

*Distributed Software Development*  
*Using Amazon's Web Services with REST*

Chris Brooks

Department of Computer Science  
University of San Francisco

## 11-2: Review: REST

---

- REST Takes a data-oriented approach to web services
- Rather than specifying how a client should interact with a service, we specify a reference to a data object in the form of a URI.
- Web as a shared information space, rather than as a medium for transporting messages between hosts.
  - Argument: the rest of the Web operates according to REST, so Web Services should as well.

## 11-3: REST

---

- REST stands for Representational State Transfer
  - Idea: Applications work with a representation of a resource (i.e. an XML representation of a song)
  - These representations are shared, or transferred between components.
  - These representations allow client applications to manage their state.
- Data-centric: all services and resources can be referenced with URIs.
- Servers respond to a request by providing a representation of an object.

## 11-4: REST

---

- REST is really more of an architectural model than a protocol.
  - A recipe for building web-scale applications
- In practice, it refers to:
  - encoding requests within an URI
  - using HTTP to deliver them
  - returning results via XML.

## 11-5: Using REST with Amazon.com

---

- We'll begin with Amazon's REST interface for this lab
  - Gentle introduction to web services from the client side.
  - Relatively well-documented.
  - No funky third-party libs needed.
- At its essence, A RESTful program to interface with Amazon just needs to open and read a URI, then parse the resulting XML.

## 11-6: An example

---

```
#!/usr/bin/python
import urllib
from xml.dom import minidom
## a string that holds the base URL and my subscription ID.
base='http://webservices.amazon.com/onca/xml?
Service=AWSECommerceService&SubscriptionId=00DZ9HPDQ8Z2R2WPWCG2'
## open the URI and fetch the contents
returnstr = urllib.urlopen(base +
 '&Operation=ItemSearch&SearchIndex=Books&Keywords=buffy').read()
## returnstr is XML - let's parse it and find all the titles.
xmldoc = minidom.parseString(returnstr)
for node in xmldoc.getElementsByTagName('Title') :
    print node.firstChild.data
```

## 11-7: URI format

---

- The URI consists of two parts:
  - A base, which is everything before the '?' character.
  - A set of key/value pairs, which is everything after the '?'.
    - Separated by '&'
  - Your program needs to construct a URI using the proper base and keys.

## 11-8: Using Amazon's Web service

---

- First, register with Amazon to get a SubscriptionId.
- Data available:
  - Product data
  - Customer content (lists, reviews)
  - Product listings, including third-party marketplaces.
  - Shopping carts
- Some of this is more helpful if you want to set up your own e-Commerce site that uses Amazon's services.

## 11-9: Making REST requests to Amazon

---

- The base URI is:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
```

- The remainder of the URI is composed of *request parameters*.
- These indicate your ID, the type of operation you want performed, and other parameters relevant to the search.

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
    &SubscriptionId=[your subscription ID here]
    &Operation=ItemSearch
    &SearchIndex=Books
    &Keywords=buffy
```

## 11-10: Types of operations

---

- ItemLookup - get information associated with an ItemId
- ItemSearch - get information associated with a keyword(s)
- SimilarityLookup - find items similar to a given ItemId
- BrowseNodeLookup - find the 'browseNode' associated with a given ItemId
- List Lookup/Search - find wish lists or Listmania lists.
- Shopping Cart operations
- Seller lookup - get seller information for a given seller ID.

## 11-11: Amazon's Data Model

---

- The data returned by Amazon consists of two parts:
  - OperationRequest - gives the parameters received by Amazon
  - Data - gives the result of the query
- The exact structure depends on the request - each response has a different element.

## 11-12: Amazon Product Data

---

- Product data is contained within an 'Items' element.
- This contains a 'Request' element that indicates the input parameters used to generate the reply.
- 'Total pages' indicates the number of pages
- 'Item' contains data for each product.
- The only required subelement is ASIN - most products have many other subelements.

## 11-13: Examples

---

- Find books related to 'buffy'

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&  &SubscriptionId=[your subscription ID here]
    &Operation=ItemSearch
    &SearchIndex=Books&Keywords=buffy
```

- Find information on a particular Buffy book.

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&  &SubscriptionId=[your subscription ID here]
    &Operation=ItemLookup
    &ItemId=1569714290
```

- Find items similar to this Buffy book:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&  &SubscriptionId=[your subscription ID here]
    &Operation=SimilarityLookup
    &ItemId=1569714290
```

## 11-14: responseGroups

---

- You can also specify what data you would like to get back from a request.
- Small, medium, large
- Specific elements
- To get images:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&  &SubscriptionId=[your subscription ID here]
    &Operation=ItemLookup
    &ItemId=1569714290&ResponseGroup=Images
```

- To get all info:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&  &SubscriptionId=[your subscription ID here]
    &Operation=ItemLookup
    &ItemId=1569714290&ResponseGroup=Large
```

## 11-15: Other data

---

- Browse nodes provide access to category information.
- Listmania nodes let you get access to user-created lists.
- CartLookup lets you access your shopping cart
- SimilarityLookup - find similar items.

## 11-16: Troubleshooting

---

- You are limited to one request per second per IP address.
- If you're not getting the results you expect, make sure you're specifying the correct response groups.
- Don't forget your SubscriptionId
- You can test out REST queries in your browser.

## 11-17: Summary

---

- REST is a data-centric way of viewing Web Services
- Every resource or object is represented by a URI.
- Advantages:
  - Integrates into the rest of the Web
  - Easy to use
  - No specialized third-party code needed, except for an XML parser.
- Disadvantages:
  - Working with URIs may be unwieldy for complex data structures.
  - Most useful for data retrieval applications
  - Harder to use with applications that require two-way exchange with a server.