Web Services

- create a Service Oriented Architecture
- *loose coupling among interacting software agents*
  - agents are generally programs, not users
  - separate data from computing and viewing
- example
  - a company needs to ship some packages overseas, so it uses a program to look up package delivery services, compare prices, purchase the best deal, and schedule pickup
- requires
  - service discovery
  - interfaces
  - standardized and extensible protocols
- [XML.com](http://www.xml.com)
UDDI: Universal Discovery, Description and Integration

- platform-independent, XML-based registry listing available web services
- a place where service providers can advertise available services and do business with partners
**WSDL: Web Services Description Language**

- XML format for describing web services
- standardized by W3C:
  - [Web Services Description Working Group](https://www.w3.org/2001/02/wsdl)
- example: see Section 2.1 of the [WSDL Version 2.0 Primer](https://www.w3.org/TR/wsdl20primer/)

---

**Web Services Architecture**

- UDDI Registry
- Points to description
- WSDL
- Points to Service
- Describes Service
- Web Service
- SOAP: Communicates with XML messages
- Service Consumer
- Finds Service

---

**Introduction**

**Examples**

Daniel Zappala 3/7
Web Services Architecture

- **SOAP**: Simple Object Access Protocol
  - protocol for obtaining services using XML messages
  - description of service must be in WSDL
SOAP Example

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <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 or JSON formats for results
- based on the concept of a resources, identified by URIs
- representation: data that encodes information about resource state, e.g. HTML, XML, JSON
  - 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
**REST Example: FamilySearch**

```xml
<?xml version='1.0' encoding='utf-8'?>
<familytree xmlns="http://api.familysearch.org/familytree/v2">
<persons>
  <person id="KW3B-2DB" requestedId="KW3B-2DB" version="25770197004">
    <assertions>
      <names>
        <name>
          <value type="Name">
            <forms>
              <form>
                <fullText>John Henry Doe</fullText>
                <pieces>
                  <piece type="Given">
                    <predelimiters></predelimiters>
                    <value>John</value>
                    <postdelimiters></postdelimiters>
                  </piece>
                  <piece type="Given">
                    <predelimiters></predelimiters>
                    <value>Henry</value>
                    <postdelimiters></postdelimiters>
                  </piece>
                  <piece type="Family">
                    <predelimiters></predelimiters>
                    <value>Doe</value>
                    <postdelimiters></postdelimiters>
                  </piece>
                </pieces>
              </form>
            </value>
          </name>
        </names>
        ...
      </assertions>
    </person>
  </persons>
</familytree>
```