



Autodesk
University
2007

Creating Commercial Web 2.0 Applications with Autodesk MapGuide® Studio and DM Solutions Group's Fusion

DM Solutions Group, Paul Spencer
Autodesk, Geoff Zeiss
GSC Corp, Nicole Jung

GS318-3L This class is aimed at developers who are interested in the technical aspects of developing commercial software on a Web 2.0 open-source geospatial platform. The class is based on the next-generation web platform MapGuide Open Source and the web development tools in Autodesk MapGuide Studio and DM Solutions Group's Fusion technology. The class will introduce development of commercial applications on an open-source platform, the technical enablers underlying Web 2.0 including AJAX, and creating dynamic Web 2.0 mapping applications. You'll learn how to use the web development tool in Autodesk MapGuide Studio to develop and deploy web applications and DM Solutions Group's Fusion application development framework to develop interactive web mapping applications.

About the Speaker:

Paul joined DM Solutions Group in 1998. He started as the lead developer for applications development, working on several open-source Web mapping projects and software development frameworks. Paul now serves as the company's chief technology officer and has been active in the MapGuide Open Source community, leading the development of a web-based management tool and, more recently, a new application development framework for MapGuide Open Source called Fusion. He has a Master's degree in Software Engineering.

pspencer@dmsolutions.ca



Autodesk
University
2007



Creating Commercial Web 2.0 Applications with Autodesk MapGuide® Studio and DM Solutions Group's Fusion

Overview

The generic components that make up a DMSG Fusion application are:

- MapGuide Server
- MapGuide Web Tier
- MapGuide Resource Documents
- DMSG Fusion
- Jx

MapGuide Server

Every MapGuide installation contains at least one MapGuide Server and one MapGuide Web Tier. The MapGuide Server holds all the data and performs all the basic operations of creating a map image from that data.

MapGuide Web Tier

The MapGuide Web Tier is the link between the MapGuide Server and a web mapping application. The Web Tier is integrated with a Web Server and provides two different mechanisms for web applications to communicate with the MapGuide Server:

- MapAgent, a Web Server extension that provides convenient access to basic functions of the MapGuide Server, and
- An Application Programming Interface (API) that developers can use to work with the MapGuide Server to create custom functionality for their web applications

MapGuide Resource Documents

A MapGuide Server is a repository for various types of Resource documents that are used by the server primarily to generate map images for web applications. It is beyond the scope of this guide to describe all types of Resource documents and their purpose. However, there are 4 Resource document types that contribute directly or indirectly to the creation of a web mapping application with Fusion.

A Feature Source is a Resource document that provides the MapGuide Server with the configuration information needed to access GIS data.

A Layer Definition is a Resource document that provides the cartographic representation of a Feature Source to the MapGuide Server.

A Map Definition is a Resource document that defines the appearance of a map in a web mapping application.



An Application Definition is a Resource document that provides instructions to Fusion on how to build a web mapping application. It determines what functional components (called Widgets) will be present in a Fusion application, certain aspects of the presentation of the Widgets, and optional configuration of the functionality of each Widget.

The first three Resource documents (Feature Source, Layer Definition and Map Definition) all combine to produce an image of a map in the final application. It is important that you know they exist because you will have to create them before you can build your application. However, there is not much more that you need to know about them in order to create a Fusion application. The Application Definition Resource directly impacts on the appearance and functionality of your application and is discussed further in the following section.

Feature Source, Layer Definition and Map Definition Resources can be created using with MapGuide Enterprise Studio or MapGuide Open Source Web Studio. A discussion of using these tools with MapGuide is beyond the scope of this document.

Application Definition

An Application Definition Resource document must be created manually at this time. It is an XML file that can be created by any text editor or XML editor application.

The root element of the document is an <ApplicationDefinition> tag. There is exactly one ApplicationDefinition tag allowed in an Application Definition Resource document.

Within the ApplicationDefinition tag are the following:

Title

There is exactly one Title tag in the ApplicationDefinition. The Title of the application is displayed in the title bar of the client's browser

MapSet

There is exactly one MapSet in the ApplicationDefinition. The MapSet contains one or more MapGroup tags that define the maps that are available to the application.

MapGroup

There may be one or more MapGroup tags in a MapSet. A MapGroup contains one or more Map tags.

Map

There may be one or Map tags in a MapGroup. A Map tag supplies the Resource Id of a MapDefinition that Fusion will use when this Map is loaded as part of an application

WidgetSet

There may be one or more WidgetSet tags in an ApplicationDefinition. Each WidgetSet defines containers and widgets that operate on a single map. Each WidgetSet has exactly one



MapWidget and represents one Map in the final application. It is possible to have more than one map in your application by defining more than one WidgetSet.

Container

A Container is a grouping of one or more widgets into a toolbar or menu. The use of Containers is not mandatory. They are available because the use of Toolbars and Menus in applications is very common. It is entirely possible, and in some cases desirable, to build an application without the use of Containers.

MapWidget

Each WidgetSet requires exactly one MapWidget. The MapWidget provides an API to other widgets to be able to manipulate the map, for instance to zoom or pan the map.

Widget

There are one or more Widget tags in a WidgetSet. Each Widget defines a specific type of functionality to be included in the resulting application.

DMSG Fusion

DMSG Fusion is the MapGuide component that creates a web mapping application by combining an Application Definition and a Map Definition with an HTML web page (called a Template).

A DMSG Fusion application is started when the user loads a Fusion-enabled HTML web page into their browser. With some minimal required JavaScript in the HTML web page, DMSG Fusion is able to retrieve an Application Definition, determine what functionality is required in the application, and create the necessary components on the fly.

HTML Templates

An HTML template can be any HTML file that is used with DMSG Fusion to create an application. From DMSG Fusion's point of view, an HTML page is a template because it provides places to put Widgets and Containers. An HTML file can be created in any web-authoring environment such as Dreamweaver or even a simple text editor.

In general, DMSG Fusion widgets are contained by an HTML element present in the template without interfering with any other HTML in the template. When designing a DMSG Fusion application, it is then easy to use HTML to create blocks as placeholders for functionality and concentrate on the appearance of the web page. Then when the DMSG Fusion JavaScript is added to the template, the Widgets automatically appear where you want them to without changing the appearance of the rest of the web page.



Jx

Jx is a JavaScript library that facilitates the creation of applications in a web environment. It contains generic components that are useful in any web application, including buttons, toolbars, menus, trees, grids, dialogs, panels, and layout management.

Jx is included in all DMSG Fusion applications because DMSG Fusion uses several components of Jx to represent some widgets and containers. DMSG Fusion primarily makes use of Buttons, Toolbars and Menus. Many Fusion applications will also take advantage of other Jx components, such as dialogs, panels and layout management, to create more appealing applications.

Definition of Widget

In DMSG Fusion, all functionality is encapsulated as a widget.

widget /wɪdʒɪt/

(noun informal) In computing, a component of a user interface that operates in a particular way.

In DMSG Fusion, we use the term Widget to describe discrete, independent components that have specific functionality in an application.

Discrete: widgets are intended to be individually separate and distinct. Each Widget has specific functionality that it provides to an application.

Independent: widgets are designed to operate independently from all other widgets present in the same application, including other widgets of the same type. The presence or absence of any widget should not change the behavior of other widgets in an application.

What is a Widget?

A widget consists of several pieces:

- an XML block in an ApplicationDefinition
- a JavaScript file that implements the functionality of the widget
- a visual representation of the widget in the application
- supporting server-side code that uses the MapGuide Web Tier API to access more advanced functionality

XML Tag

The XML tag that defines a widget in an ApplicationDefinition lives within a <WidgetSet>. It has the following basic structure:

<Name>



The Name element is used to determine where the widget will appear in the application. A widget can be placed in two ways.

The first way is for the widget to be placed directly in the HTML page. This happens when an HTML tag has an ID property that has the same value as the Name element. Fusion automatically looks for HTML tags that match the Name element and places the widgets inside the HTML tag.

The second way is for the widget to be placed inside a Container that is defined in the ApplicationDefinition. In this case, there is no HTML tag with a matching ID. Rather, the Container's XML block references the widget by Name and the Container has its own location in the HTML page.

<Type>

The Type element is used to determine which widget will appear in the application. This is used to determine which JavaScript file to load for the widget and what JavaScript class to instantiate to create the widget.

<StatusText>

The StatusText element is used to provide additional information about a widget when the user interacts with it. The StatusText value is put into the status bar of the Web Browser.

<Extension>

The Extension element is used to record additional configuration information for widgets. There is no preset format to the contents of the Extension tag. The documentation for each widget will describe what, if any, additional configuration can be included in the Extension element.

In addition to the basic tag structure for all widgets, those that represent themselves as Buttons or Menu Items have some additional tags that can appear after the Extension element.

<ImageUrl>

The ImageUrl element is used to identify an icon that is placed in a button or menu item. This is optional and can be left empty if only a label is desired. The CSS skin that is being used for the application determines the size of the icon. Typically, it will be 16 x 16 pixels.

<ImageClass>

The ImageClass element is used to associate a CSS class name with the icon embedded in a button or menu item. This element is not normally used. However, if you are using 24 bit PNG images with alpha blending and the images do not display correctly in Internet Explorer 6, you can specify "png24" in this field to make the image display correctly.

<Tooltip>

The Tooltip element is used to specify a tooltip that is displayed when the user hovers their mouse over a button or menu item.

<Label>



The Label element is used to specify a label that will be displayed in the button or menu item.

<Disabled>

The Disabled element may have a value of “true” or “false” and defaults to “false” if any other value is entered. If this is set to “true” then the widget is initially disabled when the application starts and must be programmatically enabled by the application developer.

JavaScript File

The functionality of each widget is implemented in a JavaScript file that lives in the “widgets” folder of a DMSG Fusion installation. The name of the JavaScript file must match (case sensitive) the <Type> element in the widget’s XML block described above. The JavaScript implementation of the widget allows it to manipulate the HTML web page when the widget is created so that the widget can create additional HTML elements dynamically to represent the widget.

CSS Styling

The physical appearance of a widget in an application is controlled by CSS. A Jx skin CSS file controls the appearance of widgets that represent themselves as buttons or menu items. Other widgets may have their own CSS file included in a subfolder of the “widgets” folder in a Fusion installation. These CSS files can be overridden by any user supplied styles so that an application designer can have full control over the look and feel of an application. See individual widget documentation for details on what CSS classes can be defined for each widget.

Server Side Functionality

Most widgets are implemented purely in JavaScript. However, some widgets implement functionality that requires direct access to the MapGuide Server. In these cases, the additional functionality is implemented in PHP on top of the MapGuide Web Tier API. If a widget users server side functionality, this information will be included in the widget documentation.

Types of Widgets

There are two basic categories of widgets, Action and Information. Most widgets fall exclusively into one of the two categories. There are several widgets that are in both.

Action Widgets

Action widgets are those that cause something to happen to the map. They include basic navigation widgets such as panning and zooming, as well as more advanced widgets that change the map such as creating a buffer.

Action widgets are typically represented as buttons and menu items in a user interface. Usually only one Action widget is active at any time. Activating a different Action widget causes the currently active widget to be deactivated. Some action widgets, such as Buffer and



SelectWithin, require more complex interaction with the end use and provide panels that are presented to the user when the widget is activated.

Information Widgets

Information widgets are those that display some information about the current state of the map to the user. Typically, these widgets are not represented as buttons or menu items but rather as some sort of HTML textual output that is dynamically updated. This includes widgets such as CursorPosition, ViewArea.

Combination Widgets

There are several widgets that provide information and perform some action on the map. The Legend and EditableScale widgets are examples of this.

How do I get more widgets?

There is a set of standard widgets provided with DMSG Fusion that implement most of the functionality required by basic web mapping applications. If your application needs functionality beyond that provided by the basic widget set, you have the following options:

InvokeURL and Custom Tasks

The InvokeURL widget is provided to allow developers to easily extend DMSG Fusion applications with custom functionality. The InvokeURL widget embeds a developer's web page inside a DMSG Fusion application and allows both client and server side functionality to be implemented.

Fusion API

In many cases, developers can implement additional functionality in JavaScript just by using the DMSG Fusion API. This can be useful for rapid development, for customizing output of query results, or for accessing external functionality such as client databases that are not directly linked to the spatial application.

Write Your Own Widget

For functionality that needs to be more tightly integrated with DMSG Fusion, or highly reusable beyond a single application, new widgets are created by copying an existing widget and modifying its functionality.

3rd Party Widgets

In some cases, 3rd party developers will create widgets or sets of widgets that they sell or otherwise provide for use. The 3rd party developer should provide installation instructions for their widgets, after which the widget should work like any other DMSG Fusion widget.