REST Best Practices

D. Keith Casey, Jr

So who are you?

- D. Keith Casey, Jr
  - General Annoyance, Blue Parabola
  - Developer Evangelist, Twilio
  - Project Lead, Web2Project
  - Community: Helped organize php|tek*3, antagonized DCPHP, agitating in Austin PHP

In the beginning...

- We had single stack applications
  - Self-contained
  - Completely Independent
  - Built for humans by humans

In the un-beginning...

- Web Services
  - SOAP
  - XML-RPC
  - XML over HTTP
  - Other random junk..

Image Credit: Mashery.com

Sanity: REST

- Six Constraints
  - Client-Server
  - Stateless
  - Cacheable
  - Layered System
  - Uniform Interface
  - Code on Demand (optional)

“Strictly RESTful”

REST is not a standard

What REST is not..

- Pretty URLs
- XML over HTTP
- JSON over HTTP

“-ilities”

accessibility accountability accuracy adaptability administrability affordability agility auditability autonomy availability credibility process capabilities compatibility composability configurability correctness customizability debugability degradability determinability demonstrability dependability deployability discoverability distributability durability effectiveness efficiency evolvability extensibility failure transparency fault-tolerance fidelity flexibility inspectability installability Integrity interchangeability interoperability learnability maintainability manageability mobility modifiability modularity nomadicity operability orthogonality portability precision predictability producibility provability recoverability relevance reliability repeatability reproducibility resilience responsiveness reusability robustness safety scalability seamlessness self-sustainability serviceability (a.k.a. supportability) securability simplicity stability standards compliance senility survivability sustainability tailorability testability timeliness traceability ubiquity understandability upgradability usability

“-ilities”

accessibility accountability accuracy adaptability administrability affordability agility auditability autonomy availability credibility process capabilities compatibility composability configurability correctness customizability debugability degradability determinability demonstrability dependability deployability discoverability distributability durability effectiveness efficiency evolvability extensibility failure transparency fault-tolerance fidelity flexibility inspectability installability Integrity interchangeability interoperability learnability maintainability manageability mobility modifiability modularity nomadicity operability orthogonality portability precision predictability producibility provability recoverability relevance reliability repeatability reproducibility resilience responsiveness reusability robustness safety scalability seamlessness self-sustainability serviceability (a.k.a. supportability) securability simplicity stability standards compliance senility survivability sustainability tailorability testability timeliness traceability ubiquity understandability upgradability usability

Client-server

- We get this one
  - By separating the two, we can vary them
  - Web servers & database servers
- Scalability & Reliability

Stateless

- Each request stands on its own
- This is where we struggle
- Sessions, cookies, etc
  - Synchronization
  - Sticky sessions

Stateless

```
curl -X POST 'https://api.twilio.com/
2010-04-01/Accounts/ACxxxx/SMS/
Messages.xml' \
-d 'From=%2B15125551212' \
-d 'To=7035551212' \
-d 'Body=This+is+just+a+test+message+to+see
+what+happens.' \
-u ACxxxx:{AuthToken}
```


Stateless - Why?

- It's WEB SCALE
  - Stability
  - Reliability
  - Flexibility

Cacheable

- GET, PUT, and DELETE should be idempotent or “safe”
  - The word "safe" means that if a given HTTP method is invoked, the resource state on the server remains unchanged.
- POST... stupid POST

... wha?

- Within Twilio SMS:
  - /2010-04-01/Accounts/{AccountSid}/SMS/Messages

- GET {optional: To, From, DateSent}
- POST {required: To, From, Body ; optional: StatusCallback, ApplicationSid}
- PUT n/a
- DELETE n/a

... wha?

- Within Twilio Voice Recordings:
  - /2010-04-01/Accounts/{AccountSid}/Recordings/{RExxx}

- GET {none}
- POST n/a
- PUT n/a
- DELETE {none}

Layered System

- Don't count on the Client communicating directly to the Server
- We use this on the web every single day
- Adds silent, invisible dependencies

Layered System - Why?

- Don't count on the Client communicating directly to the Server
  - Allows
    - Load Balancers, Caches
    - Logging, Audit trails
    - Authentication & Authorization

Skynet Day

Ref: http://www.twilio.com/engineering/2011/04/22/why-twilio-wasnt-affected-by-todays-aws-issues

Code on Demand

(optional)

- A request doesn't just retrieve a resource but also the code to act upon it

- We don't have to know or understand the code, just how to run it

- Allows for flexibility, upgradability

Ummm... gmail?

Uniform Interfaces

- Four Principles
  - Identification of Resources
  - Manipulation of Resources through these Representations
  - Self-descriptive Messages
  - Hypermedia as the engine of application state (HATEOAS)

Identification of Resources

- Generally
  - /noun/id
  - /noun/action/id
- But not required
  - /?n=noun&id=id
  - /?n=noun&a=action&id=id

Manipulation through those Interfaces

- Within Twilio:
  - /2010-04-01/Accounts/{AccountSid}/Calls/{CAxxx}
  - /2010-04-01/Accounts/{AccountSid}/Conferences/{CFxxx}
  - /2010-04-01/Accounts/{AccountSid}/Notifications/{NOxxx}
  - /2010-04-01/Accounts/{AccountSid}/Recordings/{RExxx}
  - /2010-04-01/Accounts/{AccountSid}/SMS/{SMxxx}
  - /2010-04-01/Accounts/{AccountSid}/Transcripts/{TRxxx}

    - GET {none}
    - POST {only for Calls & SMS}
    - PUT n/a
    - DELETE {only for Recordings}

Self Descriptive

- Each message should tell you:
  - how to process itself;
  - how to request the next resource;
  - if that resource is cachable;

HATEOAS

Clients make state transitions only through actions that are dynamically identified within hypermedia by the server (e.g. by hyperlinks within hypertext). Except for simple fixed entry points to the application, a client does not assume that any particular actions will be available for any particular resources beyond those described in representations previously received from the server.

Source: http://en.wikipedia.org/wiki/Representational_state_transfer#RESTful_web_services

HATEOAS - not good

```
  $ curl -I https://api.github.com/
HTTP/1.1 302 Found
Server: nginx/1.0.4
Content-Type: text/html;charset=utf-8
Connection: keep-alive
Status: 302 Found
X-RateLimit-Limit: 5000
Location: http://developer.github.com
X-RateLimit-Remaining: 4993
Content-Length: 0
```


HATEOAS - good

```
  $ curl https://api.twilio.com/2010-04-01
<?xml version="1.0"?>
<TwilioResponse>
  <Version>
    <Name>2010-04-01</Name>
    <Uri>/2010-04-01</Uri>
    <SubresourceUris>
      <Accounts>/2010-04-01/Accounts</Accounts>
    </SubresourceUris>
  </Version>
</TwilioResponse>
```


HATEOAS - more good

```
  <TwilioResponse>
<Account>
  <Sid>ACxxxx</Sid>
  <FriendlyName>Do you like my friendly name?</FriendlyName>
  <Type>Full</Type>
  <Status>active</Status>
  <DateCreated>Wed, 04 Aug 2010 21:37:41 +0000</DateCreated>
  <DateUpdated>Fri, 06 Aug 2010 01:15:02 +0000</DateUpdated>
  <AuthToken>redacted</AuthToken>
  <Uri>/2010-04-01/Accounts/ACxxxx</Uri>
  <SubresourceUris>
    <AvailablePhoneNumbers>/2010-04-01/Accounts/ACxxxx/AvailablePhoneNumbers</AvailablePhoneNumbers>
    <Calls>/2010-04-01/Accounts/ACxxxx/Calls</Calls>
    <Conferences>/2010-04-01/Accounts/ACxxxx/Conferences</Conferences>
    <IncomingPhoneNumbers>/2010-04-01/Accounts/ACxxxx/IncomingPhoneNumbers</IncomingPhoneNumbers>
    <Notifications>/2010-04-01/Accounts/ACxxxx/Notifications</Notifications>
    <OutgoingCallerIds>/2010-04-01/Accounts/ACxxxx/OutgoingCallerIds</OutgoingCallerIds>
    <Recordings>/2010-04-01/Accounts/ACxxxx/Recordings</Recordings>
    <Sandbox>/2010-04-01/Accounts/ACxxxx/Sandbox</Sandbox>
    <SMSMessages>/2010-04-01/Accounts/ACxxxx/SMS/Messages</SMSMessages>
    <Transcriptions>/2010-04-01/Accounts/ACxxxx/Transcriptions</Transcriptions>
  </SubresourceUris>
 </Account>
</TwilioResponse>
```


HATEOAS - more good

```
  <TwilioResponse>
<Account>
  <Sid>ACxxxx</Sid>
  <FriendlyName>Do you like my friendly name?</FriendlyName>
  <Type>Full</Type>
  <Status>active</Status>
  <DateCreated>Wed, 04 Aug 2010 21:37:41 +0000</DateCreated>
  <DateUpdated>Fri, 06 Aug 2010 01:15:02 +0000</DateUpdated>
  <AuthToken>redacted</AuthToken>
  <Uri>/2010-04-01/Accounts/ACxxxx</Uri>
  <SubresourceUris>
    <AvailablePhoneNumbers>/2010-04-01/Accounts/ACxxxx/AvailablePhoneNumbers</AvailablePhoneNumbers>
    <Calls>/2010-04-01/Accounts/ACxxxx/Calls</Calls>
    <Conferences>/2010-04-01/Accounts/ACxxxx/Conferences</Conferences>
    <IncomingPhoneNumbers>/2010-04-01/Accounts/ACxxxx/IncomingPhoneNumbers</IncomingPhoneNumbers>
    <Notifications>/2010-04-01/Accounts/ACxxxx/Notifications</Notifications>
    <OutgoingCallerIds>/2010-04-01/Accounts/ACxxxx/OutgoingCallerIds</OutgoingCallerIds>
    <Recordings>/2010-04-01/Accounts/ACxxxx/Recordings</Recordings>
    <Sandbox>/2010-04-01/Accounts/ACxxxx/Sandbox</Sandbox>
    <SMSMessages>/2010-04-01/Accounts/ACxxxx/SMS/Messages</SMSMessages>
    <Transcriptions>/2010-04-01/Accounts/ACxxxx/Transcriptions</Transcriptions>
  </SubresourceUris>
 </Account>
</TwilioResponse>
```


HATEOAS - more good

```
  <TwilioResponse>
<Account>
  <Sid>ACxxxx</Sid>
  <FriendlyName>Do you like my friendly name?</FriendlyName>
  <Type>Full</Type>
  <Status>active</Status>
  <DateCreated>Wed, 04 Aug 2010 21:37:41 +0000</DateCreated>
  <DateUpdated>Fri, 06 Aug 2010 01:15:02 +0000</DateUpdated>
  <AuthToken>redacted</AuthToken>
  <Uri>/2010-04-01/Accounts/ACxxxx</Uri>
  <SubresourceUris>
    <AvailablePhoneNumbers>/2010-04-01/Accounts/ACxxxx/AvailablePhoneNumbers</AvailablePhoneNumbers>
    <Calls>/2010-04-01/Accounts/ACxxxx/Calls</Calls>
    <Conferences>/2010-04-01/Accounts/ACxxxx/Conferences</Conferences>
    <IncomingPhoneNumbers>/2010-04-01/Accounts/ACxxxx/IncomingPhoneNumbers</IncomingPhoneNumbers>
    <Notifications>/2010-04-01/Accounts/ACxxxx/Notifications</Notifications>
    <OutgoingCallerIds>/2010-04-01/Accounts/ACxxxx/OutgoingCallerIds</OutgoingCallerIds>
    <Recordings>/2010-04-01/Accounts/ACxxxx/Recordings</Recordings>
    <Sandbox>/2010-04-01/Accounts/ACxxxx/Sandbox</Sandbox>
     <SMSMessages>/2010-04-01/Accounts/ACxxxx/SMS/Messages</SMSMessages>
    <Transcriptions>/2010-04-01/Accounts/ACxxxx/Transcriptions</Transcriptions>
  </SubresourceUris>
 </Account>
</TwilioResponse>
```


HATEOAS - more good

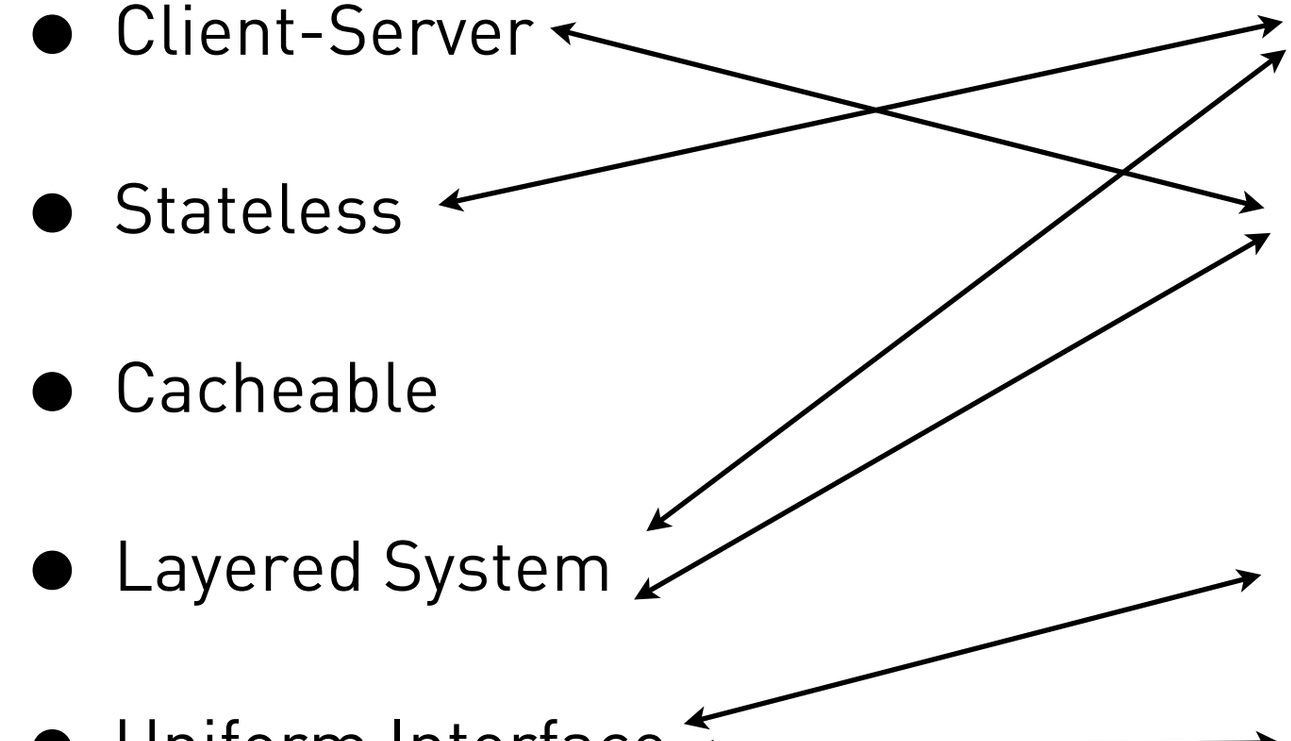
```
  <TwilioResponse>
<Account>
  <Sid>ACxxxx</Sid>
  <FriendlyName>Do you like my friendly name?</FriendlyName>
  <Type>Full</Type>
  <Status>active</Status>
  <DateCreated>Wed, 04 Aug 2010 21:37:41 +0000</DateCreated>
  <DateUpdated>Fri, 06 Aug 2010 01:15:02 +0000</DateUpdated>
  <AuthToken>redacted</AuthToken>
  <Uri>/2010-04-01/Accounts/ACxxxx</Uri>
  <SubresourceUris>
    <AvailablePhoneNumbers>/2010-04-01/Accounts/ACxxxx/AvailablePhoneNumbers</AvailablePhoneNumbers>
    <Calls>/2010-04-01/Accounts/ACxxxx/Calls</Calls>
    <Conferences>/2010-04-01/Accounts/ACxxxx/Conferences</Conferences>
    <IncomingPhoneNumbers>/2010-04-01/Accounts/ACxxxx/IncomingPhoneNumbers</IncomingPhoneNumbers>
     <Notifications>/2010-04-01/Accounts/ACxxxx/Notifications</Notifications>
    <OutgoingCallerIds>/2010-04-01/Accounts/ACxxxx/OutgoingCallerIds</OutgoingCallerIds>
    <Recordings>/2010-04-01/Accounts/ACxxxx/Recordings</Recordings>
    <Sandbox>/2010-04-01/Accounts/ACxxxx/Sandbox</Sandbox>
    <SMSMessages>/2010-04-01/Accounts/ACxxxx/SMS/Messages</SMSMessages>
    <Transcriptions>/2010-04-01/Accounts/ACxxxx/Transcriptions</Transcriptions>
  </SubresourceUris>
 </Account>
</TwilioResponse>
```


-ilities

accessibility accountability accuracy adaptability administrability affordability agility auditability autonomy availability credibility process capabilities compatibility composability configurability correctness customizability debugability degradability determinability demonstrability dependability deployability discoverability distributability durability effectiveness efficiency evolvability extensibility failure transparency fault-tolerance fidelity flexibility inspectability installability Integrity interchangeability interoperability learnability maintainability manageability mobility modifiability modularity nomadicity operability orthogonality portability precision predictability producibility provability recoverability relevance reliability repeatability reproducibility resilience responsiveness reusability robustness safety scalability seamlessness self-sustainability serviceability (a.k.a. supportability) securability simplicity stability standards compliance sterility survivability sustainability tailorability testability timeliness traceability ubiquity understandability upgradability usability

REST vs OOP

- REST Constraints
  - Client-Server
  - Stateless
  - Cacheable
  - Layered System
  - Uniform Interface
  - Code on Demand (optional)
- OOP Principles
  - Single Responsibility
  - Open/Closed
  - Liskov Substitution
  - Interface Segregation
  - Dependency Inversion



Additional Resources

(no pun intended)

- http://en.wikipedia.org/wiki/HATEOAS
- http://blog.steveklabnik.com/2011/07/03/nobody-understands-rest-or-http.html - Steve Klabnik
- http://shop.oreilly.com/product/9780596529260.do
- http://videos.restfest.org
- http://devzone.zend.com/1915/solid-oo-principles/

D. Keith Casey, Jr.

keith@twilio.com

keith@blueparabola.com

keith@caseysoftware.com

caseysoftware just about everywhere online

For Twilio Txt: [redacted]