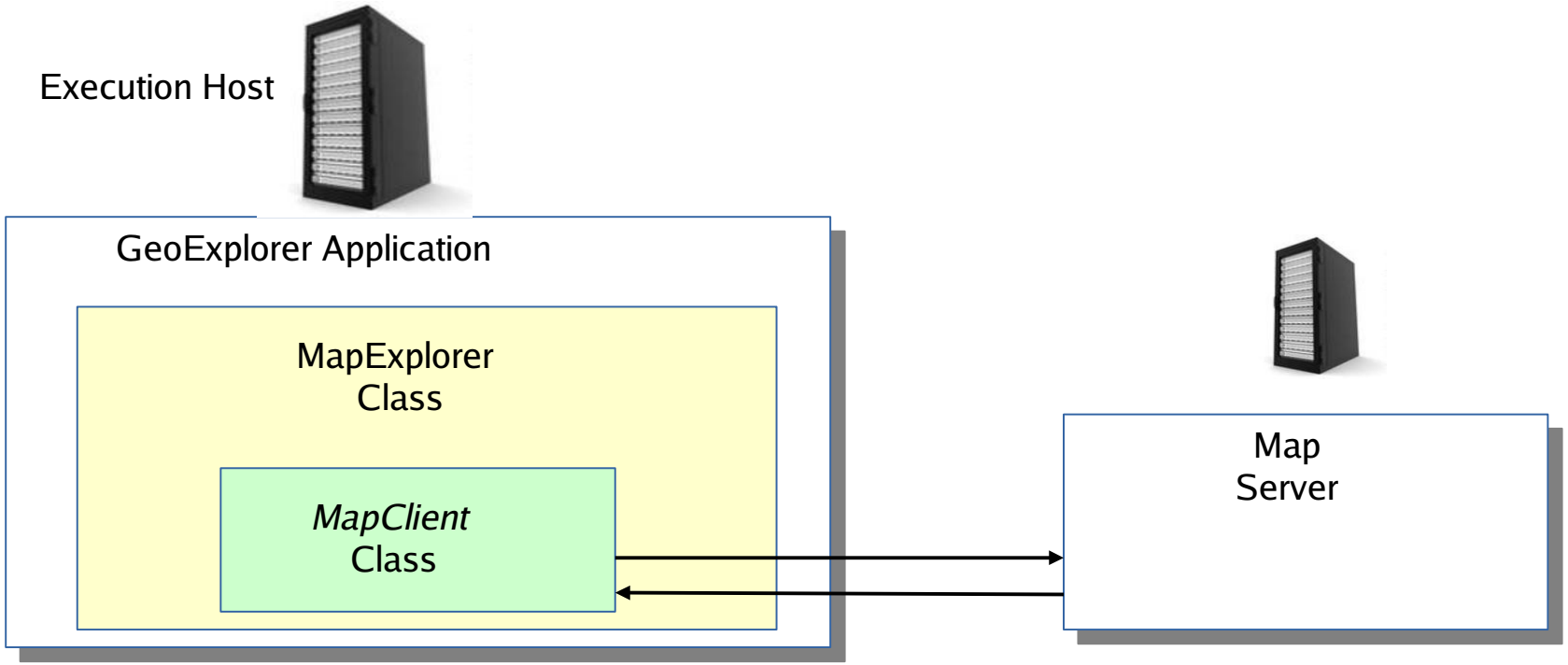
MapExplorer Architecture



- GeoExplorer app uses MapExplorer class. MapExplorer creates a *MapClient* widget instance.
- *MapClient* talks to Map Server. MapExplorer knows nothing about the server. Only calls methods in *MapClient* class.

MapClient Class Public API

```
itcl::class Rappture::MapClient {
    inherit itk::Widget

    constructor { args } {} {}
    destructor {}
    public method background {}
    public method coordstolatlong { list }
    public method envelope {}
    public method invtransform { list }
    public method latlongtocoords { list }
    public method layer { option args }
    public method load { file }
    public method pan { x y }
    public method resize { w h }
    public method transform { list }
    public method zoom { option args }
}
```

- *MapClient* has public methods used by MapExplorer class.
- Layer and zoom methods have multiple options.

Map Object XML

```
<map>
    <about>
        <label>World map</label>
        <description>Map of the world.</description>
    </about>
    <stylesheet>
      ${TOOLDIR}/data/stylesheet1.xml
    </stylesheet>
    <layers>
     <layer id="Building Names"/>
    </layers>
</map>
```

- Map object is simplified to what application needs.
- Map details in server-specified stylesheet.
- Layers tag used to provide user-friendly names to application.

MapClient Class Public API

```
itcl::body Rappture::MapClient::constructor {args} {
    private variable _image

    itk_component add canvas {
        canvas $itk_interior.canvas -highlightthickness 0 -borderwidth 0
    } {
        usual
        ignore -highlightthickness -borderwidth  -background -width -height
    }
    set _image(map) [image create photo -width 500 -height 500]
    set c $itk_component(canvas)
    $c create image 0 0 -anchor nw -image $_image(map)
    # Fix the scrollregion in case we go off screen
    $c configure -scrollregion [$c bbox all]
    blt::table $itk_interior \
        0,0 $itk_component(canvas) -fill both
    eval itk_initialize $args
}
```

- *MapClient* creates Tk canvas component and image item.
- This is a widget API. Could be smaller: only a canvas or canvas item API.

MapClient Class Public API

```
#
# background --
#
#   This public method used to get the name of the image used in the
#   widget.
#
itcl::body Rappture::MapClient::background {} {
    return $_image(map)
}
```

- Background method returns image of map.
- Used for download, etc.

MapClient Class Public API

```
#
# envelope --
#
#   This public method is called by clients using this widget to get
#   the map coordinates of the extents of the map.  The coordinates
#   as the two opposite corners.
#
itcl::body Rappture::MapClient::envelope {} {
    set x1 0
    set y1 0
    # x2 and y2 are outside the area of the map.
    set x2 101
    set y2 101
    return [list $x1 $y1 $x2 $y2]
}
```

- Envelope method returns bounding box of map in map coordinates.
- Used for saving zoom region, etc.

MapClient Class Public API

```
#
# transform --
#
#   This public method is called by clients using this widget to transform
#   screen coordinates to map coordinates.  The input list is a list of x-y
#   screen coordinates.  The output list is a list of map coordinates.
#
itcl::body Rappture::MapClient::transform {list} {
}
```

- Transform converts screen coordinates to map coordinates.
- Coordinates are X-Y pairs. Same number of map coordinate pairs returned.
- Used for picking, placing points, etc.

MapClient Class Public API

```
#
# invtransform --
#
#   This public method is called by clients using this widget to transform
#   map coordinates back to screen coordinates. The input list is a list of
#   x-y mp coordinates.  The output list is a list of screen coordinates.
#
itcl::body Rappture::MapClient::invtransform {list} {
}
```

- **Invtransform** converts map coordinates to screen coordinates.
- Coordinates are map coordinate pairs. Same number of X-Y coordinate pairs returned.
- Used for placing pins and markers, etc.

MapClient Class Public API

```
#
# latlongtocoords --
#
#   This public method is called by clients using this widget to transform
#   lat/long coordinates to map coordinates.
#
itcl::body Rappture::MapClient::latlongtocoords {list} {
}
```

- Latlongtocoords converts lat/long coordinates to map coordinates.
- Coordinates are lat/long pairs. Same number of map coordinate pairs returned.
- Used for placing pins and markers, etc.

MapClient Class Public API

```
#
# coordstolatlong --
#
#   This public method is called by clients using this widget to transform
#   map coordinates back to lat/long coordinates.
#
itcl::body Rappture::MapClient::coordstolatlong {list} {
}
```

- Coordstolatlong converts map coordinates to lat/long coordinates.
- Coordinates are map coordinate pairs. Same number of lat/long coordinate pairs returned.
- Used for placing pins and markers, etc.

MapClient Class Public API

```
#
# load --
#
#   This public method is called by clients using this widget to load map
#   spreadsheet data into the server.
#
itcl::body Rappture::MapClient::load { contents } {
}
```

- Load loads map stylesheet into server.
- MapExplorer doesn't parse or verify stylesheet contents.
- Contents are specific to Map Server.

MapClient Class Public API

```
#
# loadfile --
#
#   This public method is called by clients using this widget to load map
#   spreadsheet data into the server.
#
itcl::body Rappture::MapClient::loadfile { fileName } {
}
```

- Loadfile passes file path of stylesheet to server.
- MapExplorer doesn't parse or verify stylesheet file.
- Stylesheets are specific to Map Server.

MapClient Class Public API

```
#
# layer --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
itcl::body Rappture::MapClient::layer {option name args} {
    # Test if name is a valid layer.

    # Handle option
    switch -- $option {
        "create" {
        }
        "delete" {
        }
        ...
    }
}
```

- **Layer** method has several subcommands: **create**, **delete**, **exists**, **configure**, **cget**, and **names**.
- Layer name must be verified by *MapClient*.

MapClient Class Public API

```
#
# layer create --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer create type layerName ?option value ...?

itcl::body Rappture::MapClient::Layer::create {type layerName args} {

}
```

- Create method creates new layer.
- Option and value are specific to the type of layer.
- Layer names must be unique.
- Not used yet.

MapClient Class Public API

```
#
# layer delete --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer delete ?layerName ...?

itcl::body Rappture::MapClient::Layer::delete {args} {

}
```

- Delete method deletes zero or more existing layers.
- Not used yet.

MapClient Class Public API

```
#
# layer exists --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer exists layerName

itcl::body Rappture::MapClient::Layer::exists { name } {

}
```

- Exists method returns 1 if layer exists, 0 otherwise.
- Not used yet.

MapClient Class Public API

```
#
# layer configure --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer configure layerName ?option value…?

itcl::body Rappture::MapClient::Layer::configure { name args } {
}
```

- **Configure** method configure attributes of layer.
- Layer must already exist.
- Currently only option is “-visible bool”.
- Option and value are specific to the type of layer.

MapClient Class Public API

```
#
# layer cget --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer cget layerName option

itcl::body Rappture::MapClient::Layer::cget { name option } {
}
```

- Cget method returns value of specified option.
- Layer must already exist.
- Currently only option is “-visible”.
- Option and value are specific to the type of layer.

MapClient Class Public API

```
#
# layer names --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer names ?pattern …?

itcl::body Rappture::MapClient::Layer::names { args } {
}
```

- Names method returns names of all layers.
- Zero or more pattern arguments may be present.
- Uses glob-type patterns.

MapClient Class Public API

```
#
# layer type --
#
#   This public method is called by clients using this widget to manipulate
#   layers.
#
#       layer type layerName

itcl::body Rappture::MapClient::Layer::type { layerName } {
}
```

- Type method returns type of given layer.
- Not used presently.

MapClient Class Public API

```
#
# pan --
#
#   This public method is called by clients using this widget to pan the
#   map. X and Y are deltas in screen coordinates.
#
#       pan x y
#
itcl::body Rappture::MapClient::pan {dx dy} {
}
```

- Pan method pans the map by the given x and y deltas.
- Deltas are in pixels.

MapClient Class Public API

```
#
# resize --
#
#   This public method is called by clients using this widget to resize the
#   map on the server.  Width and height are in pixels.
#
itcl::body Rappture::MapClient::resize { w h } {
}
```

- Resize method resize the map by the given width and height.
- Width and height are in pixels.

MapClient Class Public API

```
#
# zoom --
#
#   This public method is called by clients using this widget to set the
#   zoom level of the map.
#
#       zoom in
#       zoom out
#       zoom reset
#       zoom to
#
itcl::body Rappture::MapClient::zoom {option args} {
    switch -- $option {
    …
}
```

- **Zoom** method zooms the map.
- Subcommands are **in**, **out**, **reset** and to.

MapClient Class Public API

```
#
# zoom in --
#
#   This public method is called by clients using this widget to set the
#   zoom level of the map.
#
#
itcl::body Rappture::MapClient::Zoom::in {} {
}
```

- In method zooms the map in one level.
- Example: “zoom in”

MapClient Class Public API

```
#
# zoom out --
#
#   This public method is called by clients using this widget to set the
#   zoom level of the map.
#
#
itcl::body Rappture::MapClient::Zoom::out {} {
}
```

- Out method zooms the map out one level.
- Example: “zoom out”

MapClient Class Public API

```
#
# zoom reset --
#
#   This public method is called by clients using this widget to set the
#   zoom level of the map.
#
#
itcl::body Rappture::MapClient::Zoom::reset {} {
}
```

- Reset method returns map to default zoom level.
- Example: “zoom reset”

MapClient Class Public API

```
#
# zoom to --
#
#   This public method is called by clients using this widget to set the
#   zoom level of the map.
#
#
itcl::body Rappture::MapClient::Zoom::to { x y scale } {
}
```

- To method sets zoom center at x, y with zoom factor of scale.
- X and Y are map coordinates.
- Example: “zoom to 100 100 0.9”
- Not used presently.

MapClient Class Public API

```
#
# zoom box --
#
#   This public method is called by clients using this widget to set the
#   zoom level of the map.
#
#
itcl::body Rappture::MapClient::Zoom::box { list } {
}
```

- Box method sets zoom from bounding box in list.
- Bounding box list is “x1 y1 x2 y2”. Coordinates are map coordinates. X2 and y2 are not in the box.
- Example: “zoom box [envelope].