="http://starbucks.example.org/payment"
    uri="https://starbucks.example.com/payment/order/1234"
    type="application/xml"/>
</order>
```

Story 2:

As a customer, I want to be able to change my drink to suit my tastes

OPTIONS

OPTIONS /order/1234 HTTP 1.1
Host: starbucks.example.org

200 OK Allow: GET, PUT

Look before you leap

PUT /order/1234 HTTP 1.1
Host: starbucks.example.com
Expect: 100-Continue

100 Continue

or

417 Expectation Failed

PUT

```
PUT /order/1234 HTTP 1.1
Host: starbucks.example.com
Content-Type: application/xml
Content-Length: ...

<order xmlns="http://starbucks.example.org/">
  <additions>shot</additions>
</order>
```

```
200 OK
Location: http://starbucks.example.com/order/1234
Content-Type: application/xml
Content-Length: ...

<order xmlns="http://starbucks.example.org/">
  <drink>latte</drink>
  <additions>shot</additions>
  <cost>4.00</cost>
  <next xmlns="http://example.org/state-machine"
    rel="http://starbucks.example.org/payment"
    uri="https://starbucks.example.com/payment/order/1234"
    type="application/xml"/>
</order>
```

Story 3:

As a customer, I want to be able to pay my bill to receive my drink

How to pay?

```
<next xmlns="http://example.org/state-machine"
  rel="http://starbucks.example.org/payment"
  uri="https://starbucks.example.com/payment/order/1234"
  type="application/xml"/>
```

OPTIONS/payment/order/1234 HTTP 1.1
Host: starbucks.example.com

Allow: GET, PUT

Payment PUT Headers

```
PUT /payment/order/1234 HTTP 1.1
Host: starbucks.example.com
Content-Type: application/xml
Content-Length: ...
Authorization: Digest username="Jane Doe"
realm="starbucks.example.org“
nonce="..."
uri="payment/order/1234"
qop=auth
nc=00000001
cnonce="..."
reponse="..."
opaque="..."
```

PUT Entity

```
<payment xmlns="http://
starbucks.example.org/">
    <cardNo>123456789</cardNo>
    <expires>07/07</expires>
    <name>John Citizen</name>
    <amount>4.00</amount>
</payment>
```

PUT Response

201 Created
Location: https://starbucks.example.com/payment/order/1234
Content-Type: application/xml
Content-Length: ...

```
<payment xmlns="http://starbucks.example.org/">
  <cardNo>123456789</cardNo>
  <expires>07/07</expires>
  <name>John Citizen</name>
  <amount>4.00</amount>
</payment>
```

Story 4:

As a barista, I want to see the list of drinks that I need to make, so that I can serve my customers

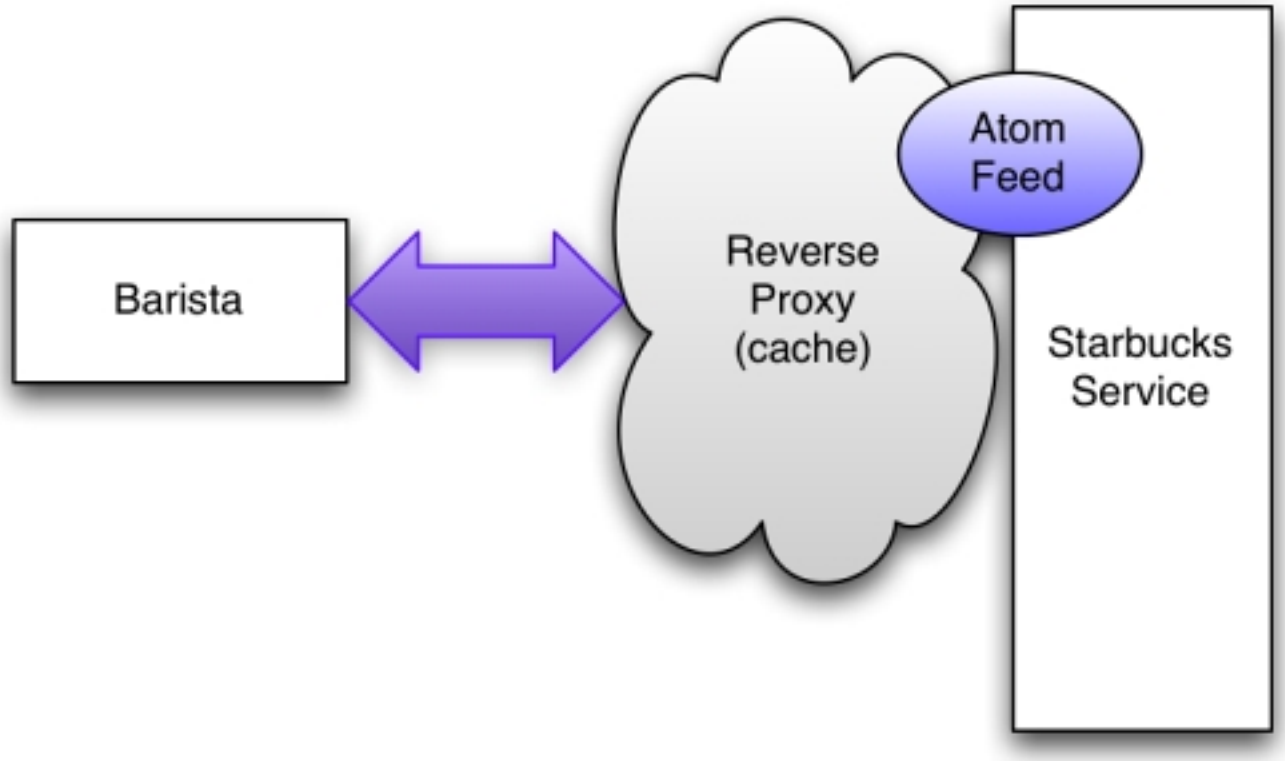
GET Drinks to be made

```
200 OK
Expires: Thu, 12Jun2008 17:20:33 GMT
Content-Type: application/atom+xml
Content-Length: ...

<?xml version="1.0" ?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Coffees to make</title>
  <link rel="alternate"
    uri="http://starbucks.example.org/orders"/>
  <updated>2008-06-10T19:18:43Z</updated>
  <author><name>Barista System</name></author>
  <id>urn:starkbucks:barista:coffees-to-make</id>

  <entry>
    <link rel="alternate" type="application/xml"
      uri="http://starbucks.example.org/order/1234"/>
    <id>http://starbucks.example.org/order/1234</id>
      ...
  </entry>
 ...
</feed>
```

Cacheing



AtomPub lets us update state

```
<entry>
  <published>2008-06-10T19:18:43Z </published>
  <updated>2008-06-10T19:20:32Z</updated>
  <link rel="alternate" type="application/xml"
    uri="http://starbucks.example.org/order/1234"/>
  <id>http://starbucks.example.org/order/1234</id>
  <content type="text+xml">
    <order xmlns="http://starbucks.example.org/">
      <drink>latte</drink>
      <additions>shot</additions>
      <cost>4.00</cost>
    </order>
    <link rel="edit"
      type="application/atom+xml"
      href="http://starbucks.example.org/order/1234/>
  ...
  </content>
</entry>
```

Update state

```
PUT /order/1234 HTTP 1.1
Host: starbucks.example.com
Content-Type: application/atom+xml
Content-Length: ...

<entry>
  ...
  <content type="text+xml">
    <order xmlns="http://starbucks.example.org/">
      <drink>latte</drink>
      <additions>shot</additions>
      <cost>4.00</cost>
      <status>preparing</status>
    </order>
    ...
  </content>
</entry>
```

Story 5:

As a barista, I want to check that a customer has paid for their drink so that I can serve it

GET

GET /payment/order/1234 HTTP 1.1 Host: starbucks.example.org

401 Unauthorized WWW-Authenticate: Digest realm="starbucks.example.org", qop="auth", nonce="ab656...", opaque="b6a9..."

Retry with credentials

```
GET /payment/order/1234 HTTP 1.1
Host: starbucks.example.org
Authorization: Digest username="barista joe"
realm="starbucks.example.org“ nonce="..."
uri="payment/order/1234" qop=auth nc=00000001
cnonce="..." reponse="..." opaque="..."”

200 OK
Content-Type: application/xml
Content-Length: ...
<payment xmlns="http://starbucks.example.org/">
   <cardNo>123456789</cardNo>
   <expires>07/07</expires>
   <name>John Citizen</name>
   <amount>4.00</amount>
</payment>
```

Story 6:

As a barista, I want to remove drinks I have made from the pending list so that I don't make duplicates

DELETE

```
DELETE /order/1234 HTTP 1.1
Host: starbucks.example.org

200 OK
```

Enjoy