Contents

Geography 485L/585L - Introductions & Course Outline

Introductions

- Who am I?
- Who are you?
- What brought you here?

We will be working on answering these questions during the first class collaboratory from 5:00-6:15 on Wednesday

Syllabus

Outline

- Instructor
- Description & Objectives
- Class format
- Class Readings
- Evaluation & Grading
- Topics
- Communication

Instructor

Karl Benedict

Director, Earth Data Analysis Center

Research Asst. Professor. University Libraries, Geography Dept.

kbene@edac.unm.edu

Office: Bandelier West, rm. 107

(505) 277-3622 x234

Description & Objectives

- The basic concepts behind web mapping technologies that enable the delivery of maps and mapped data through web browsers
- The Open Standards that facilitate the exchange of map images and geospatial data over the internet
- The use of published standards-based services in desktop mapping applications that implement those standards
- The deployment of standards-based geospatial map and data services that other systems and users may make use of

Class Format

- Online Lecture & in-person collaboratory in each class week
- Focus on hands-on experience with standards, technologies, and capabilities
- Exploratory and problem-based
- Cumulative

Class Readings

- The class readings are a combination of conceptual outlines and reference materials

HTML Manual of Style: A Clear, Concise Reference for Hypertext Markup Language (including HTML5), Fourth Edition (4th Edition). Larry Aronson. Addison-Wesley Professional. 2010.

Beginning Google Maps API 3. Gabriel Svennerberg. Apress. 2010.

OpenLayers 2.10 Beginner's Guide. Erik Hazzard. Packt Publishing. 2011.

Optional *Designing with Web Standards (3rd Edition).* Jeffrey Zeldman & Ethan Marcotte. New Riders Press. 2009.

Evaluation and Grading

Class Grade =

- Weekly Web Portfolio Milestones (including peer review when required)
- "Deep Dive" Assignments
- Exams

Evaluation and Grading

13 Weekly Portfolio Milestones - 40 points at mid-term review, 40 points at final review

4 "Deep Dive" Assignments

- 25 points/assignment (100 pts total)
- Focussed on small project
- Reinforcing lab activities
- Added to your portfolio and added to your portfolio score

Evaluation and Grading

2 Exams

- 100 Points each
- Midterm Exam - Take Home - due 3/12
- Final Exam - Take Home - due 5/14
- Focus on *concepts* and *technical implementation*

Evaluation and Grading

- A: 360 - 400 points
- B: 320 - 359.9 points
- C: 280 - 319.9 points
- D: 240 - 279.9 points
- F: $<$ 240 points

Class Topics

- Internet Mapping Clients: Basic HTML, Javascript, CSS; Google Maps API; OpenLayers javascript library
- Geospatial Services Oriented Architectures (SOA)
- Open Standards: Open Geospatial Consortium (OGC - WMS, WFS, WCS, KML); Extensible Markup Language (XML)
- Desktop client use of Open Standards
- Data sharing/publication using Open Standards

Basics

Outline

- What is Internet Mapping?
- Definitions
- Tools

What is Internet Mapping

Extended Desktop Mapping Use of open standards based remote data and map services in desktop applications

Geospatial Data Sharing Establishing open standards based services to share geospatial data and mapping capabilities over the Internet

Web-client Mapping The delivery of mapping and geospatial data tools through web browsers, again based upon open standards

Definitions

Internet The global computer network of computers that typically connect with each other over TCP/IP

World Wide Web The subset of applications that are run over the Internet, typically using the HTTP protocol in combination with data (HTML, XML, XHTML), presentation (CSS), and behavior (JavaScript) components

Mapping The generation of cartographic products that include map images (pictures of geospatial data) and other elements (e.g. legends, tools, scale information, north-arrow)

Definitions

Analysis The development of models (statistical and otherwise) that enable the exploration of geospatial data and testing of hypotheses using those data

Open Standards While the definition varies from one organization to the next, Open Standards are often characterized by the following:

- Developed through a public process by a national or international standards group
- May be implemented royalty-free

Definitions

Interoperability Ability of systems to share data and information with each other

COTS Commercial Off-the-Shelf Software. Applications that are "purchased" from vendors, often with license terms that restrict the use the software to the specific platform for which it is licensed. Often comes with implicit or explicit technical support

Open Source Software licensed under terms that are consistent with the Open Source definition, which includes access to source code, and freedom to modify and redistribute

Definitions

Data Actual values associated with geographic locations. For example - numeric elevation values associated with locations within a Digital Elevation Model.

Metadata Data about a particular data product or service. Metadata provide critical documentation that supports the discovery and use of data products and data and mapping services

Tools

Server Platform Linux server with GeoServer, Apache, GDAL and Proj libraries

Operating System (one of the following) Microsoft Windows Vista or 7

Mac OS 10.6 or above

Linux (*speak to Dr. Benedict*)

Geographic Information System (GIS) Quantum GIS (platform specific download)

ArcGIS 10 (*optional* - request free student version installation CD from Dr. Benedict - *Windows Only*)

Tools

Geographic Data Processing/Analysis (one of the following) FWTools (*Windows* & *Linux* - free download)

GDAL and related frameworks (*Mac* - the current "GDAL Complete" convenience package available here)

Text Editor Notepad (*Windows* - included with operating system)

Notapad++ (*Windows* - free download)

TextEdit (*Mac* - included with operating system)

TextWrangler (*Mac* - free download)

Tools

Secure File Transfer Protocol Client WinSCP (*Windows* - free download)
Fugu (*Mac* - free download)

Secure Shell (SSH) Client PuTTY (*Windows* - free download)
Terminal (*Mac* - included with operating system)

Web Browser (at least one of the following) Firefox (*All Operating Systems* - free download)
Chrome (*All Operating Systems* - free download)

Communication

This is the first iteration with the class as a hybrid course, so the most productive communication model will evolve over the semester, but I commit to the following:

- I will respond to email questions within ~24 hours
- I will share responses to common questions with the rest of the class through the online discussion board

I also *strongly* encourage that questions be submitted through the discussion board so that other students can both *learn from* and *contribute to* the answers provided.

Module 2.1 - Introduction to HTML, CSS, and Javascript

Overview

- Web Development
- Parts of a web page
- Web Site Components
  - Structure (X/HTML)
  - Presentation (CSS)
  - Behavior (Javascript)
- Simple Web Pages
- More Complete Web Page Example

Web Development

- Requirements
  - Web Server
  - File location that the web server accesses for requested content
  - Files must be readable by all users
- General Process
  - Create basic content in HTML or XHTML (structure)
  - Change appearance of content through the definitions of styles using CSS (presentation)
  - Add dynamic capabilities to content through Javascript (behavior)
  - REPEAT over and over and over and over again

Parts of a Web Page

```
<html>
    <!-- The HTML block is the container for all of your page content -->
    <head>
        <!-- The head is where you include pointers to external resources
        (i.e. style sheets and javascript files), blocks of Javascript code
        , styles, etc.  -->
        <title>The page title also goes in here</title>
    </head>
    <body>
        <!-- The body is where you put all of the content for the page
        (i.e. the material that will be displayed in the web browser)  -->
        <h1>Headers</h1>
        <div>Generic blocks of content</div>
        <p>Paragraphs</p>
        <table>Tables</table>
        <img ...>Images</img>
        <form ...>Forms</form>
        <ul>Unordered Lists</ul>
        <ol>Ordered Lists</ol>
        <li>List Items</li>
    </body>
</html>
```

Web Site Components - Structure

Content is defined in terms of the structural elements available in HTML/XHTML

- Sample HTML/XHTML Tags
  - Paragraphs (i.e. blocks of text) are contained within `<p>...</p>` tags
  - Headings (i.e. section headings, sub-headings) are contained within numerically defined header tags: `<h1>...</h1>`, `<h2>...</h2>`, `<h3>...</h3>`, etc.
  - Tabular data are within `<table>...</table>` tags
  - List are specified within `<ol>...</ol>` or `<ul>...</ul>` tags, depending upon whether the list is ordered (numbered) or unordered (e.g. bulleted)
  - User input elements are put within `<form>...</form>` tags
  - Blocks of content (i.e. sections or divisions) are defined within `<div>...</div>` tags
- Structure is translated into the Document Object Model (DOM) for later use by CSS and Javascript

Web Site Components - Presentation

Modifications to default rendering of HTML/XHTML elements are made through styles defined in CSS

- Styles may be
  - defined in an external file that is referenced within the `<head>` block (the preferred method when doing "real" web development)
  - directly defined within the `<head>` block of a web page
  - directly embedded in the elements to which they apply (generally not a "Good Thing")
- When not embedded within an element, a style definition consists of

  - A selector
  - The style definition, enclosed in "curly-brackets", separated by "semi-colons"
  - For example: `h1 {color:red; font-size:18px;}`

CSS Selectors

Selectors may be based on several criteria

- Element name: `h1`, `p`, `table`, `ul`, etc.
  - Element: `<h1>A top level heading</h1>`
  - Selector: `h1 {color:red; font-size:18px}`
- Element ID: a unique name assigned to HTML/XHTML elements within the structure of the document
  - Element: `<p id="para01">Some text goes here</p>`
  - Selector: `#para01 {color:blue; font-size:12px}`
- Class ID: a name assigned to multiple elements which may be modified through reference to their class
  - Element: `<p class="instructions">Here are some instructions</p>`
  - Another Element: `<p class="instructions">Here are some more instructions</p>`
  - Selector: `.instructions {color:red; font-size:12px; text-decoration:blink}`
- Selectors may be combined in a variety of ways

Web Site Components - Behavior

The most interoperable language for adding dynamic behavior to web sites is *Javascript* - supported by most browsers on most operating systems

- A full-fledged programming language
  - A non-trivial undertaking to become proficient in
  - Experience in other programming languages can contribute to learning Javascript
- Defines actions that may be taken on/by DOM elements
- Allows for modification of existing DOM elements, creation of new DOM elements after the page has finished loading from the server, retrieval of new content after page loads
  - An interactive web page that may behave like a local desktop application

Reference Links

- w3schools.com
  - HTML 4.0 / XHTML 1.0 Tag Reference
  - Cascading Style Sheet (CSS) selectors and elements
  - Javascript reference
- World Wide Web Consortium (W3C)
  - HTML and CSS Background
  - HTML and CSS Tutorial Links Page
  - Validators Page
- Webmonkey.com
  - HTML Cheat Sheet
  - CSS Guide

Simple Web Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title>This is a simple web page</title>
    </head>
    <body>
        <h1>They don't get any simpler than this!</h1>
        <p>OK, not much simpler than this.</p>
        <p>Hello World?</p>
    </body>
</html>
```

link to example

Simple Web Page with CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title>This is a simple web page - with styling</title>
        <style type="text/css">
            h1 {color:blue; font-size:large}
            p.para {color:#777777; font-size:small}
            #annoying {color:red; text-decoration:line-through}
        </style>
    </head>
    <body>
        <h1>They don't get any simpler than this!</h1>
        <p class="para">OK, not much simpler than this.</p>
        <p id="annoying" class="para">Hello World?</p>
    </body>
</html>
```

link to example

Simple Web Page with Javascript

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title>This is a simple web page with Javascript</title>
```

```
        <script type="text/javascript">
            function genericAlert() {
                alert("You just did something ...")
                document.getElementById("clickMe").style.color = "red"
            }
        </script>
    </head>
    <body>
        <h1>They don't get any simpler than this!</h1>
        <p>OK, not much simpler than this.</p>
        <p>Hello World?</p>
        <p id="clickMe" onclick="genericAlert();">What happens when you click me?</p>
    </body>
</html>
```

link to example

More Complete Web Page Example

[Figure 1 about here.]

Module 2.2 - Web-based Mapping Clients: Google Maps API - Part 1

Outline

- What is an API
- The Google Maps API
  - Version
  - Reference Information
  - Key Components
  - Examples

What is an API

- API Stands for Application Programming Interface

  An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers. – From Wikipedia: http://en.wikipedia.org/wiki/Api

- The Google Maps API provides an interface for interacting with Google's mapping services from external web applications

Google Maps API Version

- The version of the Google Maps API used in this class is v3 of the Javascript API
  - Freely usable for free applications
  - Subject to Google's Terms of Service
  - No longer requires a Google API key
- Key capabilities in v3
  - Interactive maps based on Google's mapping engine (contrast w. static maps API)
  - Optimized for desktop and mobile platforms and applications

Reference Information

Google Maps API Family http://code.google.com/apis/maps/

Javascript API Home Page http://code.google.com/apis/maps/documentation/javascript/

Javascript Basics Entry Page http://code.google.com/apis/maps/documentation/javascript/basics.html

Javascript API v3 Tutorial Page http://code.google.com/apis/maps/documentation/javascript/tutorial.html

Key Components

- Map object options

Types (required) ROADMAP
SATELLITE
HYBRID
TERRAIN

Latitude and Longitude (required) specification of where the map should initially be centered

Zoom Level (required) 0=global, higher values increasingly local. Limited by map type

Controls

- Available Controls (enabled through map options) default controls
  - Zoom Control
  - Pan Control
  - Scale Control
  - MapType Control
  - Street View Control
- Different control styles may be defined
- Controls may be positioned positioning options
- Custom controls may be defined and attached to fixed location in the map

Overlays

Overlay Types documentation

Marker points depicted by specified or defined icons at locations within the map

Polyline linear features defined by multiple points with a defined style for the line

Polygon closed features defined by multiple points. Supports multi-polygons, and donuts. Line and fill styles may be specified.

(Ground) Overlay Maps Image-based map layers that replace or overlay Google layers - registered to the map coordinates

Overlays (continued)

Info Windows floating content windows for displaying content defined as HTML, a DOM element, or text string

Layers Grouped display content assigned to a specific layer: KmlLayer, FusionTablesLayer, TrafficLayer, BicyclingLayer

Custom Overlays definition of programmatically controlled layers

Services

- Geocoding Service
  - Forward and reverse geocoding:
    * address to LatLon
    * LatLon to Nearest Address
  - May be biased to current viewport, region
- Directions
  - Based upon an origin, destination, and a variety of additional options
  - Available directions and rendered route

Services

- Elevation
  - Delivery of elevation data for locations or paths
- Streetview
  - Integration of Google Streetview within a DOM element
- Maximum Zoom
  - Provides information about the maximum available zoom level

Events

- Events provide the ability to attach custom behaviors to events in the interface. For example:
  - Changing items in the interface as the user zooms in on a map
  - Displaying additional information outside the map when the user clicks a location in the map
  - Synchronizing the behavior of multiple maps as the user interacts with one map
- Requires higher-level Javascript that we will cover in this course

Examples

Simple - Roadmap

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            html { height: 100% }
            body { height: 100%;
                margin: 0px;
                padding: 0px;
                background-color: black;
                color: #CCCCCC;
                text-align: center}
            #map_canvas { width:90%;
                height:80%;
                margin-left:auto;
                margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
            function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var myOptions = {
                zoom: 8,
                center: classroom,
                mapTypeId: google.maps.MapTypeId.ROADMAP
            };
            var map = new google.maps.Map(
                document.getElementById("map_canvas"),
                myOptions);
            }
        </script>
    </head>

    <body onload="initialize()">
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>
    </body>
</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps01.html

Simple - Satellite

```
<!DOCTYPE html>

<html>
    <head>
        <style type="text/css">
            html { height: 100% }
            body { height: 100%;
                margin: 0px;
                padding: 0px;
                background-color: black;
                color: #CCCCCC;
                text-align: center}
            #map_canvas { width:90%;
                height:80%;
                margin-left: auto;
                margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
            function initialize() {
                var classroom = new google.maps.LatLng(35.084280,-106.624073)
                var myOptions = {
                    zoom: 8,
                    center: classroom,
                    mapTypeId: google.maps.MapTypeId.SATELLITE
                };
                var map = new google.maps.Map(
                    document.getElementById("map_canvas"),
                    myOptions);
            }
        </script>
    </head>

    <body onload="initialize()">
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>
    </body>
</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps02.html

Simple - Hybrid

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
```

```
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left: auto;
            margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var myOptions = {
              zoom: 8,
              center: classroom,
              mapTypeId: google.maps.MapTypeId.HYBRID
            };
            var map = new google.maps.Map(
                document.getElementById("map_canvas"),
                myOptions);
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps03.html

Simple - Terrain

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left: auto;
```

```
            margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var myOptions = {
              zoom: 8,
              center: classroom,
              mapTypeId: google.maps.MapTypeId.TERRAIN
            };
            var map = new google.maps.Map(
                document.getElementById("map_canvas"),
                myOptions);
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps04.html

Simple - Hybrid - Zoomed

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left: auto;
            margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
```

```
            var myOptions = {
              zoom: 18,
              center: classroom,
              mapTypeId: google.maps.MapTypeId.HYBRID
            };
            var map = new google.maps.Map(
                document.getElementById("map_canvas"),
                myOptions);
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps05.html

Simple - Zoomed - Modified Controls

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left: auto;
            margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var myOptions = {
              zoom: 18,
              center: classroom,
              mapTypeId: google.maps.MapTypeId.HYBRID,
              zoomControl: true,
              zoomControlOptions: {style: google.maps.ZoomControlStyle.SMALL},
              mapTypeControl: true,
              mapTypeControlOptions: {
```

```
                style: google.maps.MapTypeControlStyle.DROPDOWN_MENU},
              streetViewControl: false
            };
            var map = new google.maps.Map(
                document.getElementById("map_canvas"),
                myOptions);
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps06.html

Markers

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left: auto;
            margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var office = new google.maps.LatLng(35.084506,-106.624899)
            var myOptions = {
              zoom: 18,
              center: classroom,
              mapTypeId: google.maps.MapTypeId.HYBRID
              };
            var map = new google.maps.Map(
              document.getElementById("map_canvas"),
              myOptions);

```

```
            var classroomMarker = new google.maps.Marker({
              position: classroom,
              title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
              });
            classroomMarker.setMap(map);

            var officeMarker = new google.maps.Marker({
              position: office,
              title:"Office, Bandelier West, Room 107"
              });
            officeMarker.setMap(map);
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps07.html

Polyline

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left:
            auto;
            margin-right: auto }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var office = new google.maps.LatLng(35.084506,-106.624899)
            var myOptions = {
              zoom: 18,
              center: classroom,
```

```
                mapTypeId: google.maps.MapTypeId.HYBRID
                };
              var map = new google.maps.Map(
                document.getElementById("map_canvas"),
                myOptions);

              var classroomMarker = new google.maps.Marker({
                position: classroom,
                title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
                });
              classroomMarker.setMap(map);

              var officeMarker = new google.maps.Marker({
                position: office,
                title:"Office, Bandelier West, Room 107"
                });
              officeMarker.setMap(map);

              var officeVisitCoordinates = [
                office,
                new google.maps.LatLng(35.084445,-106.624327),
                new google.maps.LatLng(35.084309,-106.624308),
                classroom
                ];
              var officePath = new google.maps.Polyline({
                path: officeVisitCoordinates,
                strokeColor: "#FF0000",
                strokeOpacity: 1.0,
                strokeWeight: 2
              });
              officePath.setMap(map)
            }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps08.html

Polygon

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
```

```
      padding: 0px;
      background-color: black;
      color: #CCCCCC;
      text-align: center}
    #map_canvas { width:90%;
      height:80%;
      margin-left: auto;
      margin-right: auto }
  </style>
  <script type="text/javascript"
      src="http://maps.google.com/maps/api/js?sensor=false">
  </script>
  <script type="text/javascript">
    function initialize() {
      var classroom = new google.maps.LatLng(35.084280,-106.624073)
      var office = new google.maps.LatLng(35.084506,-106.624899)
      var myOptions = {
        zoom: 18,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.HYBRID
        };
      var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        myOptions);
      var classroomMarker = new google.maps.Marker({
        position: classroom,
        title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
        });
      classroomMarker.setMap(map);
      var officeMarker = new google.maps.Marker({
        position: office,
        title:"Office, Bandelier West, Room 107"
        });
      officeMarker.setMap(map);
      var buildingCoordinates = [
        new google.maps.LatLng(35.084498,-106.624921),
        new google.maps.LatLng(35.084558,-106.624911),
        new google.maps.LatLng(35.084566,-106.624970),
        new google.maps.LatLng(35.084609,-106.624966),
        new google.maps.LatLng(35.084544,-106.624383),
        new google.maps.LatLng(35.084438,-106.624317),
        new google.maps.LatLng(35.084384,-106.623922),
        new google.maps.LatLng(35.084164,-106.623970),
        new google.maps.LatLng(35.084214,-106.624324),
        new google.maps.LatLng(35.084214,-106.624324),
        new google.maps.LatLng(35.084391,-106.624284)
        ];
      var bldgPoly = new google.maps.Polygon({
        paths: buildingCoordinates,
        strokeColor: "#FF0000",
        strokeOpacity: 0.8,
        strokeWeight: 2,
        fillColor: "#FF0000",
        fillOpacity: 0.35
```

```
            });
            bldgPoly.setMap(map)
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps09.html

Adding an Info Window

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
          html { height: 100% }
          body { height: 100%;
            margin: 0px;
            padding: 0px;
            background-color: black;
            color: #CCCCCC;
            text-align: center}
          #map_canvas { width:90%;
            height:80%;
            margin-left: auto;
            margin-right: auto }
          .infoBox { color:black }
        </style>
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false">
        </script>
        <script type="text/javascript">
          function initialize() {
            var classroom = new google.maps.LatLng(35.084280,-106.624073)
            var office = new google.maps.LatLng(35.084506,-106.624899)
            var myOptions = {
              zoom: 18,
              center: classroom,
              mapTypeId: google.maps.MapTypeId.HYBRID
              };
            var map = new google.maps.Map(
              document.getElementById("map_canvas"),
              myOptions);
            var classroomMarker = new google.maps.Marker({
              position: classroom,
              title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
              });
```

```
            classroomMarker.setMap(map);
            var officeMarker = new google.maps.Marker({
              position: office,
              title:"Office, Bandelier West, Room 107"
              });
            officeMarker.setMap(map);
            var buildingCoordinates = [
              new google.maps.LatLng(35.084498,-106.624921),
              new google.maps.LatLng(35.084558,-106.624911),
              new google.maps.LatLng(35.084566,-106.624970),
              new google.maps.LatLng(35.084609,-106.624966),
              new google.maps.LatLng(35.084544,-106.624383),
              new google.maps.LatLng(35.084438,-106.624317),
              new google.maps.LatLng(35.084384,-106.623922),
              new google.maps.LatLng(35.084164,-106.623970),
              new google.maps.LatLng(35.084214,-106.624324),
              new google.maps.LatLng(35.084214,-106.624324),
              new google.maps.LatLng(35.084391,-106.624284)
              ];
            var bldgPoly = new google.maps.Polygon({
              paths: buildingCoordinates,
              strokeColor: "#FF0000",
              strokeOpacity: 0.8,
              strokeWeight: 2,
              fillColor: "#FF0000",
              fillOpacity: 0.35
            });
            bldgPoly.setMap(map);
            var classInfoContent = '<div class="infoBox">' +
              '<p>This is the location for the Geography 485L/585L class</p>' +
              '</div>'
            var classInfoWindow = new google.maps.InfoWindow({
              content: classInfoContent
              });
            google.maps.event.addListener(classroomMarker, 'click', function() {
              classInfoWindow.open(map,classroomMarker);
              });
          }
        </script>
    </head>

    <body onload="initialize()">
      <h1>Sample Map</h1>
      <div id="map_canvas"></div>
    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps10.html

Module 2.2 - Web-based Mapping Clients: Google Maps API - Part II - Related Topics

Overview

- Additional Google Maps API Capabilities to be Aware of
  - Styling of the base maps with custom preferences
  - Fusion Tables
- Bringing it all together in a "real" web page

Getting Started with Styled Maps - Video

Styled Maps Documentation | Styled Maps Wizard

[Figure 2 about here.]

Map Example: Simple - Styled

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
  html { height: 100% }
  body { height: 100%;
    margin: 0px;
    padding: 0px;
    background-color: black;
    color: #CCCCCC;
    text-align: center}
  #map_canvas { width:90%;
    height:80%;
    margin-left:
    auto;
    margin-right: auto }
</style>
<script type="text/javascript"
    src="http://maps.google.com/maps/api/js?v=3.2&sensor=false">
</script>
<script type="text/javascript">
  function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var myOptions = {
      zoom: 8,
      center: classroom,
      mapTypeId: google.maps.MapTypeId.ROADMAP,
      styles: [
              {
                featureType: "water",
                stylers: [
                  { visibility: "on" },
```

```
                    { hue: "#0008ff" }
                  ]
                },{
                  featureType: "road.highway",
                  stylers: [
                    { hue: "#ff1a00" }
                  ]
                },{
                  featureType: "road.arterial",
                  stylers: [
                    { hue: "#ffa200" },
                    { visibility: "simplified" }
                  ]
                },{
                  featureType: "road.local",
                  stylers: [
                    { visibility: "off" }
                  ]
                },{
                  featureType: "administrative",
                  stylers: [
                    { visibility: "simplified" }
                  ]
                },{
                  featureType: "poi",
                  stylers: [
                    { visibility: "on" },
                    { hue: "#00ffff" }
                  ]
                },{
                  featureType: "poi",
                  stylers: [
                    { visibility: "on" }
                  ]
                }
              ]
    };
    var map = new google.maps.Map(document.getElementById("map_canvas"),
        myOptions);
  }
</script>
</head>

<body onload="initialize()">
  <h1>Sample Map - Styled (POIs Emphasized)</h1>
  <div id="map_canvas"></div>
</body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/gmaps_styled.html

***Google I/O 2011: Managing and visualizing your geospatial data with Fusion Tables* - Video**

Some particularly relevant sections: Introduction (0:00 - 10:30) | Google Maps API Integration (21:40 - 34:42) | Summary and Links (52:00 52:40)

Fusion Tables Documentation/Help

[Figure 3 about here.]

Bringing It All Together

```
<!DOCTYPE html>

<html>
    <head>
        <meta charset="utf-8" />
        <title>Karl's Event Diary</title>
        <link rel="stylesheet" href="./styles/base.css" media="screen">
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
        <script type="text/javascript" src="./js/base.js"></script>
        <script type="text/javascript">
            // Define a set of global coordinates for use throughout the web site
            // Place coordinates derived from GNIS database: http://geonames.usgs.gov/pls/gnispublic
            var eventPlaces = [
                {
                    name: "Albuquerque",
                    point: new google.maps.LatLng(35.0889356,-106.5747462),
                    label: "Albuquerque: Duke City Half Marathon"
                },
                {
                    name: "Durango",
                    point: new google.maps.LatLng(37.2752800,-107.8800667),
                    label: "Durango: Animas Valley/Steamworks Half Marathon"
                },
                {
                    name: "San Diego",
                    point: new google.maps.LatLng(32.7153292,-117.1572551),
                    label: "San Diego: San Diego Rock 'n' Roll Marathon"
                },
                {
                    name: "San Francisco",
                    point: new google.maps.LatLng(37.7749295,-122.4194155),
                    label: "San Francisco: Nike Women's Marathon"
                },
                {
                    name: "Orlando",
                    point: new google.maps.LatLng(28.5383355,-81.3792365),
                    label: "Orlando: Walt Disney World half- and full-marathon"
                },
                {
                    name: "Anaheim",
                    point: new google.maps.LatLng(33.8352932,-117.9145036),
                    label: "Anaheim: Disneyland Half Marathon"
```

```
                }
            ]
        </script>
    </head>

    <body onload="initialize()">
        <h1>
            My diary of endurance events that I've participated in since joining Team in Training
        </h1>

        <p>In 2008 Cynthia and I joined the Leukemia and Lymphoma Society's (<a href="http://www.lls.org
        participants to train for the Animas Valley/Steamworks Half Marathon and raise money for blood
        research and patient services. In spite of our not having any direct connection to blood cancer
        we found the goals of LLS admirable,  the combined training and fund-raising program of TNT a g
        new friends over the many seasons that we've been involved with TNT.</p>

        <p>Since 2008 we have continued to volunteer with TNT, as participants, mentors, and since 2010
        (check out my <a href="http://youtu.be/GMSKG8L6K78#t=2m13s">half-second</a> of fame in the info
        for TNT with an emphasis on training walkers for full- or half-marathons. This page provides a
        events that I've participated in in some capacity since we became involved with TNT. </p>

        <div id="event-map" name="event-map"></div>


        <h2>
            <span class="date">9/1/2013</span>
            Disneyland Half Marathon
            <span class="time">2:56:57</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[5].point, 12)">approx. map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">1/13/2013</span>
            Disney World  Marathon (Goofy - Day 2)
            <span class="time">6:46:57</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">approx. map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">1/12/2013</span>
            Disney World Half Marathon (Goofy - Day 1)
            <span class="time">3:22:48</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">approx. map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">9/29/2012</span>
            Hot Chocolate 15k
            <span class="time">1:56:46</span>
            (no map available)
        </h2>
```

```
97          <p class="eventDescription">blah, blah, blah ...</p>
98
99          <h2>
100             <span class="date">6/9/2012</span>
101             Animas Valley/Steamworks Half Marathon
102             <span class="time">no time: coached</span>
103             (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
104         </h2>
105         <p class="eventDescription">blah, blah, blah ...</p>
106
107         <h2>
108             <span class="date">1/9/2012</span>
109             Disney World  Marathon (Goofy - Day 2)
110             <span class="time">6:56:28</span>
111             (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
112         </h2>
113         <p class="eventDescription">blah, blah, blah ...</p>
114
115         <h2>
116             <span class="date">1/8/2011</span>
117             Disney World Half Marathon (Goofy - Day 1)
118             <span class="time">3:29:00</span>
119             (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
120         </h2>
121         <p class="eventDescription">blah, blah, blah ...</p>
122
123         <h2>
124             <span class="date">6/19/2010</span>
125             Animas Valley/Steamworks Half Marathon
126             <span class="time">no time: coached</span>
127             (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
128         </h2>
129         <p class="eventDescription">blah, blah, blah ...</p>
130
131         <h2>
132             <span class="date">6/6/2010</span>
133             San Diego Rock 'n' Roll Marathon
134             <span class="time">no time: coached</span>
135             (<a href="#event-map" onclick="recenter(map, eventPlaces[2].point, 11)">map</a>)
136         </h2>
137         <p class="eventDescription">blah, blah, blah ...</p>
138
139         <h2>
140             <span class="date">10/18/09</span>
141             Nike Women's Marathon
142             <span class="time">7:13:05</span>
143             (<a href="#event-map" onclick="recenter(map, eventPlaces[3].point, 12)">map</a>)
144         </h2>
145         <p class="eventDescription">blah, blah, blah ...</p>
146
147         <h2>
148             <span class="date">9/6/2009</span>
149             Disneyland Half Marathon
150             <span class="time">3:43:05</span>
```

```
            (<a href="#event-map" onclick="recenter(map, eventPlaces[5].point, 12)">map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">1/11/2009</span>
            Disney World Marathon
            <span class="time">6:57:42</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">10/19/2008</span>
            Duke City Half Marathon
            <span class="time">3:09:42</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[0].point, 11)">map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">6/21/2008</span>
            Animas Valley/Steamworks Half Marathon
            <span class="time">3:14:52</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

    </body>

</html>
```

http://karlbenedict.com/GEOG485-585/lectures/examples/tnt/index.html

Module 2.3 - Web-based Mapping Clients: OpenLayers Javascript Framework - Part I

Outline

- Capabilities
- OpenLayers = Javascript (by example)

OpenLayers Capabilities

- Support for Multiple basemaps: *Google*, *Yahoo*, *Bing*, *OpenStreetMap*
- Model for interaction with multiple map server platforms: *ArcGIS* (REST & cache), *ArcIMS*, *KaMap*, *MapServer*
- Support for key OGC standards: *WMS*, *WMTS*, *WFS*, *GML*, *KML*, *SLD*
- Multiple control types: *Navigation*, *Pan*, *Zoom*, *Overview*, *Scale*, *Feature Creation & Editing*, *Graticle*, *Layer Switcher*

- Custom styled features with associated attributes: *Curve*, *LinearRing*, *LineString*, *MultiLineString*, *MultiPoint*, *MultiPolygon*, *Point*, *Polygon*, *Rectangle*
- Support for many formats for data read and write: *ArcXML*, *ATOM*, *GeoRSS*, *GPX*, *KML*, *WKT*, any many others
- Open Source, enabling modification and integration into other systems (e.g. GeoExt)

Distinguishing Characteristics Between OpenLayers and Google Maps

- Greater emphasis on client-side processing - Client access and rendering of data files that Google's servers otherwise take care of (pros & cons to this approach)
- Integrated support for OGC services and their products
- Support for different projections (adds complexity)
- API more rich in options ==> more complexity

Resources

OpenLayers Home Page

Application Programming Interface (API) Reference

Examples

Demonstrations and Examples

- Basic Mapper (with OpenStreetMaps [OSM] base map)

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script type="text/javascript" src="http://openlayers.org/api/OpenLayers.js"></script>
    <script type="text/javascript">
        // define global variables
        var lon = -106.5;
        var lat = 36;
        var zoom = 3;
        var map;
        var layer;

        // =============== Initialization function ===================
        function init(){
            map = new OpenLayers.Map( 'map' );

            // ========== OSM Map ====================
            layer = new OpenLayers.Layer.OSM( "Open Street Map");
            map.addLayer(layer);

            map.setCenter(
                new OpenLayers.LonLat(lon, lat).transform(
                    new OpenLayers.Projection("EPSG:4326"),
                    map.getProjectionObject()
                ), zoom
            );
```

```
        }
        // ================ End of Initialization Function ============

    </script>
    <style type="text/css">
        #map {width:90%; height:500px}
    </style>
  </head>
  <body onload="init()">
    <h1>Basic OpenLayers Map</h1>
    <p>Shows the basic use of OpenLayers with the <a href="http://www.openstreetmap.org/">OpenStreetmap
    <!-- Map DIV -->
    <div id="map"></div>
  </body>
</html>
```

Demonstration and Examples - Online Resources

- Mapper with a variety of base maps (Google, Bing, Yahoo, OSM)
- Basic Mapper with Controls: No Controls, Layer Switcher, Control Array, Overlay Map, Scale Information
- Positioning Controls with the `moveTo` function: two controls moved

Next Week - Custom Features and WMS Layers

Module 2.3 - Web-based Mapping Clients: OpenLayers Javascript Framework - Part II

Outline

- More detailed Map Object Options
- More detailed Layer Object Options
- Additional Map Layer Types - With Examples

Map Object Options

Map Object Options API Reference

Two methods for constructing a new `OpenLayers.Map` object

```
    // create a map with default options in an element with the id "map1"
    var map = new OpenLayers.Map("map1");

    // create a map with non-default options in an element with id "map2"
    var options = {
        maxExtent: new OpenLayers.Bounds(-200000, -200000, 200000, 200000),
        maxResolution: 156543,
        units: 'm',
        projection: "EPSG:41001"
```

```
    };
    var map = new OpenLayers.Map("map2", options);

    // map with non-default options - same as above but with a single argument
    var map = new OpenLayers.Map({
        div: "map_id",
        maxExtent: new OpenLayers.Bounds(-200000, -200000, 200000, 200000),
        maxResolution: 156543,
        units: 'm',
        projection: "EPSG:41001"
    });
```

Map Object Options - continued

Excerpts from the API documentation

allOverlays {Boolean} Allow the map to function with "overlays" only. Defaults to false. If true, the lowest layer in the draw order will act as the base layer. In addition, if set to true, all layers will have isBaseLayer set to false when they are added to the map.

div {DOMElement|String} The element that contains the map (or an id for that element). If the OpenLayers.Map constructor is called with two arguments, this should be provided as the first argument. Alternatively, the map constructor can be called with the options object as the only argument. In this case (one argument), a div property may or may not be provided. If the div property is not provided, the map can be rendered to a container later using the render method.

Map Object Options - continued

layers {Array(OpenLayers.Layer)} Ordered list of layers in the map

tileSize {OpenLayers.Size} Set in the map options to override the default tile size for this map.

projection {String} Set in the map options to override the default projection string this map - also set **maxExtent**, **maxResolution**, and **units** if appropriate. Default is "EPSG:4326".

units {String} The map units. Defaults to 'degrees'. Possible values are 'degrees' (or 'dd'), 'm', 'ft', 'km', 'mi', 'inches'.

Map Object Options - continued

resolutions {Array(Float)} A list of map resolutions (map units per pixel) in descending order. If this is not set in the layer constructor, it will be set based on other resolution related properties (maxExtent, maxResolution, maxScale, etc.).

maxResolution {Float} Default max is 360 deg / 256 px, which corresponds to zoom level 0 on gmaps. Specify a different value in the map options if you are not using a geographic projection and displaying the whole world.

minResolution {Float}

maxScale {Float}

minScale {Float}

Map Object Options - continued

maxExtent {OpenLayers.Bounds} The maximum extent for the map. Defaults to the whole world in decimal degrees (-180, -90, 180, 90). Specify a different extent in the map options if you are not using a geographic projection and displaying the whole world.

minExtent {OpenLayers.Bounds}

restrictedExtent {OpenLayers.Bounds} Limit map navigation to this extent where possible. If a non-null restrictedExtent is set, panning will be restricted to the given bounds. In addition, zooming to a resolution that displays more than the restricted extent will center the map on the restricted extent. If you wish to limit the zoom level or resolution, use maxResolution.

numZoomLevels {Integer} Number of zoom levels for the map. Defaults to 16. Set a different value in the map options if needed.

Layer Object Options

Layer Object Options API Reference

Common Pattern of Layer Object Creation (varies some depending upon the specific layer type)

```
1     new OpenLayers.Layer.*** (
2         'layer name',
3         'layer URL',
4         {server-related options},
5         {OpenLayers Layer Object options}
6     )
```

Layer Object Options - continued

id {String}

name {String}

isBaseLayer {Boolean} Whether or not the layer is a base layer. This should be set individually by all subclasses. Default is false

displayInLayerSwitcher {Boolean} Display the layer's name in the layer switcher. Default is true.

visibility {Boolean} The layer should be displayed in the map. Default is true.

Layer Object Options - continued

attribution {String} Attribution string, displayed when an OpenLayers.Control.Attribution has been added to the map.

projection {OpenLayers.Projection} or {String} Set in the layer options to override the default projection string this layer - also set maxExtent, maxResolution, and units if appropriate. Can be either a string or an OpenLayers.Projection object when created – will be converted to an object when setMap is called if a string is passed.

units {String} The layer map units. Defaults to 'degrees'. Possible values are 'degrees" (or 'dd'), 'm', 'ft', 'km', 'mi', 'inches'.

`scales` {Array} An array of map scales in descending order. The values in the array correspond to the map scale denominator. Note that these values only make sense if the display (monitor) resolution of the client is correctly guessed by whomever is configuring the application. In addition, the units property must also be set. Use resolutions instead wherever possible.

Layer Object Options - continued

`resolutions` {Array} A list of map resolutions (map units per pixel) in descending order. If this is not set in the layer constructor, it will be set based on other resolution related properties (maxExtent, maxResolution, maxScale, etc.).

`maxExtent` {OpenLayers.Bounds} The center of these bounds will not stray outside of the viewport extent during panning. In addition, if displayOutsideMaxExtent is set to false, data will not be requested that falls completely outside of these bounds.

`minExtent` {OpenLayers.Bounds}

`maxResolution` {Float} Default max is 360 deg / 256 px, which corresponds to zoom level 0 on gmaps. Specify a different value in the layer options if you are not using a geographic projection and displaying the whole world.

`minResolution` {Float}

Layer Object Options - continued

`numZoomLevels` {Integer}

`minScale` {Float}

`maxScale` {Float}

`displayOutsideMaxExtent` {Boolean} Request map tiles that are completely outside of the max extent for this layer. Defaults to false.

`transitionEffect` {String} The transition effect to use when the map is panned or zoomed.

There are currently two supported values `null` No transition effect (the default).

`resize` Existing tiles are resized on zoom to provide a visual effect of the zoom having taken place immediately. As the new tiles become available, they are drawn over top of the resized tiles.

Additional Map and Layer Object Functions & Events

Both Map and Layer Objects have a number of associated functions as well

- Retrieving object properties programmatically with `Get` functions.
- Modifying existing object properties with `Set` functions
- Map destruction, and reconfiguration
- Linkage of object events with Javascript functions

WMS Layer Configuration

Some key issues to be aware of when using the WMS Layer Class:

- The *projection* of the map object must be supported by the included WMS service (review the WMS GetCapabilities response to see what projections are supported by the service)
- The *layers* parameter/property must be provided as part of the server-related property list (the layer names also come from the GetCapabilities response)
- Other WMS parameters may be provided as well to "adjust" the request automatically generated by OpenLayers

Sample WMS Layer Object Creation

```
1     countiesLayer = new OpenLayers.Layer.WMS(
2         "US Counties",
3         "http://webservices.nationalatlas.gov/wms?",
4         {layers: "counties", version: '1.3.0', transparent: 'TRUE'},
5         {isBaseLayer: false, visibility: false, opacity: .8}
6     );
7     map.addLayer(countiesLayer);
```

Example

Vector Layer Configuration

Vector layers support

- External Data in a Variety of supported formats for both *reading* and *writing* (just a sample): ArcXML.Features, GeoJSON, GeoRSS, GPX, JSON, KML, WFS, WKT
- Directly encoded [geometries][OpenLayers.Geometry API Link]: Collection, Curve, LinearRing, LineString, MultiLineString, MultiPoint, MultiPolygon, Point, Polygon, Rectangle
- User created features, including support for interactive editing of features
- Styling of Vector features

Vector Layer Configuration - Continued

Vector Layer Objects are Typically Defined using three OpenLayers classes

`Protocol` Connection protocol for requesting the data that would be provided from an external source

`Format` The OpenLayers supported format of the vector data object

`Strategy` A specification of how OpenLayers should request the data from the server, and also handle the data within the client (browser).

Vector Layer Configuration - Continued

Sample Point Feature Object creation

```
    var Coord_classroom = new OpenLayers.Geometry.Point(-106.624073,35.084280);
    var Point_classroom = new OpenLayers.Feature.Vector(Coord_classroom);
    Layers["localFeatures"].addFeatures([Point_classroom])
```

Sample KML Layer Object creation

```
    Layers.counties = new OpenLayers.Layer.Vector("KML - Counties", {
        projection: map.displayProjection,
        strategies: [new OpenLayers.Strategy.Fixed()],
        protocol: new OpenLayers.Protocol.HTTP({
            url: "NMCounties.kml",
            format: new OpenLayers.Format.KML({
                extractAttributes: true
            })
        })
    });
    map.addLayer(Layers.counties)
```

Example # Module 3 - GIS and Services Oriented Architectures #

Outline

- Geographic Information Systems
  - Data Types
  - Coordinate Systems
- Services Oriented Architectures
  - Historic Context
  - Current Model - Network Computing
  - Components
  - Interoperability Standards

Geographic Information Systems

Data Types - Vector

- Vector data represent phenomena that are associated with specific bounded locations, typically represented by:
  - Points
  - Lines
  - Polygons
- Vector data include:
  - The geometries that describe the area being referenced, and
  - Attributes associated with that area

For example, a census vector data product might include the geometries that define census tracts and attributes associated with each geometry: population, income, etc.

Data Types - Raster

- Raster data are frequently used to represent values for phenomena that vary continuously across space (e.g. elevation, concentration of air pollutants, depth to ground water, etc.)
- These values are encoded over a regular grid of observation locations with a specified grid spacing - often referred to as the spatial resolution of the dataset (i.e. 10m resolution for a standard USGS Digital Elevation Model product)
- Often parts of data collections that are repeated (i.e. remote sensing data products)

Accessing and Processing Raster and Vector Data

- Two geospatial libraries and their related utility programs provide information about and tools for modifying vector and raster data sets

OGR vector data access and information

GDAL raster data access and information

These libraries are the data access and processing foundation for a growing number of open source and commercial mapping systems

Information and documentation: GDAL Home Page | OGR Home Page

Coordinate Systems/Projections

- To convert locations from a 3-dimensional oblate spherical coordinate system (such as is commonly used to represent the surface of the earth) to a 2-dimensional representation in a map, a coordinate transformation must be performed.
- There are a limitless number of potential coordinate transformations possible, and a large number have been named and defined that meet specific cartographic or other requirements

EPSG Codes

- A catalog of numeric codes and associated coordinate transformation parameters is maintained by the International Association of Oil & Gas Producers (OGP) - the successor scientific organization to the European Petroleum Survey Group (EPSG)
- These numeric codes are used by many desktop and online mapping systems to document and represent the coordinate systems of available data and services
- Links to an online version of the registry and downloadable databases of the registry are available from: http://www.epsg.org/Geodetic.html.

Projection Parameters

The parameters that define a map projection may be looked up in a number of online locations:

EPSG registry (helpful if you already know the EPSG code of the projection you are looking for)
http://www.epsg-registry.org/

GeoTIFF Projection List (helpful if you know the name of one of the broadly used projections - uneven per
http://www.remotesensing.org/geotiff/proj_list/

SpatialReference.org (decent search tool, includes non-EPSG as well as EPSG projection information, multi
http://spatialreference.org/

Coordinate Transformation Calculations

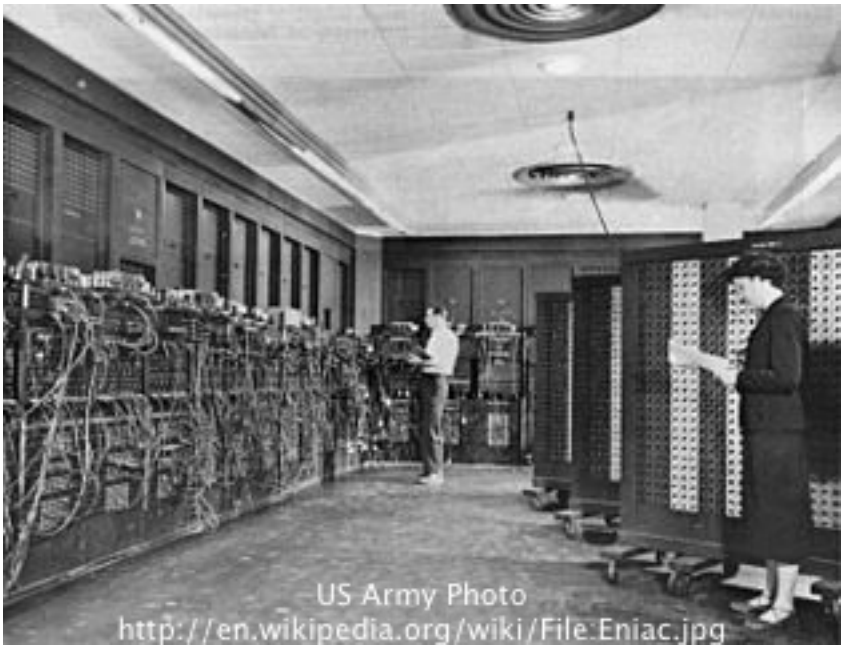
When the projection parameters are in hand, the Proj4 library (http://trac.osgeo.org/proj/) and related utilities (`cs2cs` and `proj`) can be used to perform coordinate transformation calculations. `cs2cs` is my recommended utility for coordinate conversion because of the explicit definition of both source and destination coordinate reference system.

Coordinate Transformation Calculations - Examples

```
KB:~ kbene$ cs2cs +proj=longlat +ellps=WGS84 +datum=WGS84 +to +proj=utm +zone=13 +ellps=GRS80 +datum=NA
106.75W 35N
340301.04   3874442.20 0.00
^C

KB:~ kbene$ cs2cs +init="EPSG:4326" +to +init="EPSG:26913"
106.75W 35N
340301.04   3874442.20 0.00
^C

KB:~ kbene$ cs2cs +proj=utm +zone=13 +ellps=GRS80 +datum=NAD83 +units=m +to +proj=longlat +ellps=WGS84 +
340301.04   3874442.20
106d45'W    35dN 0.000
^C

KB:~ kbene$ cs2cs +init="EPSG:26913" +to +init="EPSG:4326"
340301.04   3874442.20
106d45'W    35dN 0.000
^C
```

Services Oriented Architectures

Where have we come from - ENIAC (1946)



- First general purpose electronic computer
- Programmable, but could not store programs

Where have we come from - Early Client-Server Computing (1960s)

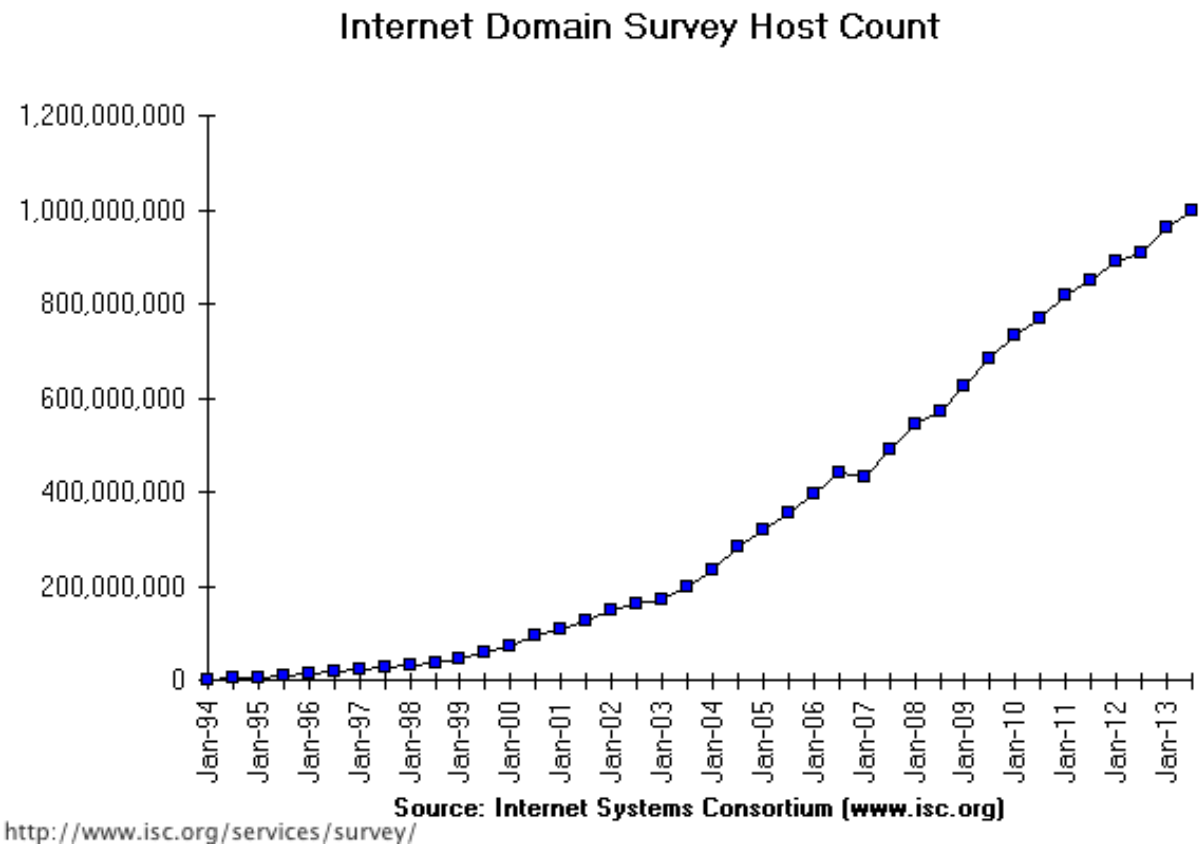
Photo courtesy of Dominic's Pics
http://www.flickr.com/photos/dominicspics/

- Mainframe computers to which client terminals connected over a local network
- Computing performed by server, client purely a display device

Where have we come from - Personal Computers (1970s)

- Desktop computers capable of running a variety of operating systems and applications
- In some environments can be interconnected to a central local server

Now - Network computing



- Predecessor to the Internet - ARPANET (1969). Interconnection between UCLA and SRI (Menlo Park)
- Adoption of TCP/IP as next generation protocol for ARPANET (1983)
- NSF commissions construction of NSFNET, also based upon TCP/IP (1983)
- NSFNET opened to commercial connections (1988). Led to interconnection of multiple, previously separate networks into an "Internet"
- Growth of internet users has expanded rapidly over the past decade

In a Phrase ...

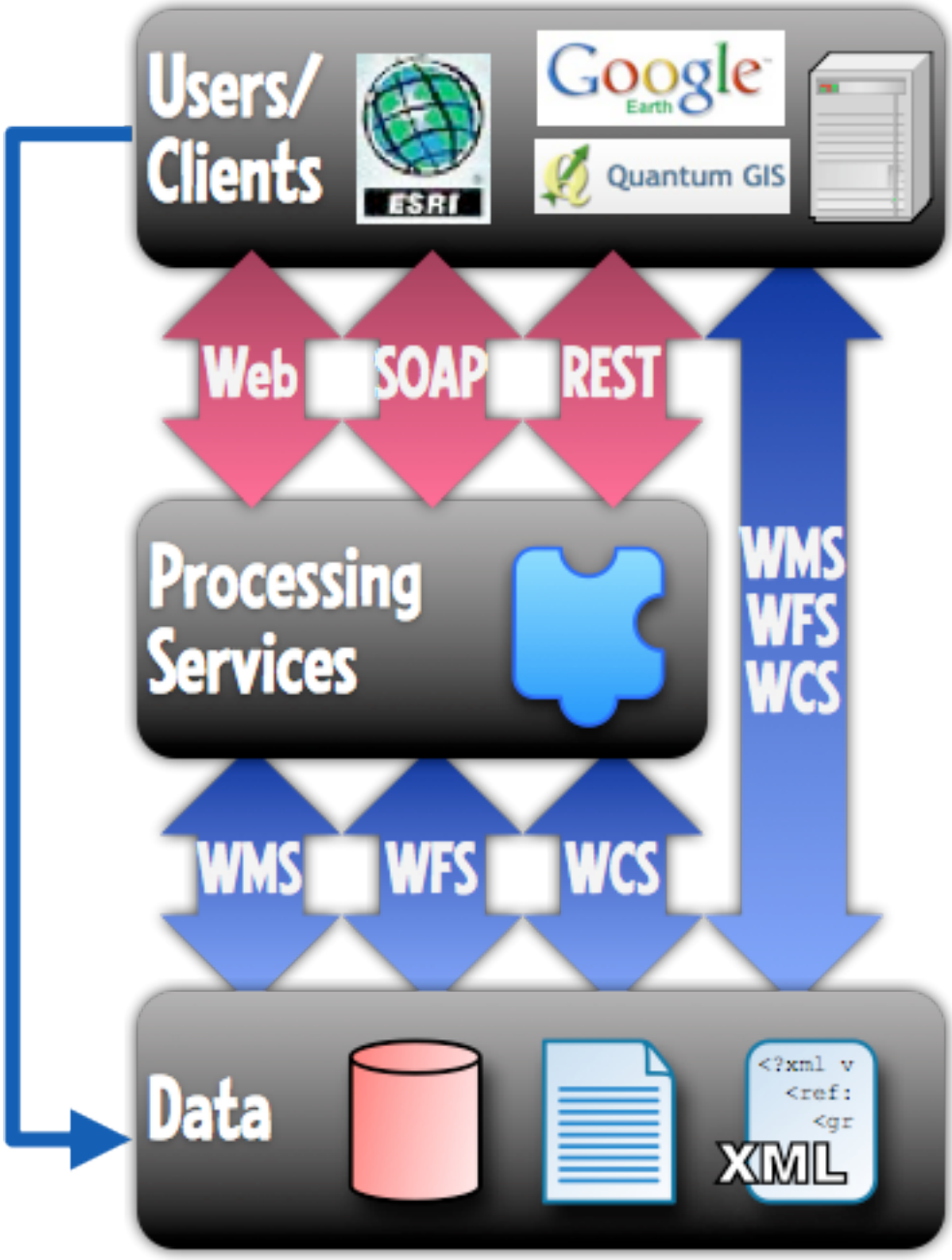
The current networking computing model consists of *Components Interacting* with Each Other

So - We Need to Answer the Following Questions

What are components?

What does it mean to interact?

The Big Picture - Services Oriented Architectures



- Services Oriented Architecture (SOA) for Geospatial Data and Processing
  - Data, Processing & Client Tiers
- Open Geospatial Consortium Interoperability Standards
  - WMS, WFS, WCS
- Geospatial Metadata Standards
  - ISO 19115, FGDC

- Internet Standards
  - Web: HTML, CSS, JavaScript, XML
  - SOAP - Simple Object Access Protocol
  - REST - Representation State Transformation

The Pieces - Components

Key Components - Data

Database systems

- Optimized for storing massive quantities of tabular data
- May be spatially enabled to support the storage of geometries (points, lines, polygons) in addition to related attribute data
- Standard language (Structured Query Language [SQL]) for interacting with many databases
- Broad support for accessing the contents of databases from many other applications and programming languages, for example:
  - Spreadsheets
  - Statistical Software
  - Geographic Information Systems (GIS)

Key Components - Data

File-based data

- Often stored on the file system
- Sometimes difficult represent data within a database structure (i.e. binary data)
- May be in a wide variety of formats
  - XML
  - ASCII Text (e.g. CSV, tab-delimited)
  - Binary files
  - Excel Spreadsheets
  - Word Processing Documents
  - Geospatial data (e.g. imagery)
- Remotely Accessible Data
  - Some data may be provided through reference to an external network resource (i.e. a web address, or other identifier) or service

Key Components - Processing Services

- Perform modification of source data to generate a new data product
- May be "chained" together to create a processing "workflow". Output from one processing service may be used as the input to another
- May be simple OGC services; or complex data processing, analysis, or visualization services. Examples include
  - Extraction of a subset of a large data set based upon provided search criteria

  - Generation of a map from a collection of data
  - Fusion of two data products into a single derived product (e.g. vegetation indices calculated from multiple remote sensing images)
  - Calculation of statistical information for an input product, and delivery of the statistical summary

Key Components - Clients

- Any system that accesses the services provided by the system may be considered a "client"
- That system may be manually operated by a human user, or triggered automatically by software
- Human operated clients include
  - Web-based applications
  - Desktop applications such as Geographic Information Systems and Statistical Analysis tools
- Machine clients include
  - Data processing services that translate requests to them into requests for other system services
  - Regularly scheduled requests that are automatically triggered by external computer systems.

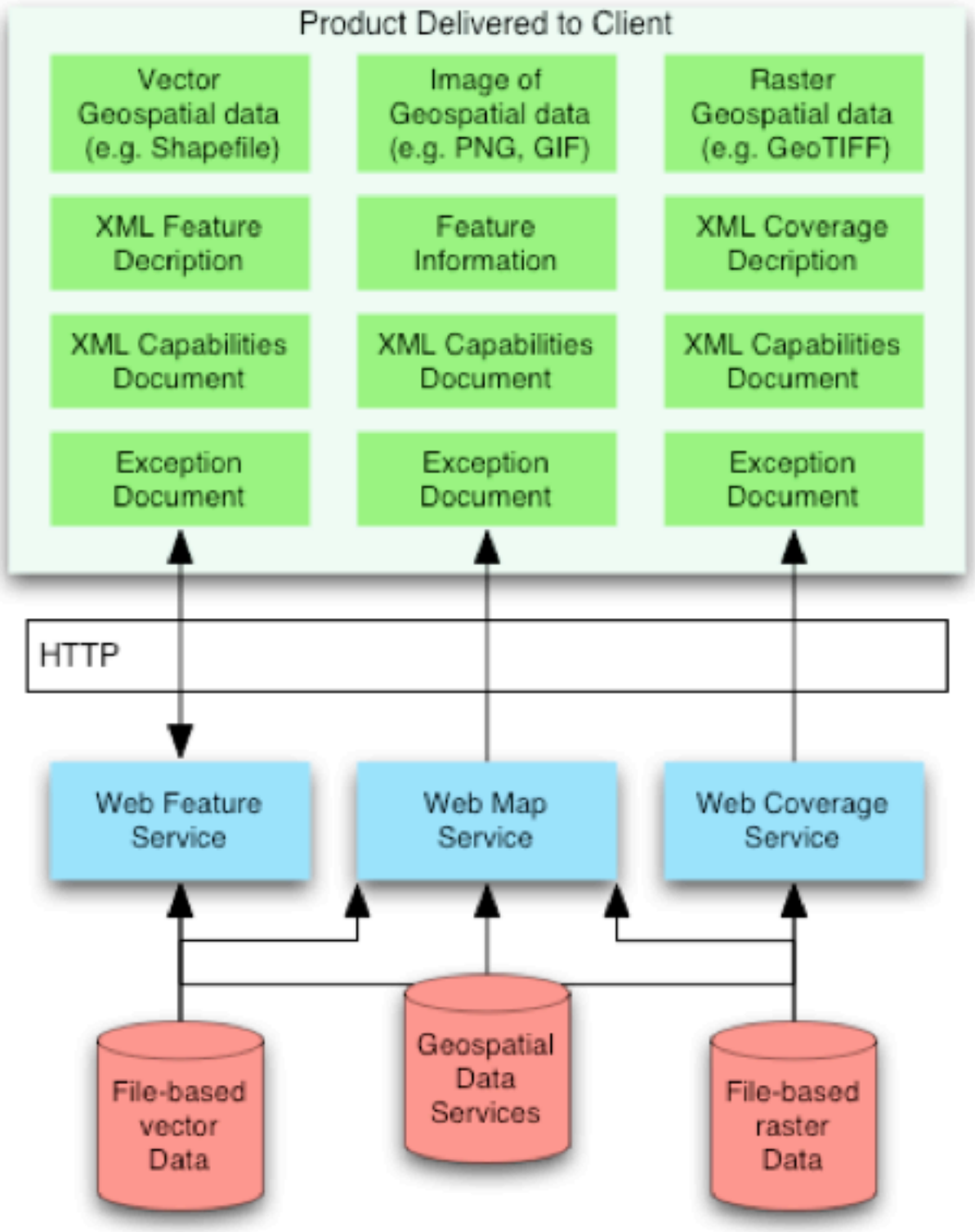
The Glue - Interoperability Standards / Service Interfaces

Open Geospatial Consortium Interoperability Standards

Open Geospatial Consortium (OGC) Standards

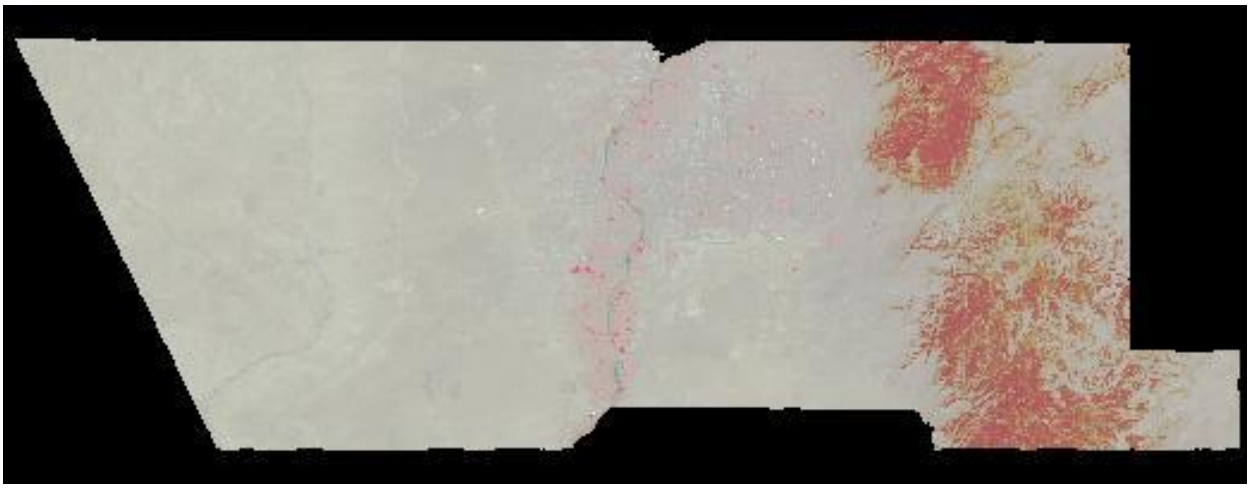
- Two Classes of Standards Considered Here
  - Geospatial Product Access Standards
  - Geospatial Data and Representation Standards
- Product Access Standards
  - Web Map Services (WMS)
  - Web Feature Services (WFS)
  - Web Coverage Services (WCS)
- Data and Representation Standards
  - Geography Markup Language (GML)
  - KML (formerly known as Keyhole Markup Language)

Comparison of OGC Service Models



OGC Web Map Services (WMS)

```
http://gstore.unm.edu/apps/rgis/datasets/b030ab7b-86e3-4c30-91c0-f427303d5c77/services/ogc/wms?
    VERSION=1.1.1&&
    SERVICE=WMS&
    REQUEST=GetMap&
    SRS=EPSG:4326&
    FORMAT=image/jpeg&
    STYLES=&
    LAYERS=bernalillo_tm2011&
    TRANSPARENT=TRUE&
    WIDTH=521&
    HEIGHT=200&
    bbox=-107.207,34.8404,-106.143,35.2487
```



- HTTP GET (required), HTTP POST (optional)
- Requests:
  - `GetCapabilities`
  - `GetMap`
  - `GetFeatureInfo`
- Returns
  - Mapped data
  - XML Capabilities Document, Feature Attributes
- Includes support for time-based requests

OGC Web Feature Services (WFS)

- Either HTTP GET or POST required
- Requests
  - `GetCapabilities`
  - `DescribeFeatureType`
  - `GetFeature/GetFeatureWithLock`
  - `GetGmlObject`

  - `LockFeature`
  - `Transaction`
- Returns
  - XML (GML)
  - Capabilities
  - Feature Data

OGC Web Coverage Services (WCS)

- Either HTTP GET or POST required
- Requests
  - `GetCapabilities`
  - `DescribeCoverage`
  - `GetCoverage`
- Returns
  - Geospatial data for coverage
  - XML Capabilities
- Includes support for time-based requests

OGC Geography Markup Language (GML)

- GML is an XML grammar for representing geospatial features and their associated attributes
- In its generic form it can encode points, lines, and polygons and their associated attributes
- As an XML schema GML was designed to be extensible by communities of practice for consistent encoding of geographic data more richly than allowed by the generic default model
- GML documents representing large complex geometries can be quite large - therefore slow to transfer over the Internet

OGC KML

- An XML specification that supports the encoding of representation and embedding of geospatial data for use in geospatial viewers
- Began as the underlying representation language of Google Earth (originally developed by Keyhole for their virtual Earth viewer)
- Adopted as an OGC standard in 2008
- Supports data linkage through
  - Embedding
  - Reference through external URLs - with WMS specifically supported through *parameterization*
- Includes support for the representation of time in relation to data objects

Implementation of the OGC Standards

- WMS
  - 1.3.0 - 284 implementations
  - 1.1.1 - 474

  - 1.1 - 238
  - 1.0 - 274
- WFS
  - 2.0 - 36
  - 2.0 transactional - 3
  - 1.1.0 - 228
  - 1.1.0 transactional - 52
  - 1.0.0 - 304
  - 1.0.0 transactional - 113
- WCS
  - 2.0 - 7
  - 1.1.2 - 19
  - 1.1.1 - 37
  - 1.1.0 - 30
  - 1.0.0 Corregendum - 190

Implementation information based upon OGC Implementation Statistics - Accessed 2/2014

Implementation of the OGC Standards

- KML
  - 2.2.0 - 74
  - 2.2 Reference (Best Practice) - 11
  - 2.1 Reference (Best Practice) - 64
- GML
  - 3.3 - 5
  - 3.2.1 - 110
  - 3.1.1 - 161
  - 3.0 - 127
  - 2.1.2 - 142
  - 2.1.1 - 100
  - 2.0 - 82
  - 1.0 - 20

Implementation information based upon OGC Implementation Statistics - Accessed 2/2014

OGC Summary

The OGC web service specifications support key geospatial data access requirements

WMS visualization of geospatial data through simple web requests

WFS delivery of geospatial data (typically points, lines, and polygons) in a format that is usable in GIS and other applications

WCS delivery of geospatial data (typically, but not limited to, raster data) usable in other applications

OGC Summary

The OGC data and representation standards support data exchange and higher level representation

GML XML schema for the representation of features and associated attributes. It may be extended for use by specific communities of users (i.e. ecological data models)

KML XML schema that supports the combination of embedded data and external data into a complete representation model that may be used by client applications to present the data through a user interface (e.g. Google Earth, WorldWind)

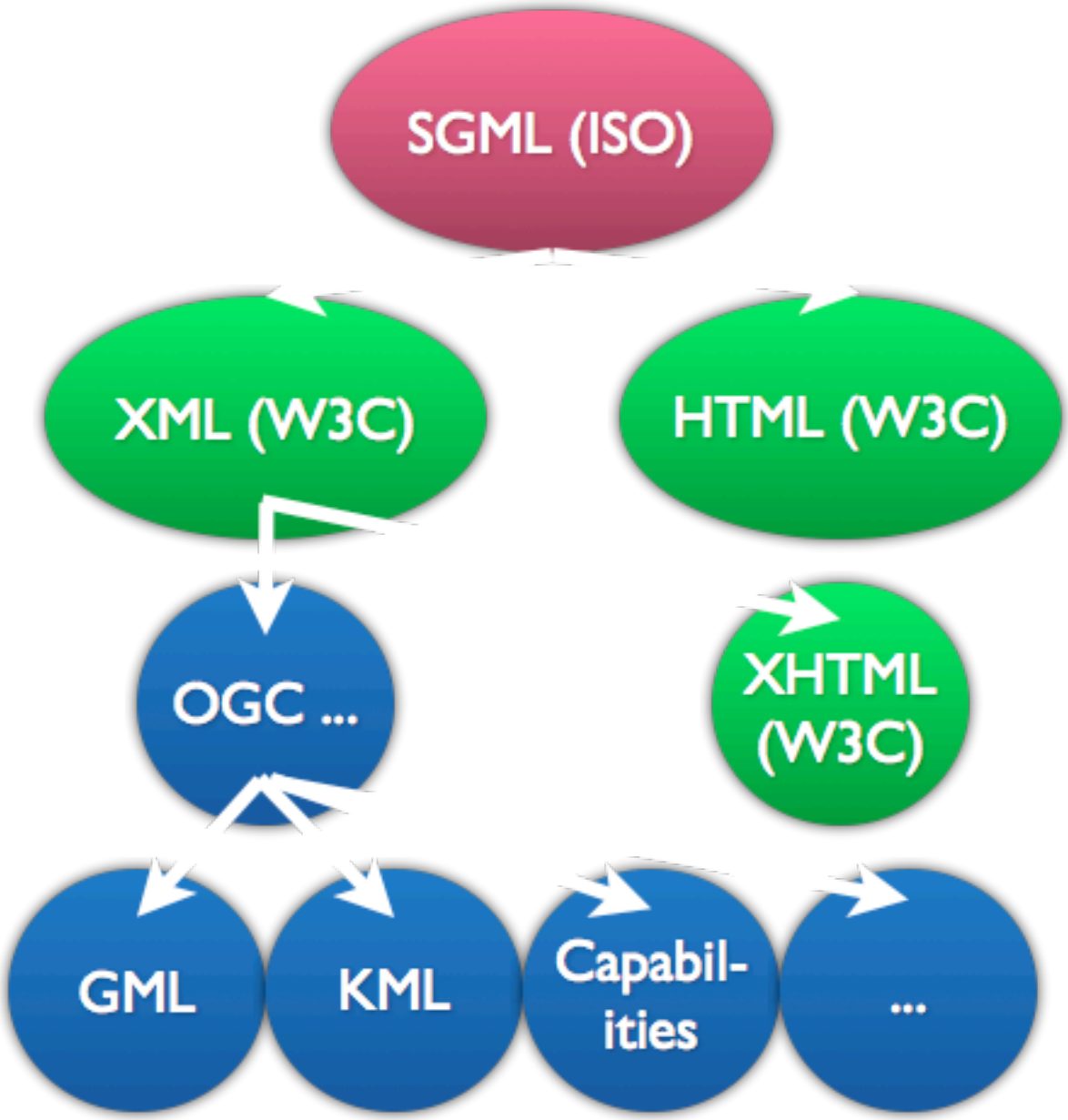
Module 4.1 - Interoperability Standards - WMS, KML, and XML

Outline

- Extensible Markup Language - XML
  - Definition of a markup language
  - Requirements
  - Extensible ???
- KML - AKA Keyhole Markup Language
  - An XML Document Format
  - Combined representation of spatial data and time
- OGC Web Map Services (WMS)
  - Requests and Results
  - GetCapabilities, GetMap, GetFeatureInfo
- Integration of WMS into KML

Extensible Markup Language - XML

XML Background



- Defined as a markup language profile of Standard Generalized Markup Language (SGML - ISO 8879:1986)
- XML 1.0 released as a W3C Recommendation in 1998
  - currently in 5th edition, released in 2008
  - version 1.1 released in 2004, but not broadly used
  - XML 1.0 (5th ed.) Recommendation

XML Design Goals

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

From XML 1.0 (5th ed.) Recommendation

XML Structure - Well Formed / Valid

- Well Formed XML - a document that conforms to the structural definition of XML. Either well-formed, or not XML
- Valid XML - a document that is both well-formed and conforms to a specific content structure defined by
  - A Document Type Definition (DTD) - the original XML specification for the definition of the content of a specific XML document
  - A Schema document - defined in a variety of languages (e.g. W3C Schema, RELAX NG, Schematron, ISO DSDL, etc.)

XML Wikipedia Article

Simple XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
</note>
```

XML Source (modified from original): w3schools

Simple XML Document - Prolog

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
```

Includes XML Declaration and Comment

Simple XML Document - Elements

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
</note>
```

Define blocks of content

Simple XML Document - Root Element

```
<note>
    ...
    ...
    ...
    ...
</note>
```

- Required
- There is only one
- It must be a pair of opening and closing tags

Simple XML Document - Content Elements

```
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
```

- Contain all other document content
- May be paired opening and closing tags, *or*
- May be self-closing with a terminal "/" in the element, e.g. `<br />`

Simple XML Document - Attributes

```
    <body type="instruction" >Don't forget me this weekend!</body>
```

Define additional information about elements as *name=value* pairs.

Simple XML Document - Element Content

```
    <body type="instruction" >Don't forget me this weekend!</body>
```

The material contained between the opening and closing tags of an *Element*.

Simple XML Document - Valid?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
</note>
```

Why is this XML *well-formed* but not *valid*?

There is no DTD or Schema defined for the document against which it can be validated

Common XML Constructs That Will be Encountered

Document Type Declaration (DTD) references (PROLOG) definition, either by reference or by direct inclusion, the allowed structure of an XML document, for example:

```
<!DOCTYPE greeting SYSTEM "hello.dtd">
```

CDATA Sections blocks of XML that contain characters that would otherwise be recognized as XML markup, for example:

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

Common XML Constructs That Will be Encountered - cont.

XML Namespace Declarations additional information included in elements to distinguish between duplicate element names, for example (declared in lines 1-3, used in lines 5-17):

```
<root
    xmlns:h="http://www.w3.org/TR/html4/"
    xmlns:f="http://www.w3schools.com/furniture">

<h:table>
    <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
    </h:tr>
</h:table>
<f:table>
    <f:legs>4</f:legs>
    <f:cost>300</f:cost>
    <f:width>3</f:width>
    <f:length>5</f:length>
    <f:height>4</f:height>
</f:table>
</root>
```

KML

KML Background

- An XML grammar originally developed as Keyhole Markup Language by Keyhole, Inc. for use in their Keyhole Earth Viewer.
- Google acquired Keyhole, Inc. in 2004
- KML version 2.2 became an OGC standard in 2008
- Two delivered KML file formats

KML an XML document, with a ".kml" extension that is directly readable and editable

KMZ a compressed (zipped) file with a ".kmz" extension, that contains at least a KML document, but may contain other files as well

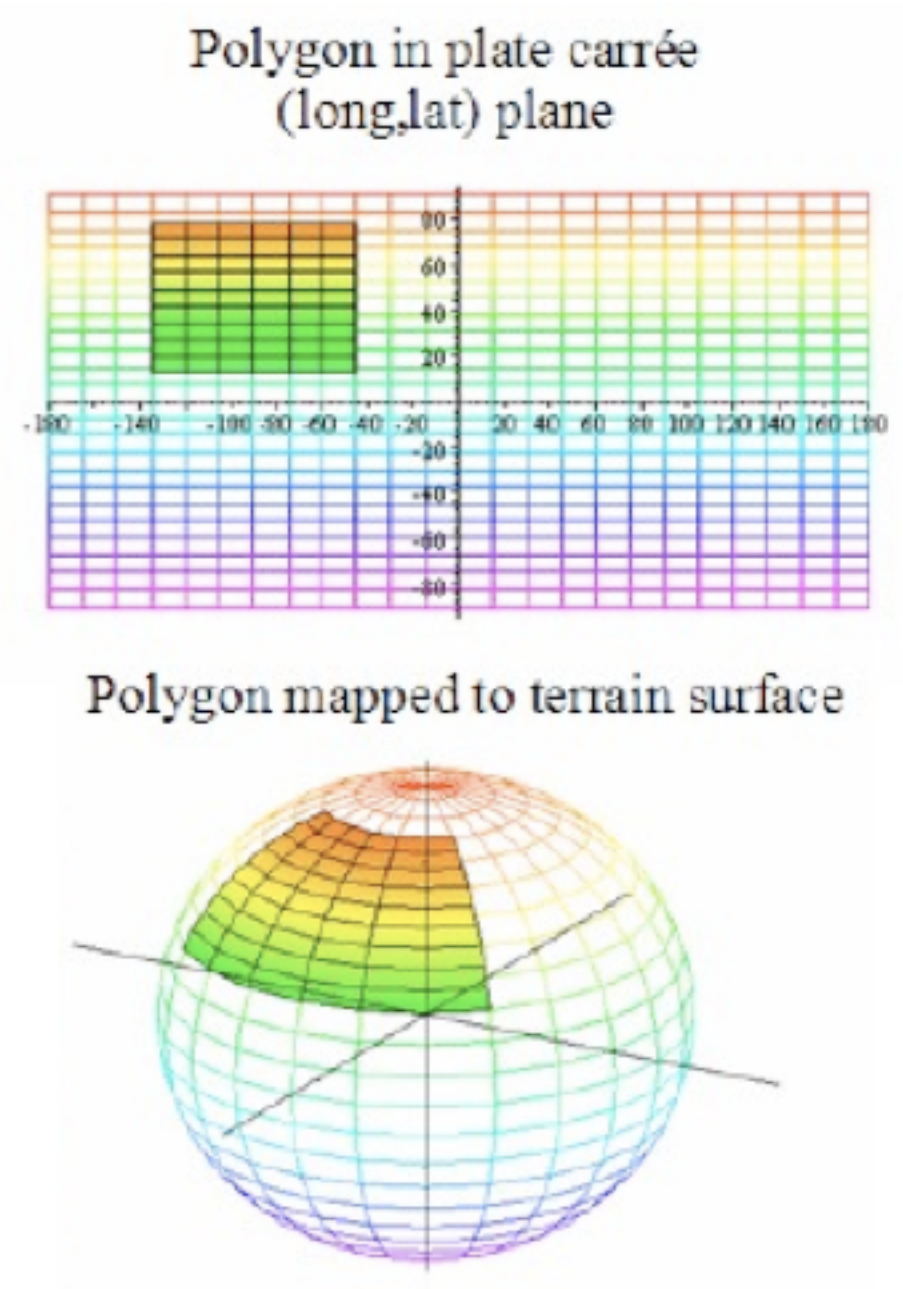
KML Capabilities

- Annotate the Earth
- Specify icons and labels to identify locations on the surface of the planet
- Create different camera positions to define unique views for KML features
- Define image overlays to attach to the ground or screen
- Define styles to specify KML feature appearance
- Write HTML descriptions of KML features, including hyperlinks and embedded images
- Organize KML features into hierarchies
- Locate and update retrieved KML documents from local or remote network locations
- Define the location and orientation of textured 3D objects

KML Content

- Model for encoding 2- and 3-dimensional geometries for use in 2-D mappers and 3-D virtual globe applications
- Uses latitude-longitude (based upon WGS84 datum) for encoding horizontal position
- Represents altitude in Meters (based upon the WGS84 ellipsoid and EGM96 geoid)

2D and 3D KML Sample




```
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
    <Placemark>
        <Polygon>
            <altitudeMode>
                clampToGround
            </altitudeMode>
            <outerBoundaryIs>
                <LinearRing>
                    <coordinates>
```

```
                        -135,78.5,300000
                        -135,12.5,300000
                        -45,12.5,300000
                        -45,78.5,300000
                        -135,78.5,300000
                    </coordinates>
                </LinearRing>
            </outerBoundaryIs>
        </Polygon>
    </Placemark>
</Document>
</kml>
```

KML Example

Example from: KML 2.2 Specification (fig. 6)

High-Level KML Content Types

Features including documents, folders, placemarks, network links

Geometries including points, linestrings, polygons, models, locations

Overlays including ground overlays, lat-lon boxes, photo overlays, screen overlays

Styles styles, substyles, icons, label styles

High-Level KML Content Types - cont.

Links read, update, create, delete, change

Views camera, look at

Time time span, timestamp

KML Demonstration and References

New Mexico State Boundary KML File | KMZ File (from NM RGIS)

New Mexico State Boundary KML File http://maps.google.com/maps?*q=http://karlbenedict.com/GEOG485-585/lectures/examples/tl_2010_35_state10.kml*

Google Code KML Documentation

OGC KML Implementation specification

OGC Web map Services - WMS

WMS - Overview

- Open Geospatial Consortium standard for requesting
  - Service Metadata (`GetCapabilities`) - an XML file representing information about a specific WMS service and its component layers

  - Map Images (`GetMap`) - graphic files representing one or more layers from a single WMS service for a specified area of interest, and, optionally, for a specified point in time
  - Feature Information (`GetFeatureInfo`) - a basic representation (in a variety of formats) of the attributes associated with a specific pixel location in a map image
- A WMS will return to the requesting system one of the above products OR an error message (in XML by default)
- Related Style Layer Descriptor standard supports dynamic updating of visualization options
- OGC WMS Documentation Access Page

WMS *GetCapabilities* Request

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
WMTVER=1.0.0	R				Request version
VERSION=version		O	O	O	Request version
SERVICE=WMS	R	R	R	R	Service type
REQUEST=capabilities	R				Request name
REQUEST=GetCapabilities		R	R	R	Request name
UPDATESEQUENCE=string		O	O	O	Sequence number or string for cache control
Vendor-specific parameters	O				Vendor-specific parameters

R=Required / O=Optional

WMS *GetMap* Request (Core)

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
WMTVER=1.0.0	R				Request version
VERSION=version		R	R	R	Request version.
REQUEST=map	R				Request name.
REQUEST=GetMap		R	R	R	Request name.
LAYERS=layer_list	R	R	R	R	Comma-separated list of one or more map layers. Optional (ver. 1.1, 1.1.1) if SLD parameter is present.
STYLES=style_list	R	R	R	R	Comma-separated list of one rendering style per requested layer. Optional if SLD parameter is present.
SRS=namespace:identifier	R	R	R		Spatial Reference System.
CRS=namespace:identifier				R	Spatial Reference System.
BBOX=minx,miny,maxx,maxy	R	R	R	R	Bounding box corners (lower left, upper right) in SRS units.
WIDTH=output_width	R	R	R	R	Width in pixels of map picture.

WMS GetMap Request (Core) - cont.

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
HEIGHT=output_height	R	R	R	R	Height in pixels of map picture.
FORMAT=output_format	R	R	R	R	Output format of map.
TRANSPARENT=TRUE or FALSE	O	O	O	O	Background transparency of map (default=FALSE).
BGCOLOR=color_value	O	O	O	O	Hexadecimal red-green-blue color value for the background color (default=0xFFFFFF).
EXCEPTIONS=exception_format	O	O	O	O	The format in which exceptions are to be reported by the WMS (default=XML).
TIME=time		O	O	O	Time value of layer desired.
ELEVATION=elevation		O	O	O	Elevation of layer desired.
Other sample dimensions		O	O	O	Values of other dimensions as appropriate.
Vendor specific parameters	O	O	O	O	Vendor specific parameters

WMS GetFeatureInfo Request

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
WMTVER=1.0.0	R				Request version.
VERSION=version		R	R	R	Request version.
REQUEST=feature_info	R				Request name.
REQUEST=GetFeatureInfo		R	R	R	Request name.
`<map_request_copy>`	R	R	R	R	Partial copy of the Map request parameters that generated the map for which information is desired
QUERY_LAYERS=layer_list	R	R	R	R	Comma-separated list of one or more layers to be queried.
INFO_FORMAT=output_format	O	O	O	R	Return format of feature information (MIME type).
FEATURE_COUNT=number	O	O	O	O	Number of features about which to return information (default=1).

WMS GetFeatureInfo Request - cont.

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
X=pixel_column	R	R	R		X coordinate in pixels of feature (measured from upper left corner=0)

I=pixel_column				R	i coordinate in pixels of feature in Map CS
Y=pixel_row	R	R	R		Y coordinate in pixels of feature (measured from upper left corner=0)
J=pixel_row				R	j coordinate in pixels of feature in Map CS
EXCEPTIONS=exception_format		O	O	O	The format in which exceptions are to be reported by the WMS (default=XML).
Vendor-specific parameters	O	O	O		Optional experimental parameters.

WMS Sample Requests - GetCapabilities

```
http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/services/ogc/wms?
    SERVICE=wms&
    REQUEST=GetCapabilities&
    VERSION=1.1.1
```

Live Link

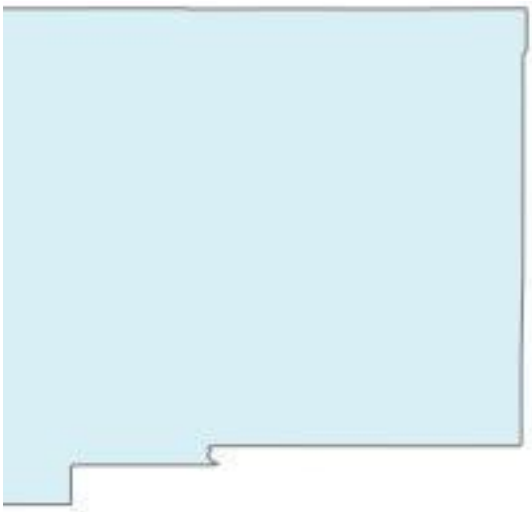
```
<?xml version='1.0' encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://schemas.opengis.net/wms/1.1.1/
WMS_MS_Capabilities.dtd"
 [
 <!ELEMENT VendorSpecificCapabilities EMPTY>
 ]>  <!-- end of DOCTYPE declaration -->

<WMT_MS_Capabilities version="1.1.1">

<!-- MapServer version 6.0.3 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML SUPPORTS=PROJ
SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=ICONV SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT
SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER
INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE -->

<Service>
  <Name>OGC:WMS</Name>
  <Title>rgis Dataset (6ca5428a-a78c-4c82-8120-da70dc92f2cc)</Title>
  <Abstract>WMS Service for rgis dataset State Boundary - 2010</Abstract>
        <KeywordList>
          <Keyword>rgis</Keyword>
          <Keyword> New Mexico</Keyword>
        </KeywordList>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://
  gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/services/ogc/wms"/>
  <ContactInformation>
    <ContactPersonPrimary>
      <ContactPerson>GStore Support</ContactPerson>
      <ContactOrganization>Earth Data Analysis Center</ContactOrganization>
    </ContactPersonPrimary>
      <ContactPosition>technical support</ContactPosition>
    <ContactAddress>
        <AddressType>Mailing address</AddressType>
```

```
        <Address>Earth Data Analysis Center, MSC01 1110, 1 University of New Mexico</Address>
        <City>Albuquerque</City>
        <StateOrProvince>NM</StateOrProvince>
        <PostCode>87131</PostCode>
        <Country>US</Country>
      </ContactAddress>
        <ContactVoiceTelephone>(505) 277-3622</ContactVoiceTelephone>
        <ContactFacsimileTelephone>(505) 277-3614</ContactFacsimileTelephone>
    <ContactElectronicMailAddress>devteam@edac.unm.edu</ContactElectronicMailAddress>
    </ContactInformation>
    <Fees>None</Fees>
    <AccessConstraints>none</AccessConstraints>
  </Service>

  <Capability>
    <Request>
      <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
        <DCPType>
          <HTTP>
            <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Get>
            <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Post>
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <GetMap>
        <Format>image/png</Format>
        <Format>image/gif</Format>
        <Format>image/jpeg</Format>
        <Format>image/png; mode=8bit</Format>
        <Format>image/tiff</Format>
        <Format>application/vnd.google-earth.kml+xml</Format>
        <Format>application/vnd.google-earth.kmz</Format>
        <DCPType>
          <HTTP>
            <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Get>
            <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Post>
          </HTTP>
        </DCPType>
      </GetMap>
      <GetFeatureInfo>
        <Format>text/plain</Format>
        <Format>application/vnd.ogc.gml</Format>
        <DCPType>
          <HTTP>
            <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
```

```
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Get>
            <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Post>
          </HTTP>
        </DCPType>
      </GetFeatureInfo>
      <DescribeLayer>
        <Format>text/xml</Format>
        <DCPType>
          <HTTP>
            <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Get>
            <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Post>
          </HTTP>
        </DCPType>
      </DescribeLayer>
      <GetLegendGraphic>
        <Format>image/png</Format>
        <Format>image/gif</Format>
        <Format>image/jpeg</Format>
        <Format>image/png; mode=8bit</Format>
        <DCPType>
          <HTTP>
            <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Get>
            <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Post>
          </HTTP>
        </DCPType>
      </GetLegendGraphic>
      <GetStyles>
        <Format>text/xml</Format>
        <DCPType>
          <HTTP>
            <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Get>
            <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href=
            "http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/
            services/ogc/wms?"/></Post>
          </HTTP>
        </DCPType>
      </GetStyles>
    </Request>
    <Exception>
      <Format>application/vnd.ogc.se_xml</Format>
      <Format>application/vnd.ogc.se_inimage</Format>
```

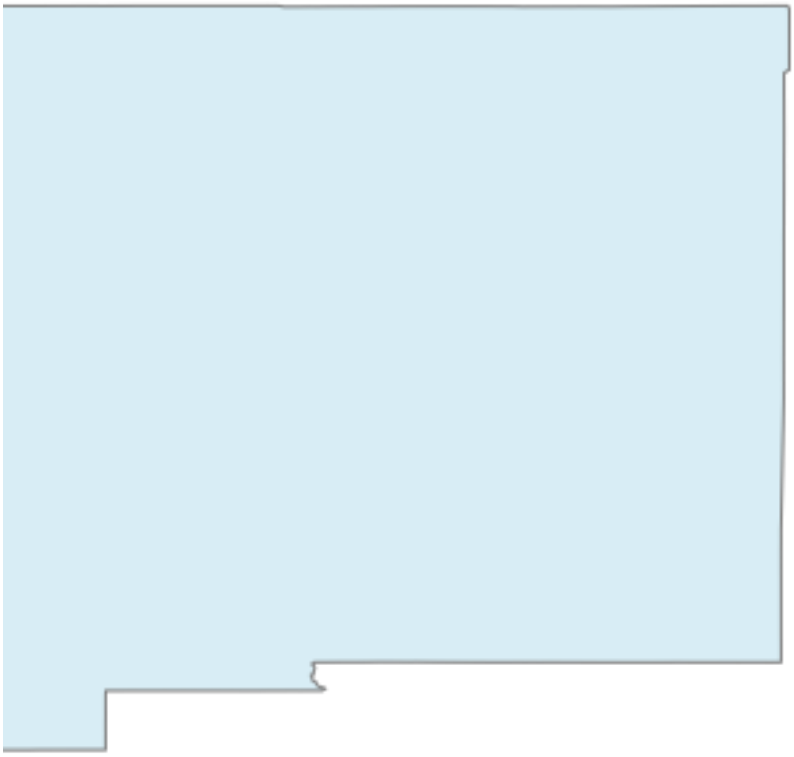
```
    <Format>application/vnd.ogc.se_blank</Format>
  </Exception>
  <VendorSpecificCapabilities />
  <UserDefinedSymbolization SupportSLD="1" UserLayer="0" UserStyle="1" RemoteWFS="0"/>
  <Layer>
    <Name>RGIS_Dataset</Name>
    <Title>rgis Dataset (6ca5428a-a78c-4c82-8120-da70dc92f2cc)</Title>
    <Abstract>WMS Service for rgis dataset State Boundary - 2010</Abstract>
    <KeywordList>
     <Keyword>rgis</Keyword>
     <Keyword> New Mexico</Keyword>
    </KeywordList>
    <SRS>EPSG:4269</SRS>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:4267</SRS>
    <SRS>EPSG:26913</SRS>
    <SRS>EPSG:26912</SRS>
    <SRS>EPSG:26914</SRS>
    <SRS>EPSG:26713</SRS>
    <SRS>EPSG:26712</SRS>
    <SRS>EPSG:26714</SRS>
    <SRS>EPSG:3857</SRS>
    <LatLonBoundingBox minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
    <BoundingBox SRS="EPSG:4326"
                minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
    <Layer queryable="0" opaque="0" cascaded="0">
        <Name>tl_2010_35_state10</Name>
        <Title>tl_2010_35_state10</Title>
        <Abstract>State Boundary - 2010</Abstract>
        <KeywordList>
          <Keyword></Keyword>
        </KeywordList>
        <SRS>epsg:4326</SRS>
        <LatLonBoundingBox minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
        <BoundingBox SRS="epsg:4326"
                    minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
        <MetadataURL type="FGDC-STD-001-1998">
          <Format>text/xml</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/
          6ca5428a-a78c-4c82-8120-da70dc92f2cc/metadata/FGDC-STD-001-1998.xml"/>
        </MetadataURL>
    </Layer>
  </Layer>
</Capability>
</WMT_MS_Capabilities>
```

WMS Sample Requests - GetMap



```
http://gstore.unm.edu/apps/rgis/datasets/
6ca5428a-a78c-4c82-8120-da70dc92f2cc/
services/ogc/wms?
    VERSION=1.1.1&
    SERVICE=WMS&
    REQUEST=GetMap&
    BBOX=-109,31,-102.9,37.1&
    LAYERS=tl_2010_35_state10&
    WIDTH=200&
    HEIGHT=200&
    SRS=EPSG:4326&
    FORMAT=image/jpeg&
    STYLES=
```

link



```
http://gstore.unm.edu/apps/rgis/datasets/
6ca5428a-a78c-4c82-8120-da70dc92f2cc/
services/ogc/wms?
    VERSION=1.1.1&
    SERVICE=WMS&
    REQUEST=GetMap&
    BBOX=-109,31,-102.9,37.1&
    LAYERS=tl_2010_35_state10&
    WIDTH=300&
    HEIGHT=300&
    SRS=EPSG:4326&
    TRANSPARENT=TRUE&
    FORMAT=image/png&
    STYLES=
```

link

Integraton of WMS and KML

- The KML GroundOverlay element may be used to integrate a network accessible map image into a client
- A WMS service may be used to as the source of a KML GroundOverlay element
- KML includes parameterizations that allow for dynamic generation of WMS requests using client bounding box information
- Time-enabled WMS may be accessed through use of manually configured time parameters in WMS URLs and TimeStamp or TimeSpan KML elements

Sample WMS-KML Integration

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2"
    xmlns:kml="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom">
    <GroundOverlay>
        <name>RGIS Counties WMS</name>
        <Icon>
            <href>http://gstore.unm.edu/apps/rgis/datasets/107046/services/ogc/wms?
            VERSION=1.1.1&amp;amp;SERVICE=WMS&amp;amp;REQUEST=GetMap&amp;amp;BBOX=-109,31,-102.9,37.1
            &amp;amp;LAYERS=tl_2010_35_state10&amp;amp;WIDTH=800&amp;amp;HEIGHT=800&amp;amp;SRS=EPSG:4326
            &amp;amp;FORMAT=image/png&amp;amp;STYLES=</href>
            <viewRefreshMode>onStop</viewRefreshMode>
        </Icon>
        <LatLonBox>
            <north>37.32753828398865</north>
            <south>30.86418272137246</south>
            <east>-101.3630220689848</east>
            <west>-110.6891149310152</west>
        </LatLonBox>
    </GroundOverlay>
</kml>
```

Sample KML File

Module 4.2 - Interoperability Standards - WFS & WCS

Outline

- OGC Web Feature Services (WFS)
  - Capabilities and purpose
  - Overview of the collection of WFS commands
  - Sample WFS requests
- OGC Web Coverage Services (WCS)
  - Capabilities and purpose
  - Overview of the collection of WCS commands
  - Sample WCS requests

OGC Web Feature Service (WFS)

Background

The documents related to the OGC WFS standard are available from: http://www.opengeospatial.org/standards/wfs and all operation parameter tables presented here are based upon the OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

From the Version 2.0.0 WFS Scope Section:

> This International Standard specifies the behaviour of a service that provides transactions on and access to geographic features in a manner independent of the underlying data store. It

specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions.

Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.

Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.

Locking operations allow exclusive access to features for the purpose of modifying or deleting features.

Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.

Stored query operations allow clients to create, drop, list and described parameterized query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

WFS Requests/Operations

These request types are submitted as part of the required `REQUEST` key in a KVP HTTP GET request.

`GetCapabilities` service metadata (XML) that documents the types of features supported by the service and the operations supported by each feature type

`DescribeFeatureType` metadata (XML) that describes the structure of supported feature types

`GetPropertyValue` a request for the value(s) of a specified property for a specified *featuretype*

`GetFeature (GetFeatureWithLock)` a request for actual features (XML, or other formats) from the service. The request may include both spatial and non-spatial query constraints

`LockFeature` Feature locking operation

WFS Requests/Operations - cont.

`Transaction` a request to a WFS that may create, update, or delete features

`CreateStoredQuery` a request to create a named WFS query that is stored on the server for future reuse

`DropStoredQuery` a request to remove a named WFS query that has previously been stored on the server

`ListStoredQueries` a request to retrieve a list of named WFS queries that have been stored on the server

`DescribeStoredQueries` a request for more detailed information about specific named WFS queries that are stored on the server

WFS Conformance Levels

WFS 2.0.0 Requests and their corresponding WFS Compliance Levels

Operation (`REQUEST=`)	V 1.1.0	V 2.0.0	Simple	Basic	Transactional	Locking
`GetCapabilities`	✓	✓	✓	✓	✓	✓
`DescribeFeatureType`	✓	✓	✓	✓	✓	✓
`ListStoredQueries`		✓	✓	✓	✓	✓
`DescribeStoredQueries`		✓	✓	✓	✓	✓
`GetFeature`	✓	✓	✓	✓	✓	✓
`StoredQuery`		✓	✓	✓	✓	✓
`GetPropertyValue`		✓		✓	✓	✓
`Transaction`	✓	✓			✓	✓
`GetFeatureWithLock`	✓	✓				✓
`LockFeature`	✓	✓				✓
`GetGMLObject`	✓					

Request Composition

Requests submitted to a WFS may be submitted either via

HTTP GET a request that includes all request parameters within the URL submitted to the service. Request parameters are included in the URL as "key=value" pairs (KVPs)

HTTP POST a request where the URL consists of only the Host and path, with all other request parameters included in the body of the POST document submitted to the service. The request parameters supplied to the server are encoded as XML within the POST document.

SOAP a request submitted as an encapsulated message within a SOAP transaction.

Servers implementing WFS may support either the HTTP GET, POST, or SOAP request model

Conceptually *FeatureType* = *Layer*

KVP for Base WFS Requests

Base request parameters for all HTTP GET KVP requests

Table 4 — KVP-encoding of the base request type

URLComponent	Operation	O/M[a]	Description
SERVICE	All operations.	M	See 7.6.2.4.
VERSION[b] (All operations)	All operations except GetCapabilities.	M	See 7.6.2.5.

[a] O = Optional, M = Mandatory

[b] VERSION is mandatory for all operations except the GetCapabilities operation.

`VERSION` is required for all operations *except* the `GetCapabilities` request

Sample GetCapabilities Requests

Sample request to USGS Framework Layer (Governmental Units) WFS Service linked from the USGS Framework Web Feature Services web page - Live Link

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    request=GetCapabilities&
    service=WFS
```

Sample request to NM RGIS (NM 2010 Census Block Groups) - Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/715663ba-c1c3-414c-84a7-c671526f8316/services/ogc/wfs?
    SERVICE=wfs&
    REQUEST=GetCapabilities&
    VERSION=1.0.0
```

KVP for DescribeFeatureType Request

`DescribeFeatureType` HTTP GET KVP request

Table 15 — DescribeFeatureType KVP encoding

URL Component	O/M[a]	Description
Common Keywords (REQUEST=DescribeFeatureType)		See Table 7. (Only keywords for all operations or the DescribeFeatureType operation.)
TYPENAME	O	A comma separated list of feature types to describe. If no value is specified, the complete application schema offered by the server shall be described.
OUTPUTFORMAT	O	Shall support the value "application/gml+xml; version=3.2" indicating that a GML (see ISO19136:2007) application schema shall be generated. A server may support other values to which this International Standard does not assign any meaning.

[a] O = Optional, M = Mandatory

Sample DescribeFeatureType Requests

USGS Framework Layer (Governmental Units) WFS Service linked from the USGS Framework Web Feature Services web page - Live Link

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    version=1.1.0&
    request=DescribeFeatureType&
    service=WFS&
    typeName=WFS_govunits:State_or_Territory_High-res
```

Sample request to NM RGIS (NM 2010 Census Block Groups) - Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/715663ba-c1c3-414c-84a7-c671526f8316/services/ogc/wfs?
    VERSION=1.0.0&
    SERVICE=wfs&
    REQUEST=DescribeFeatureType&
    TYPENAME=tl_2010_35_bg10
```

KVP for GetFeature Request

`GetFeature` HTTP GET KVP request

Table 17 — Keywords for GetFeature KVP-encoding

URL Component	Description
Common Keywords (REQUEST=GetFeature)	See Table 7 for additional parameters that may be used in a KVP-encoded GetFeature request.
Standard Presentation Parameters	See Table 5.
Standard Resolve Parameters	See Table 6.
Adhoc Query Keywords (Mutually exclusive with Stored Query Keywords)	See Table 8.
Stored Query Keywords (Mutually exclusive with Adhoc Query Keywords)	See Table 10.

KVP for GetFeature Request - Presentation Parameters

Table 5 — KVP-encoding of standard presentation parameters

URLComponent	Operation	O/M[a]	Default	Description
STARTINDEX	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.4.
COUNT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.5.
OUTPUTFORMAT	DescribeFeatureType, GetPropertyValue, GetFeature, GetFeatureWithLock	O	application/gml+xml; version=3.2	See 7.6.3.7.
RESULTTYPE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	results	See 7.6.3.6.
[a] O = Optional, M = Mandatory				

KVP for GetFeature Request - Resolve Parameters

Table 6 — KVP encoding of standard resolve parameters

URLComponent	Operation	O/M[a]	Default	Description
RESOLVE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	None	See 7.6.4.4.
RESOLVEDEPTH	GetPropertyValue, GetFeature, GetFeatureWithLock	O	*	See 7.6.4.5. RESOLVE parameter shall have a value other than “none”.
RESOLVETIMEOUT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	Server Specific (see ResolveTimeoutDefault, Table 14)	See 7.6.4.6. RESOLVE parameter shall have a value other than “none”.
[a] O = Optional, M = Mandatory				

KVP for GetFeature Request - Ad-hoc Query Parameters

Table 8 — Keywords for Ad hoc query KVP-encoding

URL Component	O/M[a]	Description
TYPENAMES	M[b]	See 7.9.2.4.1.
ALIASES	O	See 7.9.2.4.3.
SRSNAME	O	See 7.9.2.4.4.
Projection clause	O	See Table 9.
FILTER	O	See ISO 19143:2010, 6.3.3.
FILTER_LANGUAGE	O	See ISO 19143:2010, 6.3.3.
RESOURCEID	O	See ISO 19143:2010, 6.3.3.
BBOX	O	See OGC 06-121r3.
SORTBY	O	See ISO 19143:2010, Clause 8 The SORTBY parameter is used to specify a list of property names whose values should be used to order (upon presentation) the set of feature instances that satisfy the query. The value of the SORTBY parameter shall have the form *"PropertyName [ASC|DESC][,PropertyName [AASC|DESC],]"* where the letters ASC are used to indicate an ascending sort and the letters DESC are used to indicate a descending sort. If neither ASC nor DESC are specified, the default sort order shall be ascending. An example value might be: *"SORTBY=Field1 DESC,Field2 DESC,Field3"*. In this case the results are sorted by Field 1 descending, Field2 descending and Field3 ascending

a. O = Optional , M = Mandatory

b. The TYPENAMES parameter is mandatory in all cases except when the RESOURCEID parameter is specified (see 7.9.2.4.1).

KVP for GetFeature Request - Stored Query Parameters

Table 10 — Keywords for Stored query KVP-encoding

URL Component	O/M[a]	Description
STOREDQUERY_ID	M	The identifier of the stored query to invoke.
storedquery_parameter=value	O	Each parameter of a stored query shall be encoded in KVP as a keyword-value pair. Stored query parameters shall not have names that conflict with any WFS parameter name.

a O = Optional, M = Mandatory

Sample GetFeature Requests

USGS Framework Layer (Governmental Units) WFS Service linked from the USGS Framework Web Feature Services web page - Live Link

Note: `TYPENAME` for `VERSION=1.1.0` instead of `TYPENAMES` for `VERSION=2.0.0`

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    VERSION=1.1.0&
    REQUEST=GetFeature&
    SERVICE=WFS&
    TYPENAME=WFS_govunits:State_or_Territory_High-res
```

Alternative request (Live Link) that includes an `OUTPUTFORMAT` parameter

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    VERSION=1.1.0&
    REQUEST=GetFeature&
    SERVICE=WFS&
    TYPENAME=WFS_govunits:State_or_Territory_High-res&
    OUTPUTFORMAT=text/xml;%20subType=gml/3.1.1/profiles/gmlsf/1.0.0/0
```

OGC Web Coverage Services

Background

The documents related to the OGC WCS standard are available from: [http://www.opengeospatial.org/standards/wcs][wcs] with the sample parameters in the following slides based upon the *OGC Web Coverage Service 2.0 Interface Standard - KVP Protocol Binding Extension* - Peter Baumann, editor - 2010-10-27

From the OGC WCS 2.0 *Introduction*

> The OGC Web Coverage Service (WCS) supports electronic retrieval of geospatial data as "coverages" – that is, digital geospatial information representing space/time-varying phenomena.
>
> This document specifies the WCS core; every implementation of a WCS shall adhere to this standard. This standard thus defines only basic requirements. Extensions to the core will define extensions to meet additional requirements, such as the response encoding. Indeed, additional extensions are required in order to completely specify a WCS for implementation.
>
> A WCS provides access to coverage data in forms that are useful for client-side rendering, as input into scientific models, and for other clients. The WCS may be compared to the OGC Web Feature Service (WFS) and the Web Map Service (WMS). As WMS and WFS service instances, a WCS allows clients to choose portions of a server's information holdings based on spatial constraints and other query criteria.

WCS Requests/Operations

`GetCapabilities` service metadata (XML) that documents the service, including brief information about the data coverages available from the service

`DescribeCoverage` a request for more detailed metadata (XML) for one or more coverages listed in the output of the GetCapabilities request

`GetCoverage` a request for an actual data product representing a specified coverage. The specific data formats available for delivery will vary from service to service.

Request Composition

Requests submitted to a WCS may be submitted either via the following protocols, as defined in the three extensions developed thus far for the *core* WCS standard.

HTTP GET a request that includes all request parameters within the URL submitted to the service. Request parameters are included in the URL as "name=value" pairs. Extension Link

HTTP POST a request where the URL consists of only the Host and path, with all other request parameters included in the body of the POST document submitted to the service. The request parameters supplied to the server are encoded as XML within the POST document. Extension Link

XML/SOAP a request-response model between the client that conforms with the W3C SOAP web services protocol Extension Link

KVP for Base WCS Requests

Name	Mandatory/Optional	Definition	Data Type
service	M	Identifier of the OGC service	`String`, fixed to "WCS"
request	M	Request type name	`String`, set to operation name
version	M (except for GetCapabilities)	Request protocol version	`String`

Sample WCS `GetCapabilities` requests

NOAA Global Forecast System THREDDS catalog. Live Link

```
http://nomads.ncdc.noaa.gov/thredds/wcs/gfs-004/201403/20140301/gfs_4_20140301_1200_159.grb2?
    service=WCS&
    version=1.0.0&
    request=GetCapabilities
```

New Mexico Resource Geographic Information System PRISM Precipitation Normals WCS Service. Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/2ce10b57-3925-4971-b876-b6fc66d3cca2/services/ogc/wcs?
    SERVICE=wcs&
    REQUEST=GetCapabilities&
    VERSION=1.1.2
```

KVP for DescribeCoverage Request

`DescribeCoverage` HTTP GET KVP request

Table 1 — DescribeCoverage request URL encoding

Name	Definition	Data type	Multiplicity
service	Identifier of the OGC service	String, fixed to "WCS"	One (mandatory)
version	Request protocol version	String	One (mandatory)
request	Request type name	String, fixed to "DescribeCoverage"	One (mandatory)
coverageId	List of coverage identifiers to be described	Comma-separated NCName list	One (mandatory)

Sample DescribeCoverage Request

NOAA Global Forecast System THREDDS catalog. Live Link

```
http://nomads.ncdc.noaa.gov/thredds/wcs/gfs-004/201403/20140301/gfs_4_20140301_1200_159.grb2?
    service=WCS&
    version=1.0.0&
    request=DescribeCoverage&
    COVERAGE=Categorical_Rain
```

New Mexico Resource Geographic Information System PRISM Precipitation Normals WCS Service. Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/2ce10b57-3925-4971-b876-b6fc66d3cca2/services/ogc/wcs?
    SERVICE=wcs&
    REQUEST=DescribeCoverage&
    VERSION=1.1.2&
    COVERAGE=us_ppt_1971_2000_11
```

KVP for GetCoverage Request

GetCoverage HTTP GET KVP request

Table 2 — *GetCoverage* request KVP encoding

Name	Definition	Data type	Multiplicity
service	Identifier of the OGC service	String, fixed to "WCS"	one (mandatory)
version	Request protocol version	String	one (mandatory)
request	Request type name	String, fixed to "GetCoverage"	one (mandatory)
coverageId	Identifier of coverage to be inspected	NCName	one (mandatory)
subset	boundaries of coverage subset	SubsetSpec as defined in Requirement 7	zero or more (optional)

Subset Definition for GetCoverage Request

Subset definition for the `GetCoverage` HTTP GET KVP request

Requirement 7 /req/get-kvp/getCoverage-request-subsetspec:
Each `SubsetSpec` **shall** adhere to this EBNF syntax:

```
SubsetSpec:      dimension [ , crs ] ( intervalOrPoint )
dimension:       NCName
crs              anyURI
intervalOrPoint: interval | point
interval:        low , high
low:             point | *
high:            point | *
point:           number| "token"  // " = ASCII 0x42
```

Example from the 2.0 specification:

```
http://www.myserver.org:port/path?
    service=WCS
    &version=2.0
    &request=GetCoverage
    &coverageId=C0002
    &subset=lon,http://www.opengis.net/def/crs/EPSG/0/4326(-71,47)
    &subset=lat,http://www.opengis.net/def/crs/EPSG/0/4326(-66,51)
   &subset=t,http://www.opengis.net/def/trs/ISO- 8601/0/Gregorian+UTC("2009-11-06T23:20:52Z")
```

Sample GetCoverage Request

New Mexico Resource Geographic Information System PRISM Precipitation Normals WCS Service. Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/2ce10b57-3925-4971-b876-b6fc66d3cca2/services/ogc/wcs?
    SERVICE=wcs&
    REQUEST=GetCoverage&
    VERSION=1.1.2&
    COVERAGE=us_ppt_1971_2000_11&
    CRS=urn:ogc:def:crs:EPSG::4326&
    BBOX=24.0625,-125.02083333333,49.93749998965,-66.47916669008&
    FORMAT=image/tiff&
    WIDTH=2048&
    HEIGHT=905
```

Module 4.3 - Interoperability Standards - Desktop GIS Integration

Overview

- Common Model for Client Configuration for Connections to Remote OGC Services
- Specific Client Examples

Quantum GIS (QGIS)

- WMS
- WFS
- WCS

ArcGIS

- WMS
- WFS
- WCS

Common Model

Based upon the results of a GetCapabilities request against a remote service. GetCapabilities request information provided as either:

- The base URL to which the OGC service parameters would be added
- A complete GetCapabilities request against the service

Full GetCapabilities Request

NASA Earth Observations (NEO) Imagery WMS

`http://neowms.sci.gsfc.nasa.gov/wms/wms?version=1.3.0&service=WMS&request=GetCapabilities`

USGS Framework Services

`http://frameworkwfs.usgs.gov/framework/wms/wms.cgi?SERVICE=wms&REQUEST=GetCapabilities`

Base URL for GetCapabilities

NASA Earth Observations (NEO) Imagery WMS

`http://neowms.sci.gsfc.nasa.gov/wms/wms?`

USGS Framework Services

`http://frameworkwfs.usgs.gov/framework/wms/wms.cgi?`

Quantum GIS (QGIS)

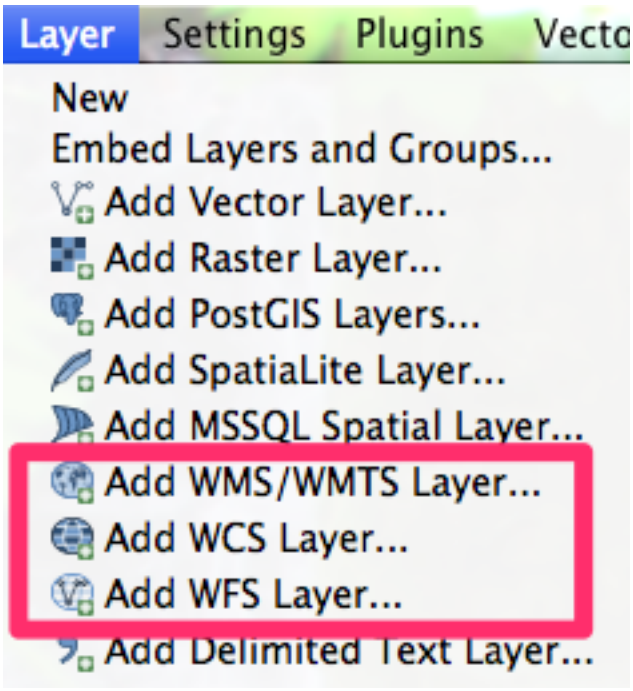
QGIS uses the Base URL approach for adding WMS, WFS or WCS layers to a project.

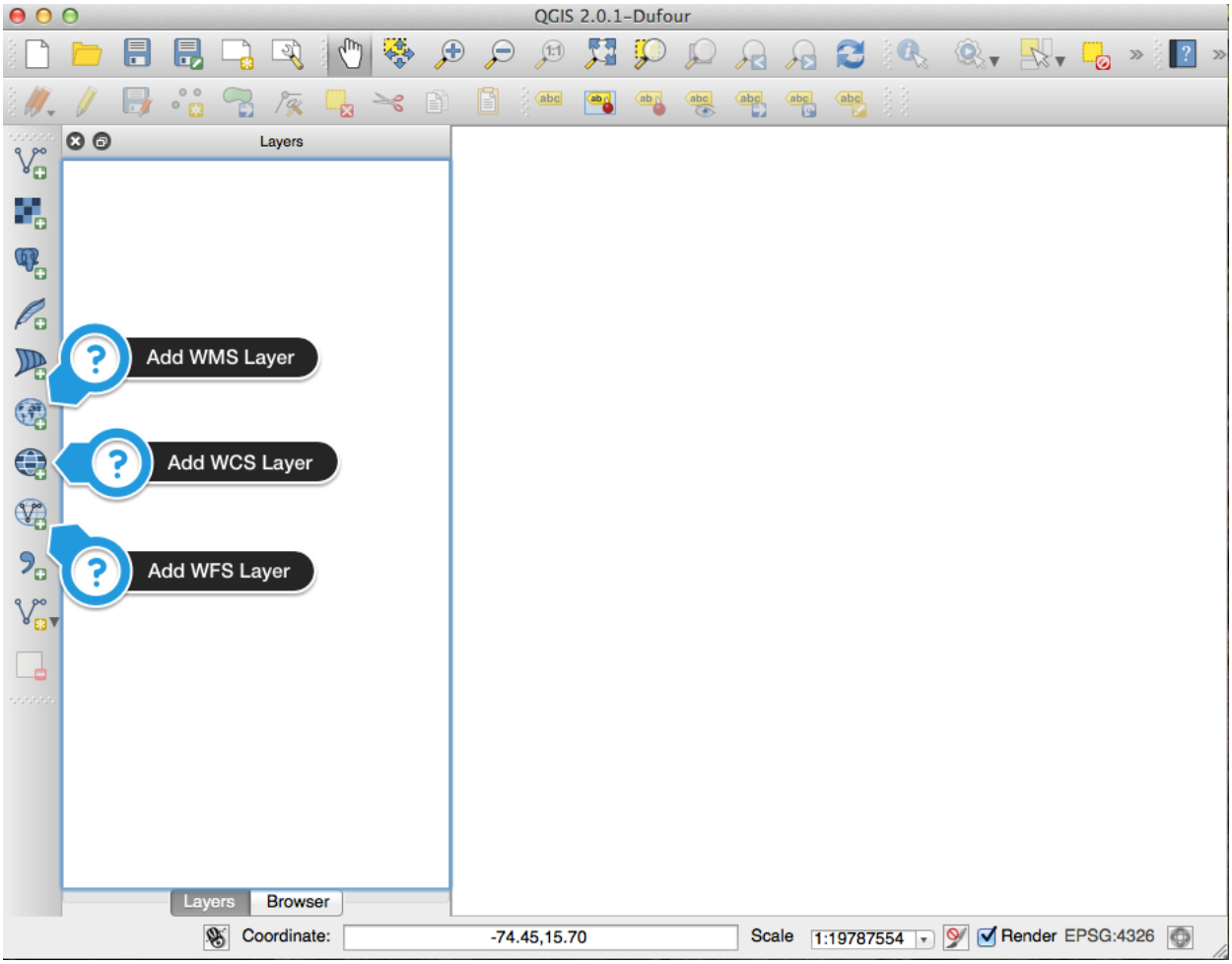
The General Process:

1. Add service, or select existing service
2. Connect to the service to retrieve the information from the GetCapabilities response for the service
3. Select layer(s)
4. Modify settings for layer(s)
5. Add layer(s)

QGIS OGC Documentation

QGIS - Adding Services and Layers - start





You need to know the GetCapabilities request for the service you want to add, for example one of the USGS Framework WMS services

```
http://frameworkwfs.usgs.gov/framework/wms/wms.cgi?
SERVICE=wms&REQUEST=GetCapabilities
```

determine the base URL -

```
http://frameworkwfs.usgs.gov/framework/wms/wms.cgi
```

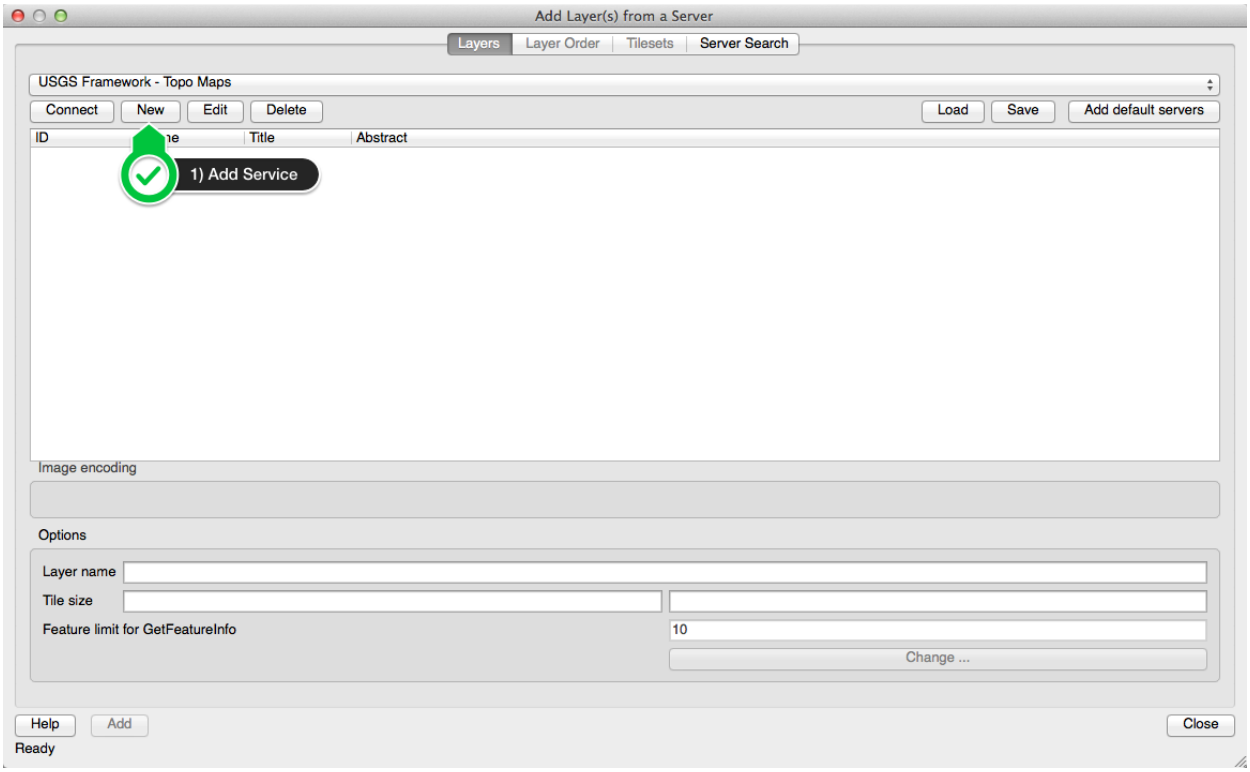
in this case

If in doubt, check the information in the metadata

Select the layer type you would like to from "Layer" menu, or click the button in the interface to add a specific layer type.

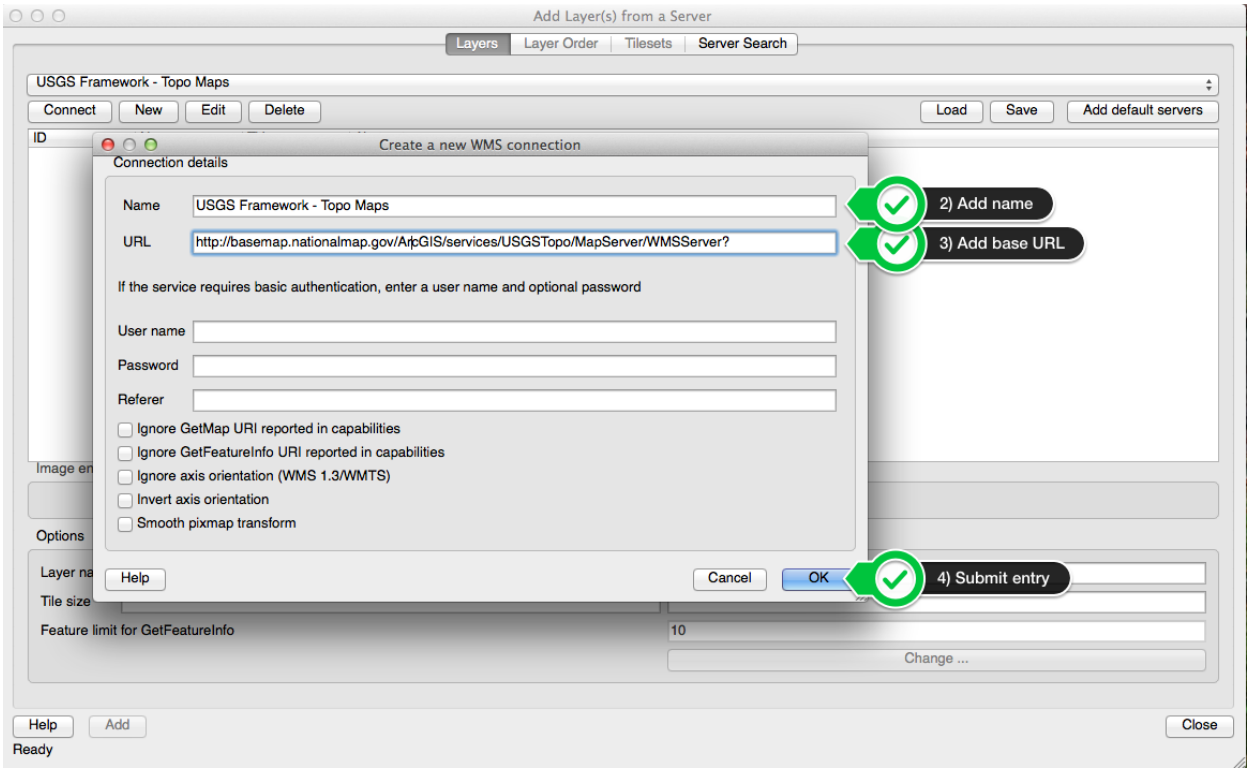
QGIS - Adding Services and Layers - adding a service

Add a service to the list of services in the menu (if necessary - QGIS retains information about previously added services) by selecting the "New" option under the service list in the "Add Layer(s) from a Server" dialog



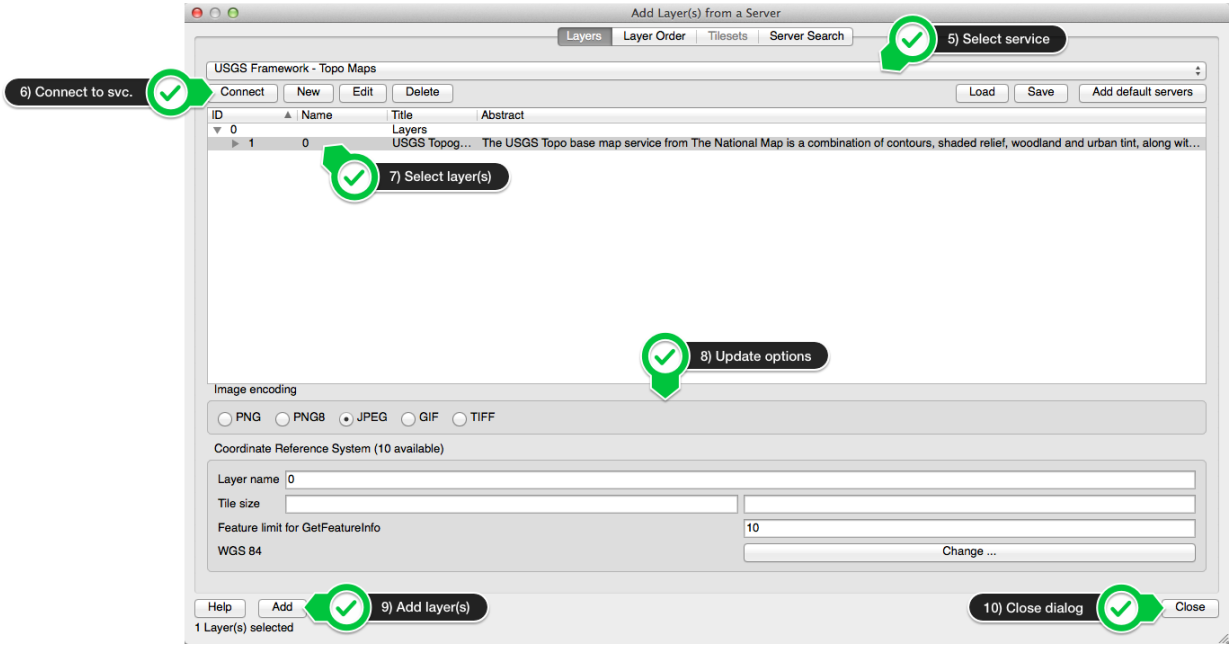
QGIS - Adding Services and Layers - adding connection information

Add the name, base URL and any additional information about the service to the connection dialog box



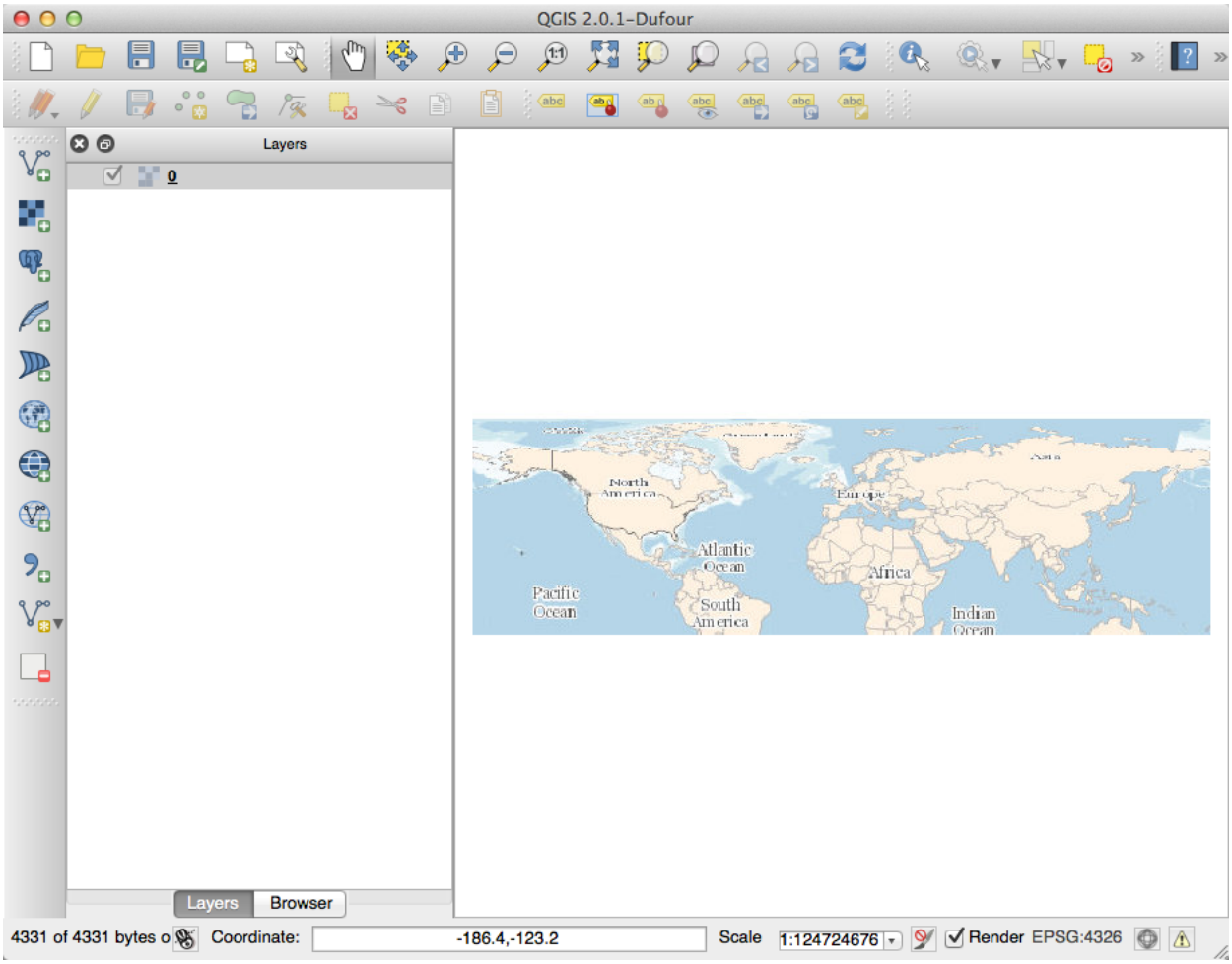
QGIS - Adding Services and Layers - connecting to and adding layers from the service

After adding the service, you can select it from the service list in the "Add Layer(s) from a Server" dialog box, connect to the service to retrieve the GetCapabilities response from the service, select the layers and other options advertised by the service through its response, and add them to your map.



QGIS - Adding Services and Layers - the final added layer

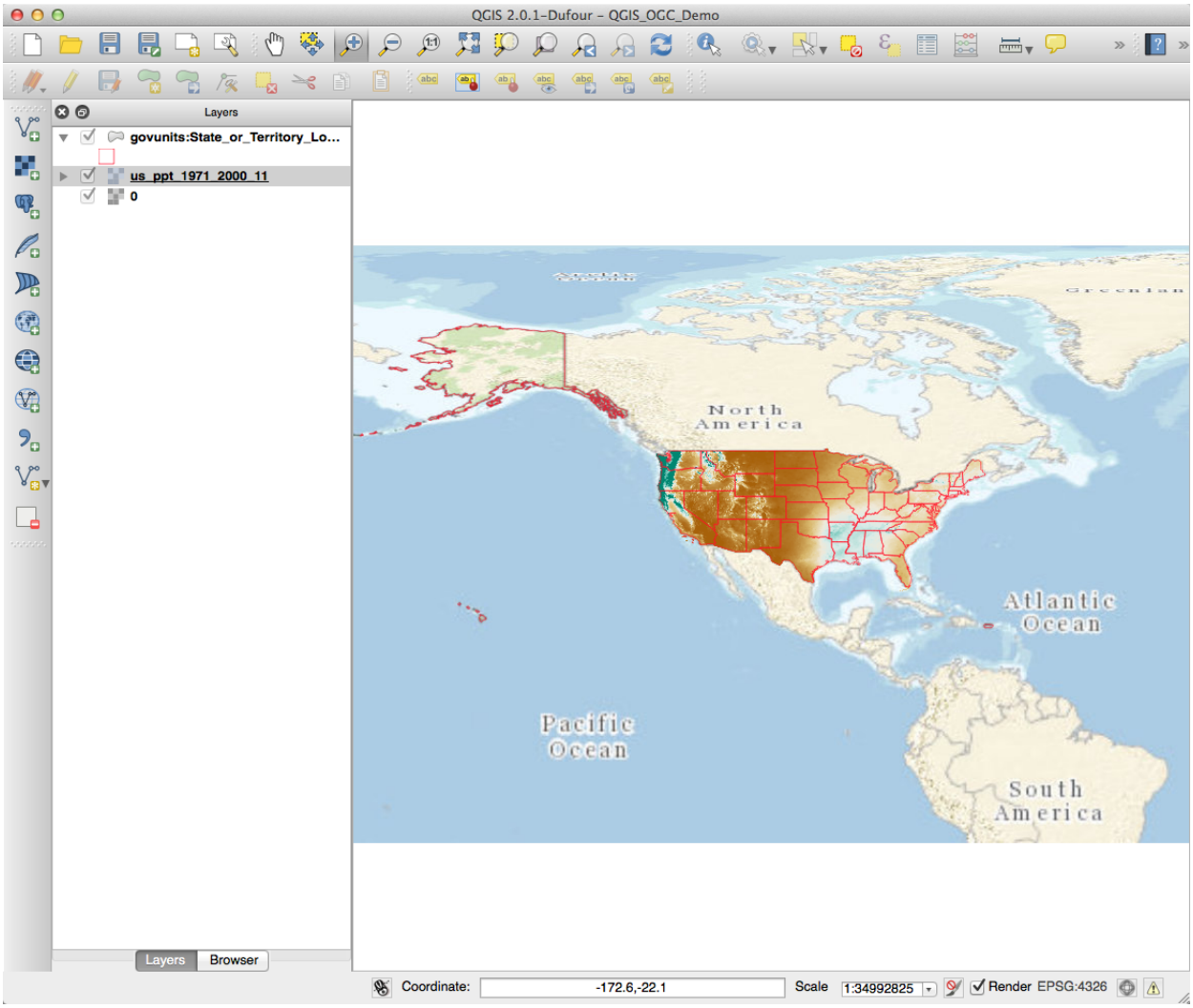
After adding the layer, it appears as an available layer in the table of contents for your map.



QGIS Demonstration with WMS, WFS and WCS Services

WMS, WFS and WCS in QGIS

Example



ArcGIS

Based upon the results of a GetCapabilities request against a remote service. GetCapabilities request information provided as either:

- The base URL to which the OGC service parameters would be added
- A complete GetCapabilities request against the service

This model applies to ArcGIS just as it did for Quantum GIS - the base URL is provided to the various ArcGIS components that support the addition of OGC services to the client interface.

ArcGIS OGC Support Documentation

ArcGIS WMS and WCS Configuration

The addition of OGC WMS and WCS layers to ArcMap is through the same process of

- Select the “add data” button

- WMS/WCS services are added through the "GIS Servers" option in the "Add Data" dialog
- If you have not previously added the service from which you want to add layers, you select "Add WMS Server" or "Add WCS Server" from the list of options in the "Add Data" dialog
  - You then provide the BASE GetCapabilities URL to the "ADD WMS/WCS Server" dialog that appears
  - Click "OK", and the new WMS/WCS service is added to the list of services that is available when you choose to add a WMS service.
- You then select the layer(s) from the service that you want to add to your map and click the "add" button in the dialog.

ArcGIS WMS and WCS Configuration Resources

- Adding WMS Services to ArcMap 10.2

ESRI Documentation

- Adding WCS Services to ArcMap 10.2

ESRI Documentation

ArcGIS WFS Configuration

- WFS support in ArcGIS 10.0 and beyond requires that the "Data Interoperability Extension" be installed (though it doesn't have to be enabled)
- Connections to WFS services are defined through ArcCatalog's "Interoperabilty Connections" "Add Interoperability Connection" option
- After defining the connection in ArcCatalog (including the specification of the interoperabilty connection type, desired feature types, and maximum number of features to return), its feature types are available through that Interoperability Connection that may be added to ArcMap and other ArcGIS components
- Once the connection is created, WFS data may be added through the "Add Data" dialog in ArcMap

ArcGIS WFS Configuration Resources

- Steps for connecting to an OGC WFS from within ArcCatalog 10.2

ESRI Documentation

- Steps for adding a WFS service to ArcMap 10.2

ESRI Documentation

Conclusions

- A GetCapabilities request is the key for configuring most OGC client applications to access remote services
- The specific way in which the GetCapabilities request is given to the client varies from client to client
- Clients can auto/*mis*-configure themselves based upon the GetCapabilities XML response - when troubleshooting problems with an advertised service, try the manual request approach for the GetCapabilities, data and maps that you have learned about to determine if the service is functioning as advertised.

List of Figures

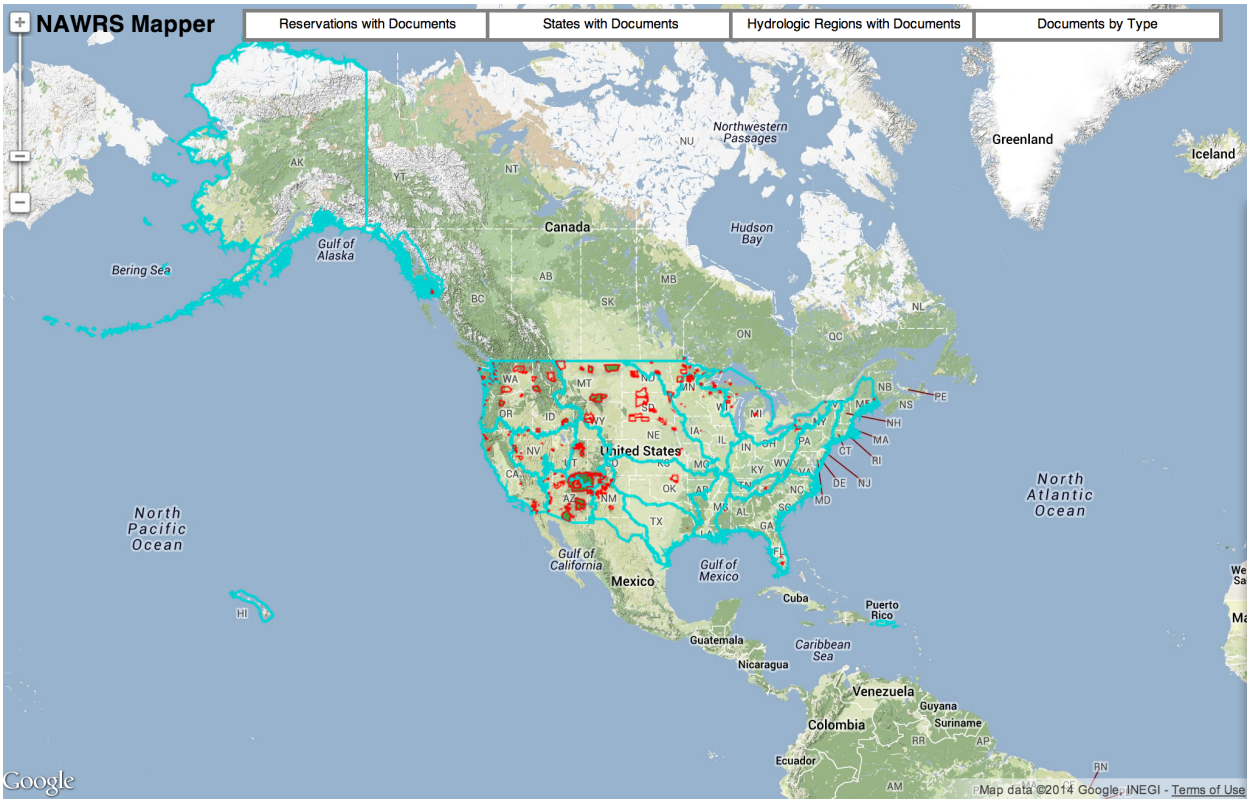


Figure 1: NAWRS Mapper. *HTML*: 39 Lines; *CSS*: 136 Lines; *core.js*: 515 Lines + Google Maps API and JQuery Framework

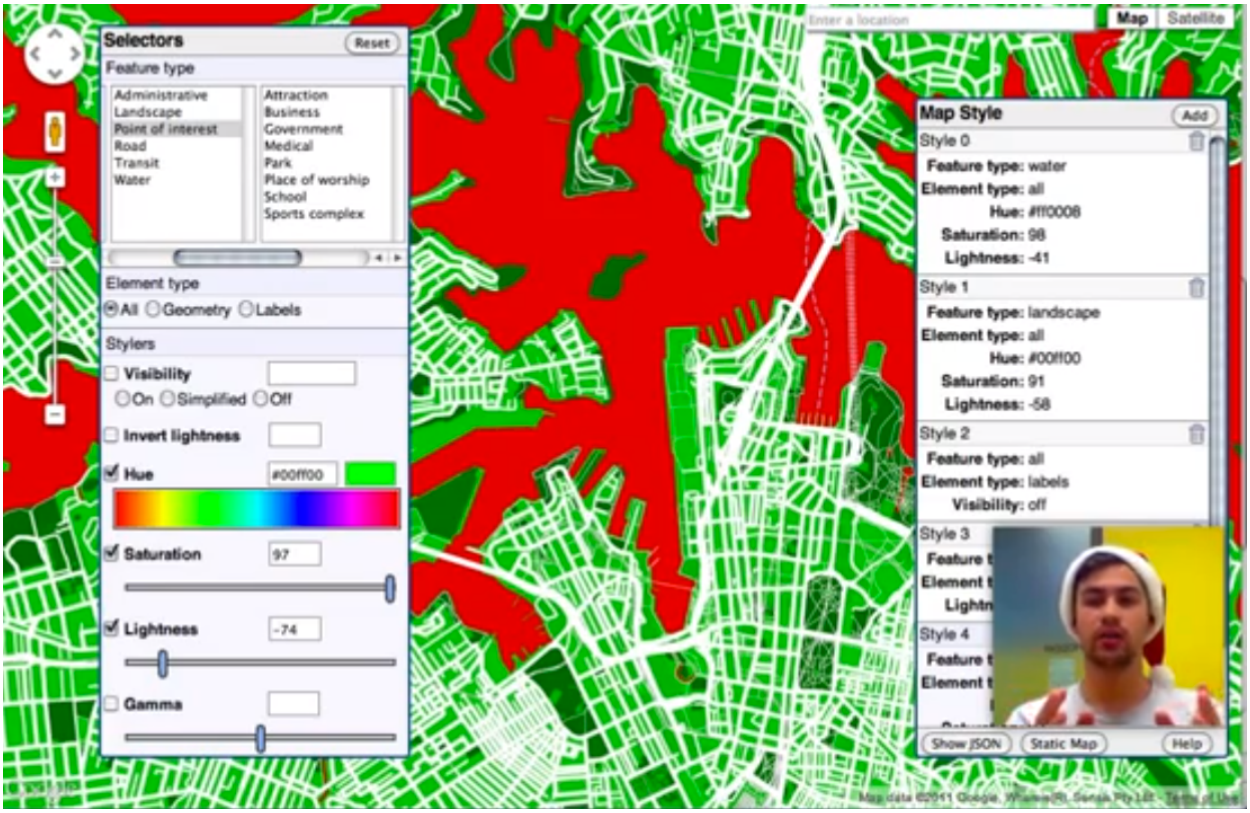


Figure 2: Google Maps Styled Maps Wizard link

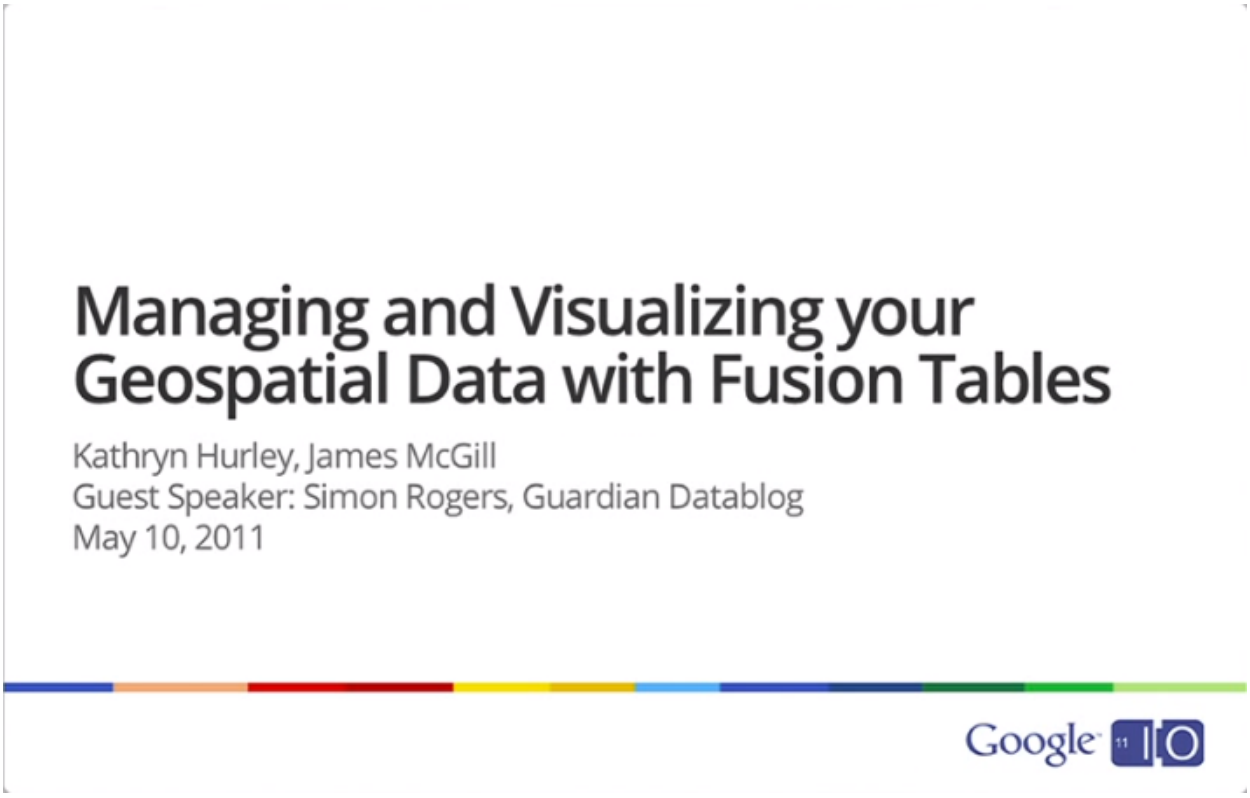


Figure 3: Google Fusion Tables Introduction Video link