The VegPlot Lookup Service uses the latest web service standards to communicate with local applications. This service is a set of functions that can be called from a remote client using two new Internet protocols, SOAP and WSDL, both standards within the World Wide Web Consortium (W3C). To facilitate easy integration, these web services are compatible with many web service toolkits. The VegPlot Lookup Services are published to a Universal Description and Discovery, and Integration (UDDI) registry so developers can quickly discover them.

SOAP

Soap brings together XML and HTTP by setting protocols for how Web services and their clients communicate with each other. Within a SOAP framework, a client sends a request in XML over HTTP to the Web service, which, in return, sends a response in XML. It does not matter what language is used to create the request or response as long as it is wrapped in XML. This flexibility allows developers to integrate components built in different programming languages.

The VegPlot Lookup Service provides a good example of how SOAP works.

Exchange **getPlotAccessionNumber**(String qureryToken);

To use this, the client invokes the "**getPlotAccessionNumber**" method, passes in a query Token (eg., latifolia) and receives a vector of accession numbers, latitude, longitude, and state
Corresponding to the vegetation Plots which have an attribute which contains the query token string.

WSDL

WSDL Web Services use another XML syntax (and W3C standard) to communicate with local applications. A WSDL document defines a GIS Web Service so the client knows what the service does. WSDL defines such things as which methods are available, what their parameters are, and the parameters' types. WSDL documents are most useful if used with a toolkit. Together, WSDL and SOAP allow any application connected to the Web to communicate with a Web Service.

UDDI

UDDI Web Services are published on the UDDI registry, a universal database of Web services. Developers can search any UDDI site to discover services on the registry, making UDDI a powerful resource for publishers and consumers alike. Discovery happens either through a Web interface (e.g., https://www-3.ibm.com/services/uddi/protect/registry.html) or SOAP calls.

The following Web sites provide overviews of SOAP, WSDL, and UDDI.
SOAP, WSDL, and
UDDI http://www-106.ibm.com/developerworks/webservices/
http://www.uddi.org/
http://dcb.sun.com/practices/webservices/

Web Service Toolkits Web service toolkits integrate disparate systems so they can work together. They are
often called frameworks because they are so comprehensive in what they

accomplish. A toolkit automatically reads the WSDL document and creates the client code for the service. Some toolkits create a context menu that displays all the possible methods and arguments. Without a toolkit, the developer needs to more fully understand the SOAP exchange between the server and client. The developer needs to construct SOAP requests and make an HTTP post to the server URL. With a toolkit, all you have to do is point the toolkit to the WSDL. The toolkit converts all parameters and requests into a SOAP message, which it sends to the service. It also takes care of converting the returned SOAP message into a usable object.

The toolkit currently used to test the VegPlot Lookup Service is:
The Mind Electric GLUE (http://www.themindelectric.com/)
Web service toolkit technology is new. For this reason, you may encounter an error when trying to use a toolkit with a particular Web service.

Example of Using a A Web service toolkit can decrease the amount of work developers have to do to
GIS Web Service with integrate services into their applications. Each toolkit is slightly different in how it
a Web Service Toolkit works, but the general methodologies and end results are the same. The example below
uses the GLUE toolkit (which uses Java) to call the VegPlot Lookup Web Service.
The definition of the service is found at the following URL:
http://vegbank.nceas.ucsb.edu:8004/vegbank/exchange.wsdl

The WSDL document defines the service as having one method.
Exchange **getPlotAccessionNumber**(String qureryToken);

GLUE comes with an executable called wsdl2java. When this is run and pointed toward the above URL it generates all the classes needed to call the GIS Web Service. In this case it creates three classes.

Class File	Description
IExchange.java	The interface that contains the method and is used to make the Web Service call.
ExchangeHelper.java	A helper class created by GLUE to instantiate the right objects.
Exchange.map	A mapping class that maps types from XML to Java.

Once these classes are created, they can be used in your program as follows:

```
// bind to web service whose WSDL is at the specified URL
String url = "http://vegbank.nceas.ucsb.edu:8004/vegbank/exchange.wsdl";
IExchange exchange = (IExchange) Registry.bind( url, IExchange.class );

// invoke the web service as if it was a local java object
Vector v  = exchange.getPlotAccessionNumber(queryString);
```

The objects are bound to the service; and the getPlotAccessionNumber request is made. The first two lines are set up and only need to be done once; the last can be repeated over and over.
To the program making the request, this looks like any other invocation method. This is because the underlying framework takes care of converting the call and parameters into

XML. Because SOAP is simply an XML protocol, it is fairly straightforward to use the SOAP protocol with an HTTP POST (but more complicated than using a toolkit).

Glue also comes with a nice tool (invoke) that can be used to invoke a web service from the command line. An example of this is:

./invoke http://vegbank.nceas.ucsb.edu:8004/vegbank/exchange.wsdl getPlotAccessionNumber latifolia

things to do:

check out the HTTP class in glue.