2 Chapter

What's New in MapXtreme?

This release of MapXtreme Java offers many new features and enhancements to better help you design and build the web mapping application that best meets your needs.

- Java 2 Support
- MapXtremeServlet
- Installation Conveniences
- Insert/Update/Delete for JDBC DataProviders
- Support for Oracle
- QueryBuilder
- Connection Pooling
- Map Definition Management
- Theme Management
- Renditions
- Labeling Enhancements
- Selectable Layer Objects
- New JavaBeans
- Localization
- Printing/Plotting Support
- Raster and Grid Support
- Shape File Support
- MapToolkit (Servlet Library)
- Sample Applications
- Changes Between MapXtreme Java 2.0.x to 3.0

Java 2 Support

This release of MapXtreme Java requires the Java 2 Platform Standard Edition. This allows MapXtreme Java to take advantage of advanced graphic display of map features and labels. We are no longer supporting JDK 1.1.x.

MapXtreme Java can optionally install a 1.22 VM, if you do not already have one.

MapXtremeServlet

MapXtreme Java has changed to a servlet architecture to better service requests for maps and data. The servlet, called MapXtremeServlet, replaces MapXtremeApp, the previous stand-alone Java server application. The Server Administrator tool, which administered MapXtremeApp, has gone away as well.

MapXtremeServlet continues to supply the same mapping services: image requests (GIF, JPEG), vector data requests (query methods), and metadata requests (table information, statistical information for columns in a table).

This servlet model requires a web server / servlet container configuration that supports servlets, such as Apache/Tomcat, Java Web Server, etc. Web servers, such as Microsoft IIS that do not support servlets directly,require a plug-in such as JRun or Tomcat.

See Chapter 3: Getting Started, to learn how to set up your servlet container with MapXtremeServlet.

Installation Conveniences

As a convenience to users, Apache Web Server is available on the CD. The installation process provides an optional step of integrating Tomcat with Apache and MapXtreme. Tomcat and Apache is a web server / servlet container that is freely distributed by the Apache Group. MapXtreme is designed to work with any web server / servlet container that supports Java 2.

For users of JDeveloper 3.0 for Windows, the installation program will optionally install a Servlet Wizard Addin to simplify the process of creating and configuring a custom servlet.

Installation and setup instructions are provided in Chapter 3: Getting Started.

Insert/Update/Delete for JDBC DataProviders

MapXtreme Java 3.0 supports read / write access of data for remote databases. This is provided through the enhanced JDBC DataProviders in MapXtreme. Layers representing Oracle8*i*, IUS, DB2, etc., are now editable. Features can be added and removed from these data sources using the same methods and techniques that have been used to edit Annotation layers (addFeatureSet, addFeature, removeFeature, replaceFeature). See Chapter 11 for more information.

This release also provides for simple transaction support. As a row is updated, it is locked and then unlocked once the update / insert / delete operation is completed.

Support for Oracle

This release of MapXtreme Java adds support for Oracle 8.1.6, as well as continued support for 8.1.5. Changes in 8.1.6 include a new format for representing GTYPEs; nodes are ordered differently than they were in 8.1.5; and a new metadata view, USER_SDO_GEOM_METADATA that replaces the older SDO_GEOM_METADATA table.

Drivers

This release supports the Oracle8*i* Thin, Thick and Thick Bequeth drivers. These ar JDBC 2.0 drivers provided by Oracle in classes12.zip.

SRID and Oracle 8.1.6

MapXtreme also supports Oracle 8.1.6 use of SRID coordinate system information. Oracle8*i* tables containing spatial geometry objects may include a Spatial Reference ID (SRID) data attribute that defines the coordinate system for the objects in the table. Queries sent to Oracle8*i* must now include an SRID if the Oracle data source includes one. If not, a Null value for the SRID can be passed.

To help you determine the SRID for the data source, MapXtreme provides a utility class OracleSRID that serves as a lookup table to map Oracle SRID to and from MapInfo CoordSys.

QueryBuilder

MapXtreme Java handles user-defined SQL queries without making any modifications to the query.Referred to as "pass-through" queries, MapXtreme will execute them as written and retrieve all the features in the layer. Note that this could return many features, even if the view port is small.

Pass-through queries are intended for power users who need complex queries to construct layer data and understand how to include the appropriate limiting conditions.

To assist such users with the limitations posed by "untouched" queries, MapXtreme Java provides an interface that will allow you to write your own call back objects to create modified query strings when rendering or performing searches on layers defined by pass-through queries. QueryBuilder is explained in Chapter 11: Features and Searches.

Connection Pooling

JDBC connection pooling can now be configured, thus aiding performance and increasing security by eliminating the need to transmit sensitive connection information over a network.

Connection pooling logic has been moved into a separate class, **MIConnectionPoolManager.** This pool manager is responsible for pre-starting named JDBC connections that are specified in a **miconnections.properties** file. This file must be on the classpath for it to be used by the pool manager. Additional pooled connections can be created programmatically through the API.

The pool manager maintains a "connection janitor" which runs periodically in a separate thread to close down unused JDBC connections.

Pooled connections can be created, edited, and deleted from the miconnections.properties file through a stand-alone program called Connections Manager (com.mapinfo.dp.util.ConnectionsManager). See Chapter 10: Accessing Remote Data, for more on connection pooling.

Map Definition Management

XML-based Map Definitions

Map Definitions in MapXtreme Java are now XML-based text files. Previously the map definitions were stored as serialized Java objects in a binary format. This change provides better forward compatibility and allows you to edit map definition files if you choose.

Map Definitions can be saved as a file or stored as a record within a RDBMS table.

With this change to XML-based map definitions, map definitions created in previous versions of MapXtreme will not work with this release of MapXtreme Java. You will need to create new map definitions either by using the Map Definition Manager or programmatically through the new MapDefContainer interface. Additionally, you can check the MapXtreme website for a 2.0 to 3.0 Map Definition converter (www.mapxtreme.com).

Saving Map Definitions to a Database

This release of MapXtreme Java extends support for saving map definitions by allowing you to save and load them from remote databases. Previously, map definitions could only be saved to a file.

Saving map definitions can be accomplished through Map Definition Manager or programmatically through the MapDefContainer interface. Both ways allow you to save the map definition as a file or to a database. When saving to a database, you can choose to save the information to the default MAPINFO.MAPDEFINITIONS table, specify a table in the database, or specify a query or insert/update statement that you want to use to save and load the information.

Coordinate System Chooser

This release now allows you to change the map's distance units and display coordinate system in the Map Definition Manager via the Map Options menu.

Theme Management

MapXtreme Java has added support for two new types of themes: Individual Value Themes, and Selection Themes.

Individual Value Themes

An Individual Value Theme allows the thematic shading of features based on specific attribute values for a specified column. For example, use Individual Value theme to modify the renditions of line features in a "Roads" table based on the values in the "StreetType" column. Street type values of "Rd" could be blue, while values of "Hwy" could be shaded red.

Selection Themes

A SelectionTheme can be used to change the rendition of all features in a Layer's Selection. This type of theme is commonly used to highlight a group of objects returned from a search method.

AddTheme Wizard Bean

A new AddTheme Wizard Bean is available to walk users through the process of creating a Ranged theme or Individual Value theme. This Bean can be added to the MapToolbar Bean for easy access. The theme can be based on any supported column and layer in the current map. Currently there is support for creating a theme based on numeric, string and date column data, and on point, line and region layers. Part of the operation will be to create a default theme legend which is associated with the new theme.

Themes and Layer Control

The Layer Control Bean has been enhanced to display themes in addition to map layers. Users can now move themes up or down and remove them from the map.

Legends for Themes and Cartographic Maps

You can now create, display, and manage legends for themes and cartographic symbols.

Theme legends are look-up keys that are tied directly to a theme. Theme Legends can be created for Ranged Themes and Individual Value Themes.

Cartographic legends represent features in a map layer.

LegendContainer Bean

As a way of managing how legends are laid out and displayed, MapXtreme Java includes a LegendContainer Bean that can be dropped in along-side a VisualMapJ object and display any legends that are associated with VisualMapJ. If the LegendContainer detects the addition or removal of a legend from VisualMapJ, it updates its display accordingly.

Renditions

The Rendering engine for MapXtreme Java 3.0 has been re-engineered to better support the many additional capabilities of the Java2D API as well as allow for future capabilities. The new engine is capable of displaying complex symbology, line and fill styles. New features include:

- Symbol paints for lines and regions
- Line markers (symbols that follow a line geometry)
- Dashed lines
- Line caps and joins
- Parallel lines
- Vector symbols
- Enhanced Font symbols
- Alpha compositing, including support for translucent fills.

Some Rendition properties have been changed or replaced in this release of MapXtreme Java. See Chapter 12: Labeling and Renditions, for details.

Labeling Enhancements

MapXtreme Java now supports more options than previous releases for enhanced label appearance and placement. In addition to the previously available bold, italic, and halo effects, text label options now include outlining, underlining, and boxing. Labeling can be horizontally and vertically aligned from an anchor position, as well as offset x and y distances from the anchor point.

In addition to controlling labeling through the API, MapXtreme Java includes a new label position chooser for the Label dialog.

Selectable Layer Objects

This release of MapXtreme Java adds a fourth characteristic to the behavior of MapJ Layer objects: selectability. Previously, Layer object properties included visible, editable, and autolabel.

Now when a (MapJ) Layer is selectable, you can choose an individual Feature or multiple features and perform additional actions on this subset.

New JavaBeans

Selection MapTools

New Selection MapTool Beans have been added to MapXtreme Java to expand the functionality of interacting with a map. The following MapTools are new to this release of MapXtreme Java:

- **ObjectSelectionMapTool**: use to select individual features when a mouse is clicked on the Feature.
- **BoundarySelectionMapTool**: use to select features contained within the bounding polygon of the unique (region) Feature in the topmost visible Layer that contains the point of the mouse click.
- **RadiusSelectionMapTool**: use to select features contained within a bounding circle formed by a mouse drag operation.
- **RectangleSelectionMapTool**: use to select features contained within a bounding rectangle formed by a mouse drag operation.
- **PolygonSelectionMapTool**: use to select features contained within a bounding polygon formed by a series of mouse click actions.

AddTheme Bean

As previously mentioned under Theme Management, above, this release of MapXtreme Java includes an AddTheme Bean, a wizard to allow you to easily add a Ranged Theme or Individual Value theme to your map.

LegendContainer Bean

Included in this release is a new Bean to display and manage theme and cartographic legends.

LayerControl Bean

The LayerControl Bean has been updated to display themes and allows them to be re-ordered or removed from the map display.

From the Layer Control users can add layers by clicking the Add button. This displays the AddLayer Wizard that guides you through the process. The Wizard is initialized from the **addlayerwizard.properties** file that contains information about your data source.

Localizatio

MapXtreme Java provides localized versions of its JavaBeans, Map Definition Manager, Connections Manager and installer in the following languages: German, Spanish, Italian, French, Swedish, Japanese, Korean, Simplified Chinese, and Traditional Chinese.

Printing/Plotting Support

In this release of MapXtreme Java, Visual MapJ has been enhanced to support printing and plotting of maps. In Windows the current 1.22 Java VM limits the print device's resolution to 72 dpi. On UNIX, MapPrinter will attempt to print to whatever the dpi for the device is.

Raster and Grid Support

This release includes support for displaying MapInfo continuous grid files. A grid file is a type of thematic map that displays data as continuous color gradations across the map. It is produced by an interpolation of point data from the source table and

displays as a raster image. To create a MapInfo grid file, use MapInfo Professional. For more on importing raster images, see Chapter 9: Mapping in Layers.

Shape File Support

ESRI Shapefiles are now a supported data source in MapXtreme Java. The Shapefile Data Provider uses the TableDescHelper and DataProviderHelper interfaces. It is functionally equivalent to theTAB Data Provider, except that character set encoding and coordinate system information is not maintained in Shapefiles. It must be passed to the TableDescHelper.

MapToolkit (Servlet Library)

MapXtreme includes a new class of helper methods for creating custom servlets. For an example of how these helper methods can be used, see the sample servlet HTMLEmbeddedMapServlet in the \sampleapps\Java\servlet directory.

Sample Applications

Several sample applications ship with MapXtreme Java. They can be found in \sampleapps after installation.

Servlet example: HTMLEmbeddedMapServlet.java

A customizable "client servlet" that uses MapXtremeServlet for generating map images that are viewed in an HTML form. This sample is further discussed in Chapter 3: Getting Started, and Chapter 7: Working With Servlets.

Servlet/AWT Applet example: MapperServlet

A customizable "client servlet" that uses MapXtremeServlet for generating map images that are viewed in an applet. The applet is AWT-based, with no Swing requirements, and no plugins or JavaBeans are involved.

Swing Applet example: SimpleMap

An applet that demonstrates using our JavaBeans (VisualMapJ, MapToolbar) to provide basic map display, map navigation, and theme/legend display using the Add Theme Wizard and LegendContainer. This applet communicates directly with

MapXtremeServlet. This applet requires the Java 2 platform (or plug-in). This example is further discussed in Chapter 6: MapXtreme JavaBeans.

HAHTsite 4.0 example: JavaHelloWorld

This Java example demonstrates basic map navigation, use of Layer Control and display of a scalebar in an HTML form. Server-side, the MapJ object and the business logic are managed by the HAHTsite Server application (which in turn uses MapXtremeServlet).

Changes Between MapXtreme Java 2.0.x to 3.0

AppSupportSettingsReader

Version 3.0 no longer supports the AppSupportSettingsReader object. Some version 2 applications used this object as a convenient way of retrieving the "Preferences" settings that were set through the MapXtreme Administrator's Preferences tab (e.g. the Maps Directory preference, which stored the directory where map files are installed).

If you are upgrading a version 2 application that used this object, you will need to delete any references to this object. For example, instead of reading settings through calls such as this:

```
AppSupportSettingsReader reader = new
   AppSupportSettingsReader(m_host, m_adminPort);

String serverMapPath = reader.getDefaultMapPath();
```

You could simply assign the values explicitly:

```
String serverMapPath = "C:\mapxtreme\maps\";
```

Or you could read such settings in from a properties file that you set up yourself.

Similarly, you would need to delete any import statements that reference the deleted object:

```
import
   com.mapinfo.mapxtreme.util.AppSupportSettingsReader;
```

DataProviderHelper

All DataProviderHelper objects that took verbose, seamed, and cached flags have been deprecated. DataProviderHelpers for JDBC Layers should be changed over to use connection pooling.

TableDescHelpers

The constructors for JDBC Layers that took either a table name or SQL statement are deprecated. Use the newer versions that specify only a table name or SQL statement. These additionally expose new functionality such as the ability to specify per-Feature Renditions.

MapXtremeDataProviderRef

The constructors that accepted a host and port to MapXtremeApp will no longer work. Call a non-deprecated constructor that takes an URL for MapXtremeServlet.

RendererParams

The constructors in version 2 that accepted a host and port to MapXtremeApp will no longer work. To have VisualMapJ access rendering services from MapXtremeServlet, call the non-deprecated constructor that takes a MapXtremeServlet URL and image MIME type.

MapDefParams

The constructors that accepted a host and port to MapXtremeApp will no longer work. To have VisualMapJ load a map definition or geoset from a MapXtremeServlet, you will need to use one of MapJ's loadGeoset() or loadMapDefinition() methods which allow this. (Use VisualMapJ's getMapJ() method to obtain a reference to its current MapJ.)

Theme Ordering

Themes within a Layer's ThemeList now follow the same ordering as Layer objects within the Layers collection. Themes at the top of the ThemeList take precedence over Themes in a lower position. In previous versions Themes at the bottom of the ThemeList had the higher priority.

Adding Layers

There are new versions of Layers' add and insert methods that take an additional parameter for the Layer name. This parameter is required for Layers defined by SQL statements. While optional for Layers defined by table names, it is highly recommended to always specify a name for JDBC Layers, as this avoids the runtime overhead involved in determining a name.

Adding/Loading MapDefinitions and Geosets

Methods that accepted a host and port to MapXtremeApp no longer work. To load a map definition from a remote Web server, use the version of **MapJ.loadMapDefinition** that accepts an InputStream.To load a remote geoset use **MapJ.loadGeoset** that accepts an input stream, data directory and servlet URL. Use **Layers.addMapDefinition** that takes an input stream and **Layers.addGeoset** that takes an input stream, data directory,and servlet URL.

Defining JDBC Layers by SQL statements

JDBC Layers defined by SQL have been problematic in the past, as MapXtreme could not always derive queries for these Layers that produced meaningful or valid SQL. MapXtreme now allows users working with query-based JDBC Layers to specify the queries MapXtreme Java should execute. This is done through a QueryBuilder interface. Use the IdentityQueryBuilder to quickly convert your project to version 3.0 and then customize the QueryBuilder to optimize your performance.