Flash, ColdFusion and ArcIMS: A New-Generation IMS-Map Portal

Prem RADHAKRISHNAN

Internet sites featuring a map service for displaying geographical information (spatial data) have become very popular. Sites such as the U. S. Geological Survey's "The National Map" (www. nationalmap. gov) typically receive millions of hits every day. The introduction of ArcIMS® by ESRI® has increased the popularity and demand for such sites. Specialized Internet map portals have become the most popular means of distributing spatial data to large number of consumers. This paper focuses on a new method of developing ArcIMS Web sites using Macromedia's® Flash® "remoting" and ColdFusion® server technologies. Flash is a popular technology used in Web design today because of its relative ease of use in building complex interactive Web sites. With the introduction of Flash remoting, it has become possible to build a Flash Web site that has dynamic content. ColdFusion is a popular scripting language that is used to communicate with databases such as Microsoft SQL Server® or Oracle® to retrieve information and can also be used to send HTTP requests and retrieve results. ArcXML is the protocol for communicating with the ArcIMS Spatial Server. ArcXML is an extended XML protocol.

INTRODUCTION

The 1990s saw the advancement of GIS from a little-known science to a well-known technology. Geographical Information Science_as theoreticians call it_ or Geographical Information Systems as programmers call it_has transcended all boundaries of what the pioneers of this science had envisioned. One of the first occurrences of geographical analysis is from an epidemiological study. Dr. John Snow used a map of plotted points of cholera occurrences to isolate a water pump that he thought was supplying contaminated water. When the tap was shut off, the outbreak of cholera was contained. Figure 1 shows part of Dr. Snow's original map.

Today GIS has changed a great deal but the basic principle remains the same: the use of visual tools to analyze data that has a spatial content for a specific purpose, in Dr. John Snow's case it was isolating a water supply that was causing cholera outbreaks. Today we use it to simulate contaminant transport through ground water, analyze population densities and distribution for urban planning, find optimal sites for landfills, help plan emergency procedures in case of a major outbreak of a disease, and more. A major development in the recent history of GIS has been the development of Internet spatial data portals, commonly known as "Internet map sites." The evolution of the Internet into a high-speed data transfer mechanism has facilitated the development of these portals, because they involve transfer of large amounts of data.

Since the content of these portals is dynamic and involves a great deal of user interaction, a reliable Internet is essential for such a portal to function smoothly. As the amount of data broadcasted through a spatial data portal increases, so does the complexity of the portal itself. We increasingly see spatial data portals dedicated to a particular purpose or a specific task, such as The Kansas Well Top Depth Stratigraphy viewer (http://drysdale.kgs.ku.edu/kgs/oilgas/strat_welltops/top_viewer.cfm), alongside more generic portals such as the "GIS Atlas for Indiana" (http://igs.indiana.edu/gisatlas), which encompasses different types of data. The focus of this paper is on data portals that specialize in delivering a highly customized interface for a very specific purpose.

Figure 1. A portion of Dr. John Snow's cholera map of 1854

Technology and Software

The heart of a data portal is the Internet. Without it, a data portal is a useless amenity parked on a computer, accessible only to those using that particular machine. The Internet makes it possible for a data portal to be accessed from anywhere on earth. The flesh and blood of a data portal is data, without which it is an empty shell having no purpose.

The software that I am focusing on in this paper is ArcIMS®, ColdFusion® and Flash®. The Environmental Scientific Research Institute (ESRI) manufactures ArcIMS, ColdFusion and Flash are products of Macromedia. ArcIMS (IMS = Internet Map Server) provides us the ability to create services that can be leveraged to publish large amounts of geographical data over the Web, as well as a set of tools to develop Web sites for this purpose. ColdFusion is a popular scripting language that is used widely today to build dynamic Websites that uses back-end databases to deliver up-to-date information. Flash is an

authoring tool that allows the developer to create anything from a simple animation to complex Web components or complete Web applications.

The conventional means of developing a spatial data portal is to use the ArcIMS Designer wizard to build the out-of-the-box viewer and then customize the viewer by modifying the HTML and JavaScript. The use of JavaScript to add interactivity to Web sites has been the prevalent method. There are a number of browsers in use today_Internet Explorer, Mozilla, and Safari, to name a few. The greatest setback in using JavaScript is that the object models for these browsers have subtle differences. This necessitates the developer to go the extra mile and write code for all the browsers separately. The developer has to make sure that his application is cross-browser compatible, which means that all the functionality of the application should work with any browser. Most of the time this is not an issue, because most simple tasks require the same code for all the browsers, the snags become more obvious as the developer tries to do more complex tasks with DHTML and interactivity. Flash, on the other hand, does not require this extra effort: the plug-in for the Flash player is available for all browsers and it only needs to be downloaded and installed once. Once this plug-in is installed, the Flash application written by the developer is deployed as a flash movie; it is recognized by the browser and creates no further problems. One of the major setbacks encountered by Flash developers in the past has been the static nature of the content. The introduction of Flash Remoting® has changed this: Flash Remoting allows the developer to communicate with Web services and request data which, in turn, can be formatted and made into the content for the flash application.

ARCIMS AND ARCXML

ArcIMS consists of both server and client components. In layman terms the functionality can be explained as a client requesting information from the server, and the server processing the request to supply this information back to the client. It is just a big Web service for generating dynamic maps. ArcIMS consists of four major components: the Manager, the Application server, the Spatial server, and ArcIMS connectors. A representation of the architecture of ArcIMS is shown in Figure 2. The Manager is used to manage ArcIMS; this component has depreciated with the release of ArcIMS 9.0 and will not be available in future releases, so this component will not be discussed any further. The Application server manages incoming requests and hands off the request to the appropriate Spatial server. The Application server also handles the response from the Spatial server and hands off the response to the Web server. ArcIMS supports a variety of Web servers, such as Internet Information Services (IIS) and Apache. The Web server acts as the intermediary between the client and ArcIMS. The spatial server takes most of the work load for ArcIMS when a request is received; the server processes the request accordingly and returns the response to the Application server. The last component of ArcIMS is the connector. The connector enables the Application server to interact with the Web server. Developers can send requests and receive responses from the connector. Different connectors are available, such as the ColdFusion connector, the ActiveX connector, Java connector, .NET link, and the ArcIMS Servlet connector. The data portal that is discussed in this paper will use the ArcIMS Servlet connector.

ArcXML is the communication protocol for ArcIMS. ArcXML is a specialized form of XML. To understand the following, it is important to know what an ArciMS service is. It can be an Image service, an ArcMap Image service, a Feature service, or a Metadata service. An ArcIMS service is created using a configuration file which is also written in ArcXML. Once a service is created, ArcXML requests can be send to the service through the Application server and appropriate responses can be generated by the Spatial server. The data portal features an Image service created using a configuration file. Part of this configuration file can be seen in Figure 3. The details of how to create a configuration file and how to create a service is beyond the scope of this paper. In simplistic terms, an ArcIMS service runs on the ArcIMS server and contains information about the service, such as the data layers that are part of the service, the location of the data layers, or the symbology that will be used to represent the data in those layers. In the snippet of the map configuration file represented in Figure 3, the details of the layer sections can be seen.

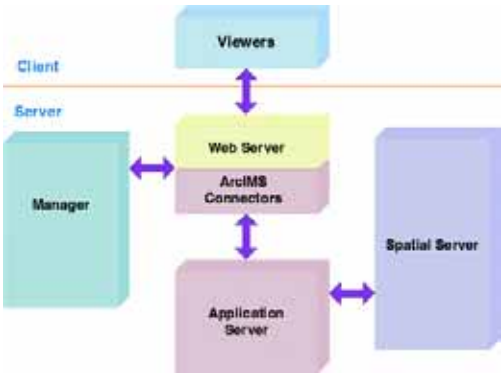

Figure 2: Architecture of ArcIMS (ArcIMS 9. 0 Documentation, ESRI 2004).

```
<?xml version="1. 0" encoding="UTF-8"?>

<ARCXML version="1. 1">
  <CONFIG>
    <ENVIRONMENT>
      <LOCALE country="US" language="en" variant="" />
      <UIFONT color="0,0,0" name="Arial" size="12" style="regular" />
      <SCREEN dpi="96" />
    </ENVIRONMENT>
    <MAP dynamic="true">
      <PROPERTIES>
                    <ENVELOPE minx="304259. 0777009068" miny="4171075. 150388279" maxx="790002.
9222990936" maxy="4636050. 849611722" name="Initial_Extent" />
                    <MAPUNITS units="meters" />
                    <BACKGROUND color="255,255,255" transcolor="255,255,255"/>
      </PROPERTIES>
                              <WORKSPACES>
                                        <SDEWORKSPACE name="ws-1" server="129. 79. 145. 2" instance="port:5152"
database="sde" user="sde" encrypted="true" password="" geoindexdir="C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\1\"
/>
                              </WORKSPACES>
                              <LAYER name="Sections" type="featureclass" id="sections/base/0" visible="false">
                                        <DATASET name="sde. sde. in_landsurvey_pl" type="polygon"
workspace="ws-1"/>
                                        <GROUPRENDERER>
                                                  <SIMPLERENDERER>
                                                            <SIMPLEPOLYGONSYMBOL antialiasing="false"
boundary="true" boundarycaptype="butt" boundarycolor="225,225,225" boundaryjointype="round"
boundarytransparency="1" boundarytype="solid" boundarywidth="1" fillcolor="254,255,255" fillinterval="6"
filltransparency="0" filltype="solid" overlap="true"/>
                                                  </SIMPLERENDERER>
                                                  <SCALEDEPENDENTRENDERER lower="1:100000">
                                                            <SIMPLELABELRENDERER field="PARCEL_ID">
                                                                      <TEXTSYMBOL font="Arial" fontstyle="regular"
fontsize="12" />
                                                            </SIMPLELABELRENDERER>
                                                  </SCALEDEPENDENTRENDERER>
                                        </GROUPRENDERER>
                              </LAYER>
.....
    </MAP>
  </CONFIG>
</ARCXML>
```

Figure 3. Part of the map configuration file used for the data portal.

Important ArcXML Tags Used

The GET_SERVICE_INFO tag is used initially to retrieve important information from the ArcIMS service, such as the layers in the ArcIMS Service, the order in which they are displayed, the full geographical extent of all the datasets in the service, and the display names of the layers. The retrieved information is stored in local Flash variables in the application.

The GET_IMAGE tag is used every time a new image is generated. For example, when the user zooms into a particular area in the map, the GET_IMAGE tag is used with the new extent to generate a new image. The GET_IMAGE tag needs as input the extent of the image to be generated, the size of the image generated, the layers to be displayed, and information about any dynamic layers that needs to be added to the image. If no parameters are supplied, then the default image of the service, as specified by the configuration file, is created.

The GET_FEATURES tag is used when attribute information is needed. For example, when a user tries to identify a feature that is displayed on the map, the GET_FEATURES tag is used to retrieve information about that feature. The GET_FEATURES tag can be used only on one data layer at a time, and a QUERY or SPATIALQUERY element that defines what features needs to be extracted is required.

COLDFUSION AND FLASH REMOTING

ColdFusion is application server software and also a scripting language. In this paper, when we talk about ColdFusion, we refer to it as a scripting language unless specified otherwise. ColdFusion can be used to communicate with the ArcIMS Servlet connector, to send requests, and to receive responses from the connector. The main ColdFusion component designed was the one that communicates with the ArcIMS Servlet connector itself. The component is shown in Figure 4. Other components have functions that deal with specific kinds of requests: one function for retrieving images, another for retrieving attribute information, and another for geometry information.

```
<cfcomponent displayName="XMLRequest">
        <cffunction name="XMLRequest" access="public" returnType="any" output="false">
                <cfargument name="xmlreqstring" type="any" required="yes">
                <cfargument name="xmlreqtype" type="string" required="no">
                <!--- XMLRequest body --->
                <cfif IsDefined("xmlreqtype") >
                        <cfset posturl = "http://127. 0. 0. 1/servlet/com. esri. esrimap.
Esrimap?ServiceName=PDMS&ClientVersion=9. 0&CustomService="& #xmlreqtype#>
                <cfelse>
                        <cfset posturl = "http://127. 0. 0. 1/servlet/com. esri. esrimap.
Esrimap?ServiceName=PDMS&ClientVersion=4. 0. 1">
                </cfif>
                <cfhttp method="post" url=#posturl#>
                        <cfhttpparam encoded="no" type="body" name="ArcXMLRequest"
value="#xmlreqstring#">
                </cfhttp>

                <cfreturn cfhttp. FileContent>
        </cffunction>
</cfcomponent>
```

Figure 4. ColdFusion component that communicates with the ArcIMS Servlet Connector.

The most important ColdFusion tag in the component in Figure 4 is the <cfhttp> tag. The <cfhttp> tag is used to pass variables to a ColdFusion page or a CGI program without the target page or program actually appearing. The target page or program processes the information and then returns the results (data) to the calling page. The calling page can further process the results and direct the results elsewhere, or

initiate the next step in the processing. The <cfhttp> has two methods: GET and POST, which are similar to the GET and POST methods of an HTML <form>. When using the GET method, data is passed only to the target page and no data is returned, whereas when using the POST method, data can be passed to the target page and data can be returned after being processed by the target page or program. In this particular component, the <cfhttp> tag is used to pass the ArcXML request from the calling function to the ArcIMS Servlet Connector and then retrieve the ArcXML response and pass it back to the calling function.

A number of ColdFusion functions were designed to handle the different kinds of requests to be sent to the ArcIMS Servlet connector and to process the response returned. One very important fact about an ArcIMS service must be mentioned here: the ArcIMS service is stateless. It does not remember anything other than what is stated in the configuration file. For example, if a user turns off a display layer on the client, it is left up to the developer to handle the modification of the ArcXML request that is to be sent to the service, and also to remember to keep that layer turned off in all future requests until the user turns it back on. When using the ArcIMS Designer to create an out-of-the-box HTML client, JavaScript code is automatically generated to create variables to store all the information required on the client side. In the HTML client, the JavaScript is the means by which the client communicates with the ArcIMS Servlet connector. The ColdFusion component shown in Figure 4 does the job in this particular application. Another important fact to be mentioned here is that this application makes use of another external database that is not a spatial database. ColdFusion can communicate with databases such as SQL Server®, Oracle®, and Microsoft Access® to retrieve information from the tables. It was considered beneficial to populate some of the data required for this application in a SQL Server® database and use ColdFusion to retrieve that information when needed.

The ColdFusion components take care of communication with the ArcIMS Application server via the ArcIMS Servlet connector. To display the information that is gathered by these components, an interface is still needed. This is where Flash comes into play. An important fact to mention here is that Flash is can be used with any scripting language, ColdFusion was chosen because it is a fast, easy environment for building dynamic applications. Using Flash Remoting objects and ActionScript it is possible for a flash movie (swf file) to communicate with server side services, such as a ColdFusion component. An instance of the server side object (ColdFusion component) is created and a call is made to the object, and two functions are set up to handle the response from the object. One function handles the response if it is returned correctly, and another function handles the faults generated if the object malfunctions. Figure 5 shows a snippet of Flash ActionScript code that uses Flash Remoting Application Programming Interface (API) to initiate an instance of a ColdFusion component (as a Service object), call a function in the component (as a PendingCall), and set up the functions to handle the response or fault (using the RelayResponder). The constructor for a service can be specified as:

Service (gatewayURI, logger, serviceName, conn, resp)

The gatewayURL is a virtual directory For ColdFusion, this directory is automatically set up when ColdFusion is installed. The logger must be a new log object to which the connection can be logged. The serviceName is a string specifying the service to access, which in this case is the ColdFusion component. All component paths are relative to the wwwroot directory in the Inetpub folder. These two folders are always present when IIS is installed on a machine. The connection parameter can be used to specify an existing connection, so that the application can share one connection. The gatewayURL and the connection parameters are mutually exclusive, the connection parameter always taking precedence over the gatewayURL. Sharing can be made clear by the following example (Flash MX Documentation, Macromedia, 2004).

```
var cust:Service = new Service( "http://localhost:8300/flashservices/gateway", null, "customerData", null, custDataResp );

var employee:Service = new Service( null, null, "employeeData", cust. connection, empDataResp );
```

The first Service (cust) establishes the gateway connection using the URL, and the second service (employee) specifies a null URL and instead uses the connection that was already established by the first one (cust). The resp parameter is used to set up a responder for the service to handle the callbacks for service function calls, and relays the result or fault to the specified objects and methods. This parameter can be specified as null, and separate responders can be set up for each service function call, as shown in the example in Figure 5.

Figure 5 shows initiation of a Service for the component "mapService.cfc" and a call to the function getMap in the Service. The function requires three parameters: the extent of the map to be retrieved, a list of layers to be retrieved, and an acetate layer parameter, if necessary. These parameters are set up as variables when the application is initialized and then are modified when the user interacts with the application. The two functions, GetMap_Result and GetMap_Fault, handle the result and the fault generated in the callback to the function. The GetMap_Result function loads the image generated, so that it is displayed in the portal. The GetMap_Fault function is used during debugging to display error messages that could be generated.

More could be done in the GetMap_Fault function, such as displaying an error message to the user and shutting down the application. An important thing to note is that the statelessness of ArcIMS is handled by Flash ActionScript. The only place where everything is tracked is in the Flash ActionScript. The major variables that must be stored are the current extent, the previous extent, the current layers being displayed, and the current active layer. The user should not have to specify all these variables every time a new image is requested. The only time the user should have to do anything is to change something in the display, such as turning off a layer.

```
//set up remoting services
GATEWAY_URL = "http://"+HOST+"/flashservices/gateway/";
//Action Script 2. 0 API to initialize a service which is a cfc  (coldfusion component)
var mapService:Service = new Service("http://127. 0. 0. 1/flashservices/gateway",
                                     new Log(),
                                     "flashims. components. mapService",
                                     null,
                                     null);
var GetMap:PendingCall = mapService. GetMap({ Extent:_global. CurrentExtent, Size:_global. imgSize,
Layers:_global. Layers, Acetate:_global. Acetate });
GetMap. responder = new RelayResponder(this, "GetMap_Result", "GetMap_Fault")

GetMap_Result = function(resultMap:ResultEvent) {
        //update the loading text
        _root. loaderProgress. label = "Receiving Map";
        _root. loaderProgress. indeterminate = true;
        //load the image
        //instanceName. contentPath
        _global. currentURL = resultMap. result["url"];
        _root. imageLoader. contentPath = _global. currentURL;
        _root. imageLoader. _x = 0
        _root. imageLoader. _y = 22
        //set the map extent
        _global. CurrentExtent = resultMap. result["Extent"];
        _global. CurrentExtent = numericArray(_global. CurrentExtent);
        if (_global. PreviousExtent["minx"] == undefined){
                //trace("First Time Prev Extent ");
                _global. PreviousExtent = new Array();
                _global. PreviousExtent["minx"] = _global. CurrentExtent["minx"];
                _global. PreviousExtent["miny"] = _global. CurrentExtent["miny"];
                _global. PreviousExtent["maxx"] = _global. CurrentExtent["maxx"];
                _global. PreviousExtent["maxy"] = _global. CurrentExtent["maxy"];
        }
        if (_global. InitialExtent["minx"] == undefined){
                //trace("First Time Initial Extent");
                _global. InitialExtent = new Array();
                _global. InitialExtent["minx"] = _global. CurrentExtent["minx"];
                _global. InitialExtent["miny"] = _global. CurrentExtent["miny"];
                _global. InitialExtent["maxx"] = _global. CurrentExtent["maxx"];
                _global. InitialExtent["maxy"] = _global. CurrentExtent["maxy"];
        }
        //Calculate the scale
        _global. mapScale = ((_global. CurrentExtent["maxx"] - _global. CurrentExtent["minx"])/((Stage. width-
240)/96))/0. 0254;
        trace(_global. mapScale);
        _root. scaleDisplay. text = String(formatNr(Math. round(_global. mapScale), ","));
        _root. scaleDisplay. _x = (Stage. width - 350);
        _root. scaleDisplay. _y = 24;

};

GetMap_Fault = function(fault:FaultEvent) {
        trace("GetMap_Fault:"+fault. fault. faultstring);
};
```

Figure 5. Sample call to a function in a component and handling the response.

THE APPLICATION

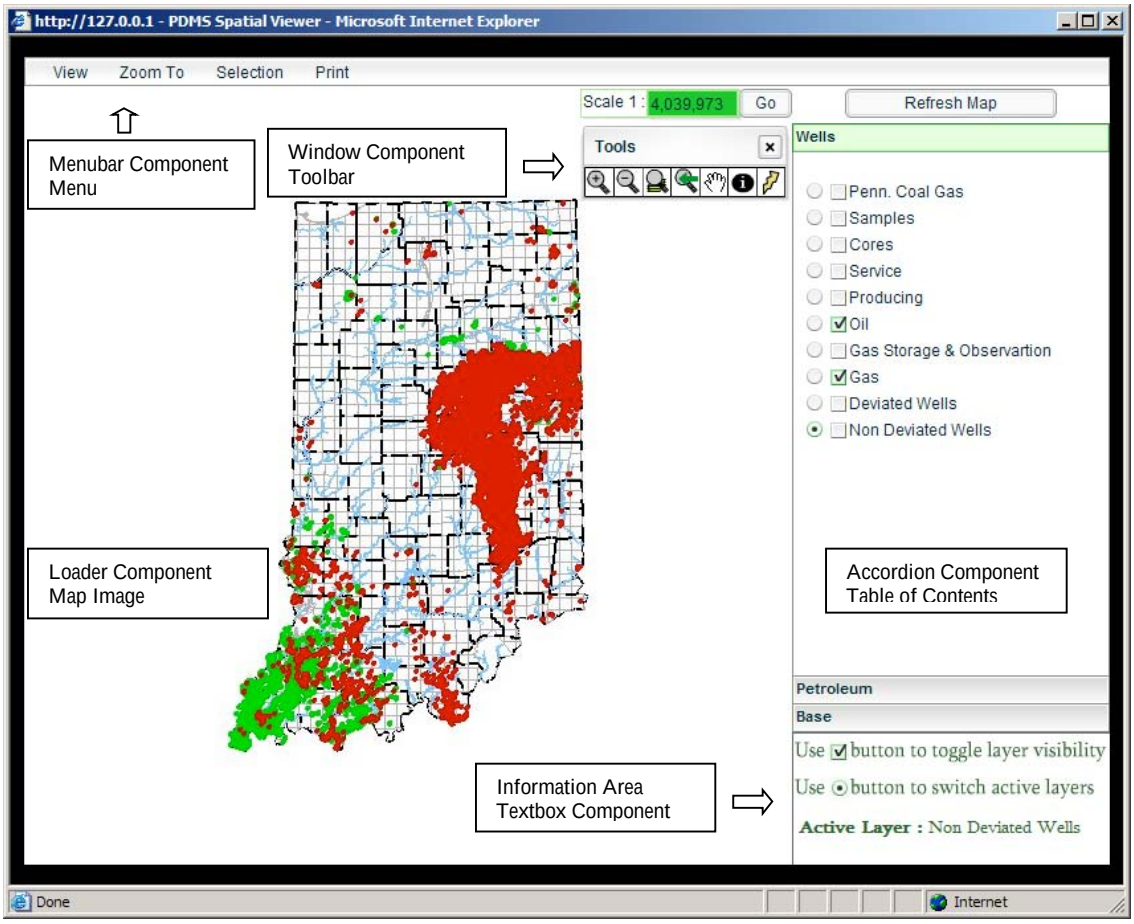

Figure 6: Data portal showing the major sections and the Flash components used.

Figure 6 shows the PDMS Spatial viewer that was developed using the techniques described in this paper. This is a highly customized portal with many custom functions and intractability options. The figure also shows some of the Flash components used for the different parts of the portal. The portal can be divided into a few main areas:

- Map Image Area;
- Table of Contents;
- Toolbar;
- Scale;
- Information;
- Menu.

Map Image Area

The map image area is where most of the interaction between the user and the portal takes place. This was implemented using the Loader component in Flash. ArcIMS was set up to generate all the images in jpeg format since the Loader component can load only jpeg format out of all the format options available in ArcIMS. Whenever a new image is generated from a call to the ArcIMS service, the image is

loaded into the Loader component and displayed to the user. The result of the user interaction in the map image area is closely related to what tool the user has selected. For example, when the user wants to zoom to a particular area in the map after the initial map is displayed, the user selects the zoom-in tool from the toolbar and drags a box on the image display. The result is a call to the ColdFusion function that generates a call to the ArcIMS Servlet connector. The ColdFusion function generates an ArcXML request that includes a GET_IMAGE tag with the new zoomed-in extent and retrieves a new image whose path is then passed back to the Flash application and loaded into the Loader component. The application always keeps track of the extent of the map that is currently being displayed and also the extent of the map that was previously displayed, using local variables. When the portal is initialized, the default extent displayed is the full extent of the map service.

Table of Contents

The table of contents displays the names of the data layers in the map service and provides the user with the option of displaying (or not displaying) the layer on the image. When the application is initialized and the portal is being loaded, the information about the layers in the service is retrieved using a function in the ColdFusion component that retrieves the information and passes it back to the Flash application. This information is then used to construct the table of contents. The ids (alphanumeric identification keys) of the layers in the configuration file are set up in a way that allows the application group the layers. The Accordion component was used to display the different groups of layers in the service. There is a checkbox and a radio button to the left of each layer name. If the checkbox is checked, the layer will be displayed; the user can use this checkbox to turn layers on and off. The radio button is used to activate a layer. If the user wants to identify a feature on the layer, he must first activate the layer that he wants to identify. Since the GET_FEATURES tag can be used only on one layer at a time, the active layer is one that is used whenever attribute information needs to be retrieved. Information about each layer is stored in a local variable, so that every time an image needs to be requested, the display state of each layer is known. Whenever the user interacts with the portal and changes the display state of a layer, the variable associated with the layer is changed accordingly. These variables are supplied to the ColdFusion component function when an image must be generated and the GET_IMAGE request is formed accordingly.

Toolbar

The toolbar provides the user with the various tools and buttons that can be used in the image area in the portal. The toolbar is a Flash window component that is created when the application is initialized. All the images that are used to symbolize the various tools are created before hand and loaded into the window when it is created. Each image has an associated function which is called when the image is clicked.

The zoom-in tool (symbolized by a magnifying glass with a '+' sign) is used to focus in to a smaller area on the map. Select the tool and draw a box on the image to zoom into that area. The zoom-in tool triggers a call to a function in the ColdFusion component that generates an ArcXML request to the ArcIMS service (in which the main tag is GET_IMAGE) to generate a new image for the new extent. The zoom-out tool (symbolized by the magnifying glass with a '-' sign) is used to expand the area that is being displayed. Click on the map once to recenter the map at that point and zoom out by a predefined fixed percent. This tool also triggers a call to the function that generates a new image. The third tool is the zoom-to-full-extent button can be used to reset the map to the full extent that is defined in the map configuration file. The button symbolized by the magnifying glass with a block arrow to the left is used to go to the previous extent of the map display. The tool symbolized by the open hand is the pan tool; and can be used to pan in any direction from the extent that is displayed. To use this tool, the user must click and drag the image in the required direction. All these functions trigger calls to the same function as the zoom-in tool to generate a new image based on the new extent, as specified by the user interaction.

The black circled 'i' is the identify tool; this tool can be used to identify features that are displayed on the map after setting that layer to be the active layer. This tool triggers a call to a ColdFusion function that generates an ArcXML request with GET_FEATURES as the main tag to retrieve attribute information about the features that are to be identified. The last tool in this application is the hyperlink tool; this tool is applicable only to layers that are predefined as hyperlinked. A hyperlinked layer is one in which each layer is linked a specific URL, which is usually determined by one of the attributes of the feature. Clicking on a feature using the hyperlink tool opens a new browser window and loads the page that is the hyperlink of that particular feature. In this particular application, if multiple features at the clicked location are hyperlinked, the user is provided with the option of selecting which he wants to open. Behind the scenes, a GET_FEATURES request is sent to the ArcIMS service to retrieve a particular attribute of the feature or features. If there is only one feature, the URL is generated using the attribute, and a new browser is opened with the URL as the target. If there are multiple features, an HTML page is generated where the user can select. Other tools can be added to the toolbar depending on the scope of the application. For example, a measuring tool can be added to measure distances on the map, or a drawing tool can be implemented for users to draw their own figures on to the image being displayed.

Scale

The scale section serves two purposes: to show the scale of the map being displayed and to make it possible for the user to zoom to any scale needed. When a user enters a scale, a new image is generated with the extent calculated using the scale that is entered.

Information

The information area displays tips and other pertinent information to the user. The active layer is always displayed in the information area.

Menu

The menu provides the user with additional functions that are not usually part of the HTML viewer. Implementing functionality like that of a menu bar is now possible with DHTML objects and JavaScript, but the amount of programming required to implement such functionality is complex and requires both considerable time and effort. Since JavaScript for different browsers has subtle differences, it becomes necessary to detect the version and type of browser and implement browser-compatible programming. This additional effort to ensure that the application is cross-browser compatible is a Web programmer's nightmare. The complete application menu is shown in Figure 6.

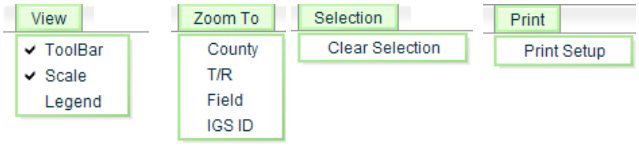

Figure 7: The different menu bar items in the data portal.

There are four items in the menu bar: View, Zoom To, Selection, Print. The Zoom-To menu is application-specific and was designed specifically for the Indiana Geological Survey's Petroleum Database Management System (PDMS) Spatial Viewer. Depending on the data, this same menu can be tailored for other portals. The View menu is used to hide or display the Toolbar, Scale, or Legend. This can be useful when the space is limited and the display of certain components is not necessary. The legend is required only when the user needs to know the symbology of the features. When the legend is asked to be displayed, a GET_IMAGE tag is used in the ArcXML request, but instead of asking for an image of the different layers, the request is made for the image of the legend. This image is then loaded into the display.

The Zoom-To menu is used to minimize the time and effort required by the user to zoom into an area of interest. The Zoom-To menu features four main layers that are displayed in this portal: County, Township/Range, Fields, and Wells. When the user clicks on the "County" option, a list of counties is provided through use of a drop-down menu, and when the user selects a county, the map automatically zooms into that area. Behind the scenes, the list of counties comes from a table stored in a database, and the list loaded at the time the application is initialized into a drop-down listbox. When the user selects a county, an ArcXML request that uses the GET_FEATURES tag is used to retrieve the bounding coordinates of the county, and then a GET_IMAGE tag is used to request a new image for that envelope. The other layers function in a similar manner, the GET_FEATURES request focusing on the particular layer of interest to retrieve the bounding coordinates.

The Selection menu has only one function to clear any features that are shown as selected. This menu can also be used to select features using predefined queries.

The Print menu is used to set up the current map image for printing.

SUMMARY

A portal that is designed using this method can be very sophisticated. The power of Flash components and ActionScript can be used to create a very powerful spatial data portal. One must take into consideration that such an interface usually requires more programming and more time to design than a conventional HTML viewer. The ESRI ArcIMS Designer can create a standard HTML viewer in a matter of minutes, and this viewer can be customized in a few days. The Flash data portal will require more planning, and the project could take weeks or sometimes months, depending on the complexity of the portal and the complexity of the data being displayed. But the rewards of such an effort can be extremely gratifying, because the resulting portal is extremely useful and very convenient to use.

ACKNOWLEDGEMENTS

I would like to thank Charly Zuppann (Petroleum Geologist), John Rupp (Assistant Director of Research), and Dr. John Steinmetz (State Geologist) of the Indiana Geological Survey for the support and guidance provided in the designing of this portal. I would also like to thank Jeremy Bartley (Assistant GIS Coordinator) of the Kansas Geological Survey for his valuable guidance and tips provided during this project.

REFERENCES

1. ArcIMS 9.0 Documentation, ESRI Publications, 2004.
2. ArXML Programmers Reference, ESRI Publications, 2004.
3. ColdFusion Users Guide, Macromedia, 2005.
4. Flash Documentation, Macromedia, 2004.

AUTHOR'S INFORMATION

Prem Radhakrishnan
Sr. GIS/Database Analyst/Programmer
pradhakr@indiana.edu
Indiana Geological Survey