RESTful Web Services

20-Jan-2011

Gordon Dickens
Chariot Solutions
gdickens@chariotsolutions.com

CHARIOT SOLUTIONS

Who Am I?

- Instructor/Mentor at chariotsolutions.com/education

CHARIOT SOLUTIONS

Chariot Education Services
Public & Private Training
One-On-One Mentoring
Lunch and Learns

- Active Tweeter for Open Source Tech Topics twitter.com/gdickens

- Certified Instructor for springsource A division of vmware

- DZone Most Valuable Blogger
  dzone.com/page/mvbs – dzone.com/user/284679
  Technophile Blog - technophile.gordondickens.com

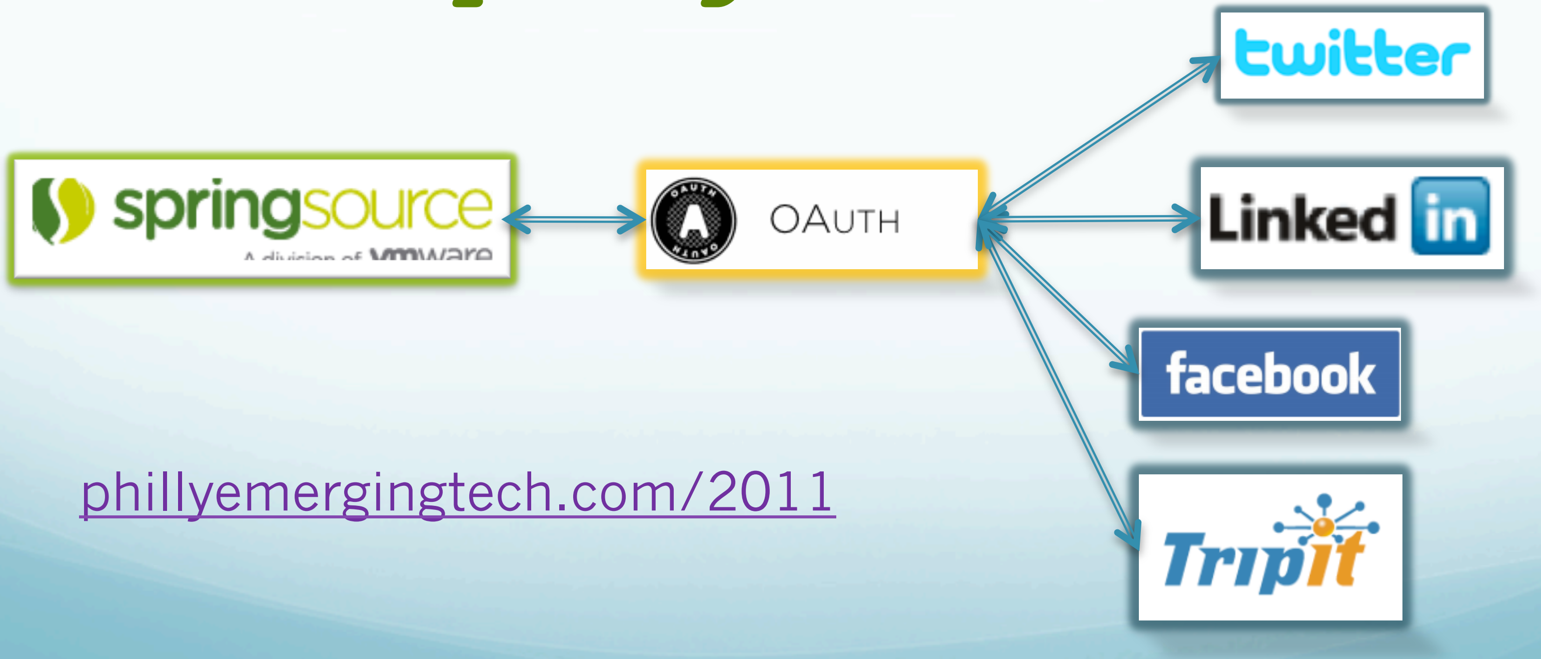
DZone

EMERGING TECHNOLOGIES
FOR THE ENTERPRISE APRIL 27-28, 2011
PHILADELPHIA, PA

- I am speaking about:

Spring Social

Spring Greenhouse



phillyemergingtech.com/2011

Web Services

- What are Web Services?
  - SOA
  - Remote messaging between systems

- SOAP != Web Services
- Web Services != SOAP
- SOAP !opposite of REST
- REST !opposite of SOAP

Who is using REST?

amazon web services

Meetup

LinkedIn

Google

salesforce

facebook

brightkite people. places. friends.

myspace.com a place for friends

NETFLIX

flickr

TripIt

plaxo

twitter

friendfeed

GUINNESS

yammer

What is REST?

- REpresentational State Transfer
  - term by Roy Fielding
  - en.wikipedia.org/wiki/Representational_State_Transfer
  - www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- Architectural Style
  - Design principle
  - Not an API
  - Not a standard
- Web Services over HTTP
  - Client: Browser, Desktop, Mobile Device, etc.
  - HTTP supported by most languages & platforms

Re – S – T

- Representational
  - Client requests data AND representation from server
  - HTML, PDF, JSON, XLS, etc.
- State
  - URIs returned in hypermedia are in context of the current resource
  - Available options for the client embedded within
  - View state to edit state
- Transfer
  - The server transfers hypermedia content to the client
- HATEOAS
  - Cool resume building buzzword
  - **H**ypermedia as the **E**ngine of **A**pplication **S**tate

REST Benefits

- Representations can be any format
  - JSON, XML, PDF, JPG, HTML, etc.
  - Client requests standard media type
- Hypermedia
  - response contains resource specific links
  - provides state transitions
- Cacheable
- HTTP
  - Existing Infrastructure
  - Language Support

REST Introduction

- Take the ROAD back to OOAD
- **Nouns** are defined in the URI
- **Verbs** are provided by HTTP
  - GET (retrieve)
  - POST (create)
  - PUT (update)
  - DELETE (delete)
- What should the server return from this URI?
  - http://myserver:8080/myapp/accounts/234

What's our Job?

- We must design the URI patterns & flow
- Define URIs with Nouns
- Include identifiers as `Path Variables`
    - GET /accounts/234
    - GET /accounts/234/orders
    - GET /accounts/234/orders/25
- `Parameters` provide hints to the server such as pagination values or max rows, etc.

URI Design

- For each Resource (noun) define behavior
- **Account**:
  - GET /accounts – returns list of accounts
  - GET /accounts/{id} – returns account by id
  - POST /accounts – inserts account data
  - PUT /accounts/{id} – Updates account by id
  - DELETE /accounts/{id} – Close account by id
- Account’s **Orders**
  - GET /accounts/{id}/orders – List all orders for account
  - GET /accounts/{id}/orders/{id} – List specific order
  - POST /accounts/{id}/orders – Insert order for account
  - PUT /accounts/{id}/orders/{id} – Update order
  - DELETE /accounts/{id}/orders/{id} – Cancel Order

HTTP Response Codes

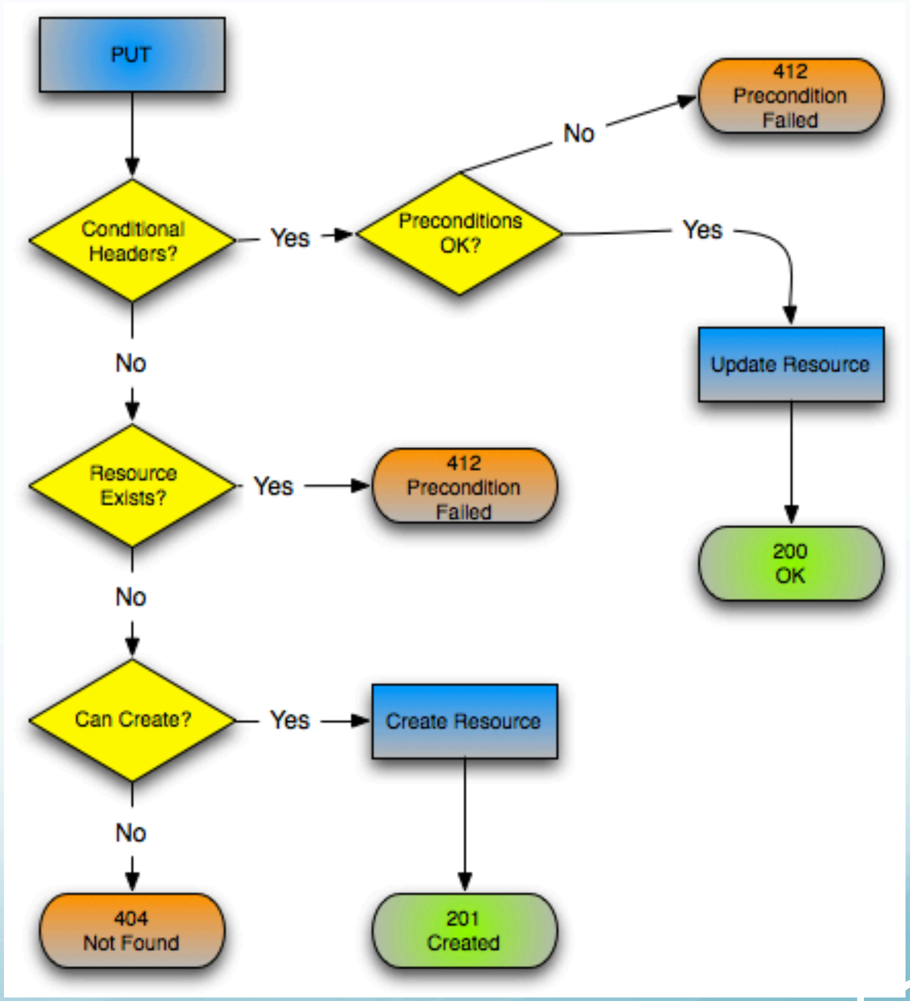
- Familiar HTTP Response codes
  - 200 OK
  - 404 Page not found
  - 500 Server is kaput

- RESTful uses standard HTTP codes
  - 1xx Informational
  - 2xx Success
  - 3xx Redirection
  - 4xx Client Error
  - 5xx Server Error

Designing the flow

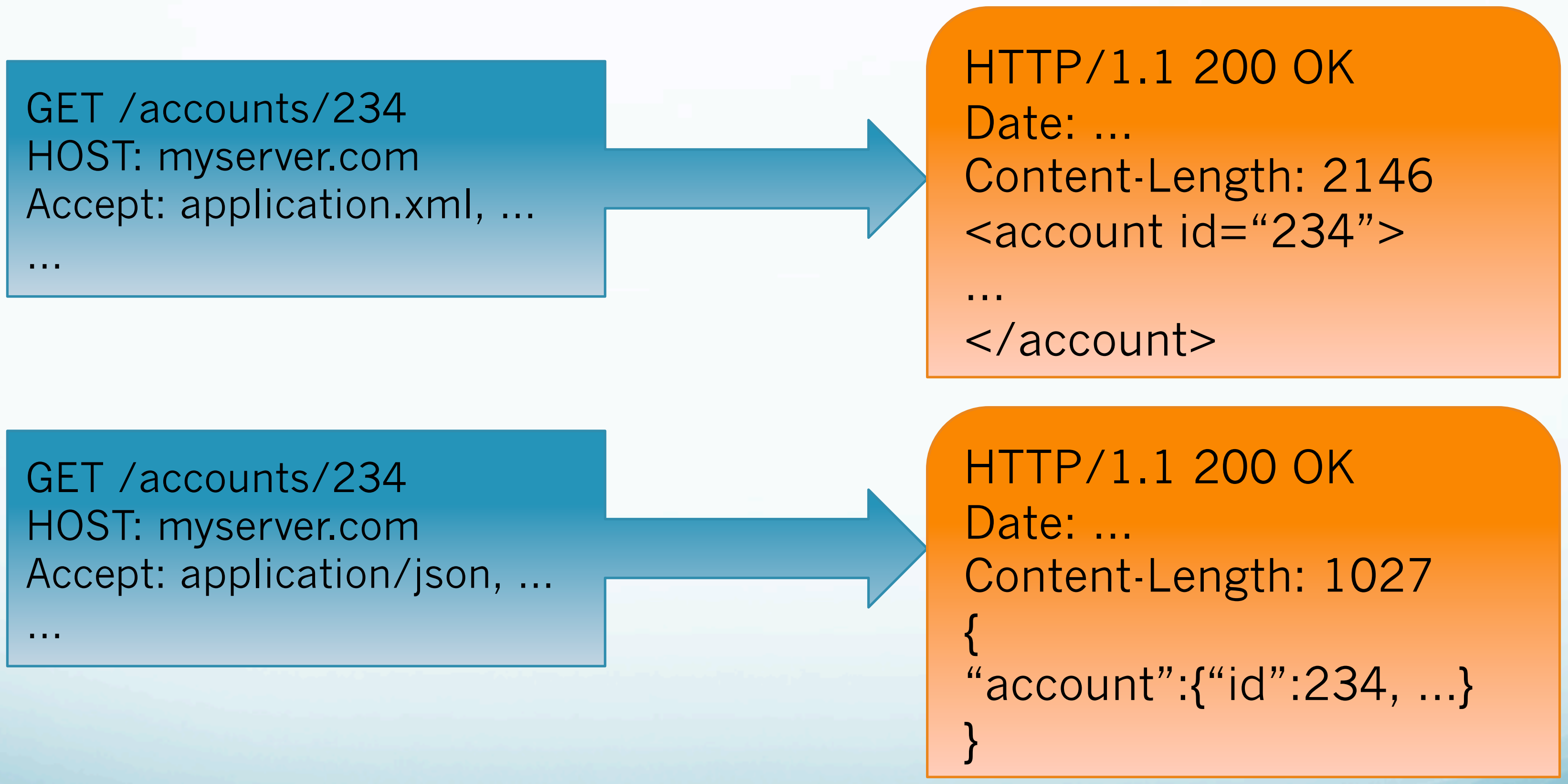
- Conditional Headers
  - If-Unmodified-Since
  - If-Match (etag)
  - If-None-Match

  - Example of Conditional PUT
  - Returns:
    - 200 - OK
    - 201 - Created
    - 404 – Not Found
    - 412 – Precond. Failed



Representation Request

- Client to Server



```
GET /accounts/234
HOST: myserver.com
Accept: application.xml, ...
...
```

```
HTTP/1.1 200 OK
Date: ...
Content-Length: 2146
<account id="234">
...
</account>
```

```
GET /accounts/234
HOST: myserver.com
Accept: application/json, ...
...
```

```
HTTP/1.1 200 OK
Date: ...
Content-Length: 1027
{
"account":{"id":234, ...}
}
```

Java & REST

- JAX-RS
  - JSR-311
  - Jersey (RI), Restlet, CXF, RestEasy
- Spring REST
  - leverages formatters & converters
  - REST Template – Easy Client development
  - Spring Roo – generates RESTful URIs

Spring REST

- Annotations for:
  - URL Path
  - HTTP Verbs
  - Request body (payload)
  - Response body
  - Header, Parameter & Path variables
  - Response Status codes

```
@RequestMapping
@ResponseStatus
@PathVariable
@RequestBody
@ResponseBody
@RequestParam
@RequestHeader
```

- Automatic marshalling/unmarshalling of resource representations
- <mvc:annotation-driven/>
  - Registers automatic formatters, converters & marshallers
  - Inspects classpath for Jackson/JSON, JodaTime, etc.

Security - Data

- Same as other messaging approaches

- Encapsulation:
  - SSL, TLS, IPsec, etc.

- Encryption:
  - PGP, S/MIME, etc.

Security - Auth

- Authentication:
  - Basic, Digest, X509, etc.
  - Spring Security

  - Client Sends

```
GET /account/123
```

  - Server Responds with:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="MyApp Realm"
```

- Authorization
  - Standard web.xml security configuration
  - Spring Security
  - OAuth

"An open protocol to allow secure API authorization in a simple and standard method from desktop to web applications."

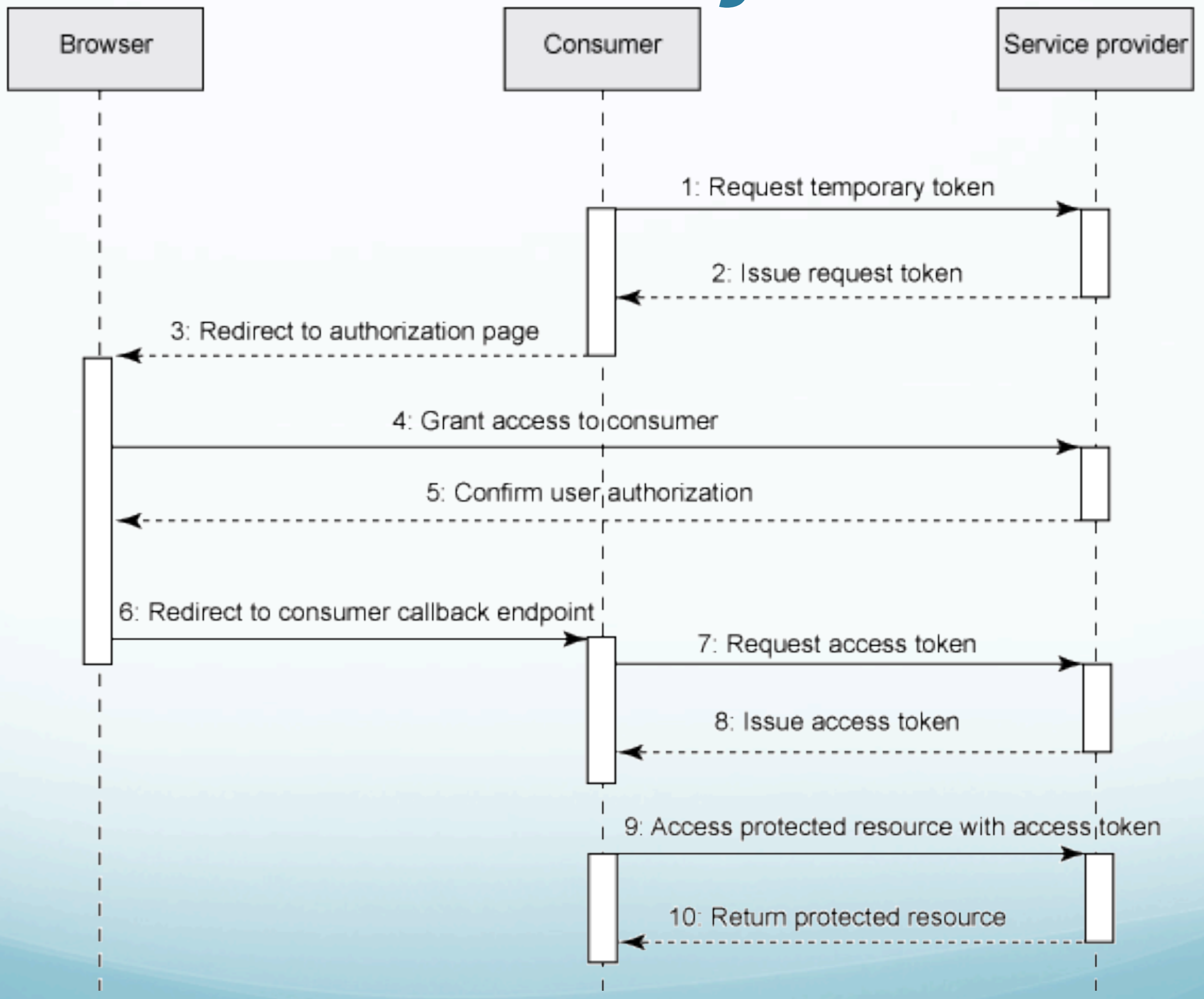
http://oauth.net

code.google.com/p/oauth/

OAuth Participants

- Client
  - Our app
- Server
  - Who we want to connecting with
- Service Provider
  - Service that authenticates credentials

OAuth Safety Dance


Browser
Consumer
Service provider
1: Request temporary token
2: Issue request token
3: Redirect to authorization page
4: Grant access to consumer
5: Confirm user authorization
6: Redirect to consumer callback endpoint
7: Request access token
8: Issue access token
9: Access protected resource with access token
10: Return protected resource

OpenID®

"OpenID is a **decentralized authentication protocol** that makes it easy for people to sign up and access web accounts."

openid.net

openid.net/developers/libraries/#java

JOpenID

- OpenID 2.0 Java 5 impl for Google Federated Login

code.google.com/p/jopenid/

dyuproject

- Java REST framework, openid 2.0, OAuth consumer & service provider, JSON IOC

code.google.com/p/dyuproject/

Who is using OpenID?

Questions?

- Instructor/Mentor at chariotsolutions.com/education

Chariot Education Services
Public & Private Training
One-On-One Mentoring
Lunch and Learns

- Active Tweeter for Open Source Tech Topics
twitter.com/gdickens

- Certified Instructor for

- DZone Most Valuable Blogger
dzone.com/page/mvbs – dzone.com/user/284679
Technophile Blog - technophile.gordondickens.com