Web Services

CS 360 Internet Programming

Daniel Zappala

Brigham Young University
Computer Science Department
Service Oriented Architecture

- **loose coupling among interacting software agents**
  1. A small set of simple and ubiquitous interfaces to all participating software agents. Only generic semantics are encoded at the interfaces. The interfaces should be universally available for all providers and consumers.
  2. Descriptive messages constrained by an extensible schema delivered through the interfaces. Only minimal system behavior is prescribed by messages. A schema limits the vocabulary and structure of messages. An extensible schema allows new versions of services to be introduced without breaking existing services.

  – [XML.com](http://xml.com)

- similar in philosophy to the design of Unix command-line utilities
Web Services

- **definition**: self-contained, self-describing, modular applications that can be published, located, and invoked programmatically across the Web
  - typically run over HTTP: operate through firewalls
  - typically use XML for message format
- **motivation**: access web databases programmatically rather than via web pages and forms
- **goals**: distributed applications, interoperability, open standards, software re-use
Web Services Architecture

- **UDDI registry**: lists available services
- **WSDL**: description language for services
- **SOAP**: protocol for obtaining services using XML messages
Service Discovery

Universal Discovery, Description and Integration (UDDI)
- platform-independent, XML-based registry listing available web services

Vision
- meant to be a place where service providers can advertise available services and do business with partners
- service registry could become a competitive business

Reality
- businesses use UDDI internally to keep track of thousands of available web services
**Service Description**

- **Web Services Description Language (WSDL)**
  - XML format for describing web services
  - parts of a WSDL document
    - namespace
    - documentation
    - types: types of messages
    - interface: abstract interface - operations and pattern
    - binding: how messages are exchanged (e.g. SOAP)
    - service: where the service is available

- standardized by W3C:
  - [Web Services Description Working Group](http://www.w3.org/2004/10/wstc)

- example: see Section 2.1 of the [WSDL Version 2.0 Primer](http://www.w3.org/TR/wsdl)
Services

- **SOAP**
  - format for XML messages
  - description of service must be in WSDL

- **REST**
  - uses current web architecture
  - uses standard HTTP GET, PUT, DELETE, POST
  - uses XML messages or URL encoding
Simple Object Access Protocol (SOAP)

- protocol for exchanging XML-based messages, usually over HTTP
- uses XML namespaces
- message contained in an envelope
  - header
  - body
SOAP Example

1. `<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"/>
2.   <soap:Body>
3.     <getProductDetails xmlns="http://warehouse.example.com/ws">
4.       <productID>827635</productID>
5.     </getProductDetails>
6.   </soap:Body>
7. </soap:Envelope>`
SOAP Example Continued

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns="http://warehouse.example.com/ws">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Toptimate 3–Piece Set</productName>
        <productID>827635</productID>
        <description>3–Piece luggage set. Black Polyester.</description>
        <price>96.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```
Representational State Transfer (REST)

- web services using the existing web architecture
  - observation: everything we need to do with web services is already supported in HTTP
  - simply need to add XML vocabularies
- based on the concept of a resources, identified by URIs
- representation: data that encodes information about resource state, e.g. HTML, XML
  - GET: obtain a representation of a resource
  - DELETE: remove a representation of a resource
  - POST: update or create a representation of a resource
  - PUT: create a representation of a resource
fictitional parts company, Parts Depot, deploys a web service

- get a list of parts
- get detailed information on a particular part
- submit a purchase order
REST Example: Get a List of Parts

2. 
3. <?xml version="1.0"?>
          xmlns:xlink="http://www.w3.org/1999/xlink">
5.     <Part id="00345"
6.     <Part id="00346"
7.     <Part id="00347"
8.     <Part id="00348"
9.   </p:Parts>
REST Example: Get Information on a Part

1 http://www.parts-depot.com/parts/00345
2
3 <?xml version="1.0"?>
5     xmlns:xlink="http://www.w3.org/1999/xlink">
6     <Part-ID>00345</Part-ID>
7     <Name>Widget-A</Name>
8     <Description>This part is used within the frap assembly</Description>
9     <Specification
11     <UnitCost currency='USD'>0.10</UnitCost>
12     <Quantity>10</Quantity>
13 </p:Part>
REST Example: Submit a Purchase Order

- client creates a PO document that conforms to the PO schema that Parts Depot has designed and publicized using WSDL
- the client submits PO.xml as the payload of an HTTP POST
- the PO service responds to the HTTP POST with a URL to the submitted PO
- the PO now exists as a resource for reading, editing, etc.