**Directions**

**Mapquest API**

Jacob Schoenfeld

**I. Purpose and Function.**

The Purpose of Directions API is to retrieve data from the Mapquest API server. How it performs this is through a combination of C#’s URI interface to communicate with the Mapquest API and also with the use of Newtonsoft’s json parser which is used to parse the returned json string.

The data returned from the Mapquest API is then used by the front end to display directions, maneuvers and other forms of data which has been returned by the server.

**II. Dependancies and Installing**

1. Mapquest API Key – The Mapquest key is the key required to access Mapquest’s API servers.
   1. Licensed Key is used to make calls to Mapquest’s server
   2. Free Licensed Key is used to make calls to Mapquest’s free server.
      1. Note: if using open key, need to change url to -open.mapquest.com.
2. Newtonsoft – Libraries required for the proper method of parsing Json strings. These json strings are the returned values from the Mapquest API server.
   1. To understand more of what Newtonsoft is, please take a look at their documentation at this site: <http://www.newtonsoft.com/json/help/html/Introduction.htm>
3. Install Procedure

If you wish to install the Directions API class into your own project please follow these steps.

* 1. Copy Directions Request Class Folder, Directions Result Class Folder and StaticMap Request Classes folders and App.config file into your project.
  2. Next, include them into your project through the Visual Studios Solution Explorer.
  3. Install Newtonsoft through the New get Packages in Visual Studios.
  4. Finally, Add references to the configuration manager and to System.drawing

**III. Directions Request Classes**

1. Address.cs
   1. Public Class
   2. Holds data for Addresses.
   3. Attributes:
      1. street
         1. Type: string
         2. The street of the address.
      2. city
         1. Type: string
         2. The city of the address.
      3. state
         1. Type: string
         2. The state of the address.
      4. postalCode
         1. Type: string
         2. This is the zip code of the address.
   4. Constructors:
      1. Address() – defaults are all empty strings
      2. Address(string street, string city, string state, string postalCode)
   5. Functions:
      1. getLocations()
         1. This is the geocoding api call which finds the locations of the addresses. This makes sure that these are valid locations and returns a list of the locations.
         2. Parameters: No parameters.
         3. returns a List of Locations.
2. DirectionsRequest.cs
   1. Abstract Public Class
   2. To retrieve the directions from the Mapquest API based off of a list of location classes.
   3. Enums:
      1. Avoids – avoid options provided by the api server
         1. LimitedAccess – Highway roads.
         2. TollRoad - If the route has toll roads
         3. SeasonalClosure – if the route closes seasonally
         4. Unpaved – If the route has unpaved roads.
         5. Ferry – If the route contains a Ferry.
         6. CountryBorderCrossing – If the route contains a crossing from one country to the other.
      2. RouteType – travel options provided by the api server
      3. DrivingStyle – driving options provided by the api server
   4. Functions:
      1. getDirections
         1. Parameters:
            1. List<Locations> locations
            2. StaticMap.Maptype mapType
            3. RouteType routeType

Default: RouteType.Fastest

* + - * 1. DrivingStyle drivingStyle

Default: DrivingStyle.Normal

* + - * 1. List<Avoid> avoids

Default: null

* + - * 1. Bool enchancedNarrative

Default: false

* + - * 1. Bool metric

Default: false

* + - * 1. Int fuelEfficiency

Default: 21 – api default

* + - 1. Returns a Directions class.

**IV. Directions Result Classes**

1. BoundingBox.cs
   1. Public Class
   2. BoundingBox contains the coordinates of the entire box.
   3. Attributes:
      1. ul
         1. Type: LatLng
         2. Permissions: Public
         3. Returns the upper left hand side latlng
      2. lr
         1. Type: LatLng
         2. Permissions: Public
         3. Returns the lower left hand side latlng
   4. Constructors:
      1. BoundingBox()
         1. Default constructor
            1. ur – default LatLng()
            2. lr – default LatLng()
         2. BoundingBox(LatLng upperLeft, LatLng lowerRight)
   5. Functions: No necessary functions
2. Directions.cs
   1. Public Class
   2. Base Class for the mapquest json string. It contains all necessary data required for user.
   3. Attributes
      1. time
         1. Type: int
         2. Permissions: Public
         3. Returns the calculated elapsed time in seconds for the route.
      2. realTime
         1. Type: int
         2. Permissions: Public
         3. Returns the current Real Time.
      3. fuelUsed
         1. Type: double
         2. Permissions: Public
         3. Returns the estimated amount of fuel used during the route.
      4. distance
         1. Type: double
         2. Permissions: Public
         3. Returns the calculated distance of the route.
      5. hasFerry
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains a Ferry attribute. Otherwise it returns false.
      6. hasUnpaved
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains an Unpaved attribute, otherwise it returns false.
      7. hasHighway
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg has a Limited Access/Highway attribute, otherwise it returns false.
      8. hasTollRoad
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains a Toll Road attribute, otherwise it returns false.
      9. hasCountryCross
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains a Country Cross attribute, otherwise it returns false.
      10. hasSeasonalClosure
          1. Type: bool
          2. Permissions: Public
          3. Returns true if at least one leg contains a Seasonal Closure attribute, otherwise it returns false.
      11. map\_
          1. Type: StaticMap
          2. Permissions: Private
          3. Storage used for storing the map returned from the Mapquest API.
      12. mapDetails
          1. Type: StaticMap
          2. Permissions: Public
          3. Stores an instance of map\_ to access the data.
      13. mainMap
          1. Type: Image
          2. Permissions: Public
          3. Part of downloads process for the map and is the container for the map image.
      14. sessionID
          1. Type: string
          2. Permissions: Public
          3. Returns the unique identifier used to refer to a session. An existing session id will be used if provided, otherwise a new one will be created. The route stored in the session will be automatically updated if the session id is provided. Expires after 30 mins.
      15. formattedTime
          1. Type: string
          2. Permissions: Public
          3. Returns the calculated elapsed time as formatted text in HH:MM:SS format.
      16. shape
          1. Type: Shape
          2. Permissions: Public
          3. A collaction of latlong coordinates or shape points for the entire route highlight based on original mapstate and/or the generalize option. Shape is an alternated array of lat/lngs. Evens are lat and odds are lng.
      17. options
          1. Type: Options
          2. Permissions: Public
          3. Returns routing options.
      18. boundingBox
          1. Type: BoundingBox
          2. Permissions: Public
          3. Returns lat.lng bounding rectangle of all points in the latlng collaction; Returns the best fit for route shape.
      19. locationSequence
          1. Type: List<int>
          2. Permissions: Public
          3. Returns a sequence array that can be used to determine the index in the original location object list.
      20. legs
          1. Type: List<Leg>
          2. Permissions: Public
          3. Returns a collection of leg objects, one for each "leg" of the route.
      21. locations
          1. Type: List<Locations>
          2. Permissions: Public
          3. Returns a collection of locations in the form of an address.
   4. Constructors
      1. Directions()
         1. mapDetails
            1. set to null
         2. time
            1. set to 0
         3. realTime
            1. set to 0
         4. fuelUsed
            1. set to 0.0
         5. distance
            1. set to 0.0
         6. hasFerry
            1. set to false
         7. hasUnpaved
            1. set to false
         8. hasHighway
            1. set to false
         9. hasTollRoad
            1. set to false
         10. hasContryCross
             1. Set to false
         11. hasSeasonalClousre
             1. Set to false
         12. sessionId
             1. Set to String.Empty
         13. formattedTime
             1. set to StringEmpty
         14. shape
             1. Set to new instance of Shape()
         15. options
             1. Set to new instance of Options()
         16. boundingBox
             1. Set to new instance of BoundingBox();
         17. locationSequence
             1. Set to new instace of List<int>()
         18. legs
             1. Set to a new List<leg>
         19. Locations
             1. Set to a new List<locations>
   5. Functions
      1. addHighlighting
         1. Parameters:
            1. List<LatLng> points
            2. Color borderColor
            3. Color fillColor
            4. int linewidth

default – 5

1. DirectionsRootObject.cs
   1. Public Class
   2. Newtonsoft requires a base class for parsing the json string.
   3. Attributes:
      1. Route
         1. Type Directions
         2. Permissions: Public
         3. Container for the directions class required for newtonsoft to parse properly
   4. Constructors:
      1. DirectionsRootObject
         1. route
            1. Set to new instance of Directions()
2. LatLng.cs
   1. Public Class
   2. The Longitude and Latitude ADT class
   3. Attributes
      1. lng
         1. Type: double
         2. Permissions: Public
         3. Returns longitude value.
      2. Lat
         1. Type: double
         2. Permissions: Public
         3. Returns latitude value.
   4. Constructors:
      1. LatLng()
         1. Lng
            1. Set to 200 because 200 isn’t a valid lat, so works as a defaulted value.
         2. Lat
            1. Set to 200 because 200 isn’t a valid lat, so works as a defaulted value.
      2. LatLng(Double Latitude\_,Double Longitude\_)
   5. Functions: No additional functions.
3. Leg.cs
   1. Public Class
   2. Describes one "leg" of a route. It contains the maneuvers describing how to get from one location to the next location. Each leg will contain a variety of information, including index, time, distance, and formattedTime.
   3. Attributes:
      1. time
         1. Type: int
         2. Permissions: Public
         3. Returns the calculated elapsed time in seconds for the route.
      2. index
         1. Type: int
         2. Permissions: Public
         3. Returns the index position of the current leg.
      3. origIndex
         1. Type: int
         2. Permissions: Public
         3. Returns the origin index. Origin index is the index of the first non-collapsed maneuver.
      4. destIndex
         1. Type: int
         2. Permissions: Public
         3. Returns the destination index which is the index of the last non-collapsed maneuver.
      5. distance
         1. Type: double
         2. Permissions: Public
         3. Returns the calculated distance of the route.
      6. hasFerry
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains a Ferry attribute. Otherwise it returns false.
      7. hasUnpaved
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains an Unpaved attribute, otherwise it returns false.
      8. hasHighway
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg has a Limited Access/Highway attribute, otherwise it returns false.
      9. hasTollRoad
         1. Type: bool
         2. Permissions: Public
         3. Returns true if at least one leg contains a Toll Road attribute, otherwise it returns false.
      10. hasCountryCross
          1. Type: bool
          2. Permissions: Public
          3. Returns true if at least one leg contains a Country Cross attribute, otherwise it returns false.
      11. hasSeasonalClosure
          1. Type: bool
          2. Permissions: Public
          3. Returns true if at least one leg contains a Seasonal Closure attribute, otherwise it returns false.
      12. formattedTime
          1. Type: string
          2. Permissions: Public
          3. Returns the calculated elapsed time as formatted text in HH:MM:SS format.
      13. destNarrative
          1. Type: string
          2. Permissions: Public
          3. Returns the rephrased destination narrative string for the destination maneuver.
      14. maneuvers
          1. Type: List<Maneuver>
          2. Permissions: public
          3. A collection of Maneuver objects.
      15. roadGradeStrategy
          1. Type: List<List<string>>
          2. Permissions: public
          3. Only for Bicycle route type, defines the road
   4. Constructors:
      1. Leg()
         1. time set to 0
         2. index set to 0
         3. origIndex set to 0
         4. destIndex set to 0
         5. distance set to 0
         6. hasFerry set to false
         7. hasHighway set to false
         8. hasUnpaved set to false
         9. hasTollRoad set to false
         10. hasCountryCross set to false
         11. hasSeasonalClosure set to false;
         12. origNarrative set to ""
         13. formattedTime set to ""
         14. destNarrative set to ""
         15. maneuvers set to new instance of List<Maneuver>()
         16. roadGradeStrategy set to new instance of List<List<string>>()
   5. Functions: no additional functions
4. Location.cs
   1. Public Class
   2. Contains data necessary to store Addresses.
   3. Attributes:
      1. linkId
         1. Type: string
         2. Permissions: Public
         3. Returns linkId
      2. dragPoint
         1. Type: bool
         2. Permissions: Public
         3. Returns whether or not the point is a drag point.
            1. True: Location is a drag point
            2. False: Location is not a drag point
      3. type
         1. Type: string
         2. Permissions: Public
         3. Returns type of location.
            1. s = stop(default)
            2. v = via
      4. street
         1. Type: string
         2. Permissions: Public
         3. Returns Street Address
      5. countryName
         1. Type: string
         2. Permissions: Public
         3. Returns Country Name
      6. countyName
         1. Type: string
         2. Permissions: Public
         3. Returns County Name
      7. cityName
         1. Type: string
         2. Permissions: Public
         3. Returns City Name
      8. postalCode
         1. Type: string
         2. Permissions: Public
         3. Returns postal code
      9. sideOfStreet
         1. Type: string
         2. Permissions: Public
         3. Specifies the side of the street:
            1. L = left
            2. R = right
            3. N = none (default)
      10. countryNameType
          1. Type: string
          2. Permissions: Public
          3. Returns countryNameType
      11. cityNameType
          1. Type: string
          2. Permissions: Public
          3. Returns cityNameType
      12. countyNameType
          1. Type: string
          2. Permissions: Public
          3. Returns countyNameType
      13. stateNameType
          1. Type: string
          2. Permissions: Public
          3. Returns stateNameType
      14. geocodeQuality
          1. Type: string
          2. Permissions: Public
          3. Returns the precision of the geocoded location.
      15. geocodeQualityCode
          1. Type: string
          2. Permissions: Public
          3. Returns the five character quality code for the precision of the geocode location.
      16. latLng
          1. Type: LatLng
          2. Permissions: Public
          3. Longitude and Latitude ADT
      17. map\_
          1. Type: StaticMap
          2. Permissions: Private
          3. Storage used for storing the map returned from the Mapquest API.
      18. mapDetails
          1. Type: StaticMap
          2. Permissions: Public
          3. Stores an instance of map\_ to access the data.
      19. mainMap
          1. Type: Image
          2. Permissions: Public
          3. Part of downloads process for the map and is the container for the map image.
   4. Constructors:
      1. Locations()
         1. mapDetails\_ set to null
         2. linkId set to String.Empty
         3. dragPoint set to false
         4. type set to String.Empty
         5. street set to String.Empty
         6. countryName set to String.Empty
         7. stateName set to String.Empty
         8. countyName set to String.Empty
         9. cityName set to String.Empty
         10. postalCode set to String.Empty
         11. sideOfStreet set to String.Empty
         12. countryNameType set to String.Empty
         13. cityNameType set to String.Empty
         14. countyNameType set to String.Empty
         15. geocodeQuality set to String.Empty
         16. geocodeQualityCode set to String.Empty
         17. stateNameType set to String.Empty
         18. latLng set to new LatLng()
         19. displayLatLng set to new LatLng()
   5. Functions: No other functions
5. LocationsRootObject.cs
   1. Public Class
   2. Base class required for newtonsoft json parsing.
   3. Attributes:
      1. Results:
         1. Type: List<Result>
         2. Permissions: Public
         3. Returns a list of results.
   4. Constructors:
      1. LocationsRootObject()
         1. Result set to new instance of List<Result>
6. Maneuver.cs
   1. Public Class
   2. Manuevers describes one step in a route narrative
   3. Enums:
      1. Attributes
         1. Public
         2. Possible attributes a given maneuver can have.
            1. None = 0
            2. portionsToll = 1
            3. portionsUnpaved = 2
            4. possibleSeasonalRoadClosure = 4
            5. gate = 8
            6. ferry = 16
            7. avoidId = 32
            8. countryCrossing = 64
            9. limitedAccessHighways = 128
      2. Direction
         1. Public
         2. Possible directions a given maneuver can go.
            1. None = 0
            2. North = 1
            3. Northwest = 2
            4. Northeast = 3
            5. South = 4
            6. Southeast = 5
            7. Southwest = 6
            8. West = 7
            9. East = 8
      3. TurnType
         1. Public
         2. Possible types of turns a given maneuver can have
            1. Straight = 0
            2. slightRight = 1
            3. right = 2
            4. sharpRight = 3
            5. Reverse = 4
            6. sharpLeft = 5
            7. left = 6
            8. slightLeft = 7
            9. rightUturn = 8
            10. leftUTurn = 9
            11. rightMerge = 10
            12. leftMerge = 11
            13. rightOnRamp = 12
            14. leftOnRamp = 13
            15. rightOffRamp = 14
            16. leftOffRamp = 15
            17. rightFork = 16
            18. leftFork = 17
            19. straightFork = 18
   4. Attributes:
      1. Time
         1. Type: int
         2. Permissions: Public
         3. Returns calculated elapsed time in seconds for maneuver
      2. Index
         1. Type: int
         2. Permissions: Public
         3. Returns the current maneuver position in the index list of maneuvers.
      3. Distance
         1. Type: double
         2. Permissions: Public
         3. Returns the calculated distance of this maneuver
      4. mapUrl
         1. Type: string
         2. Permissions: Public
         3. Returns a URL to a static map of this maneuver
      5. iconUrl
         1. Type: string
         2. Permissions: Public
         3. Returns a URL of the icon of this maneuver
      6. Narrative
         1. Type: string
         2. Permissions: Public
         3. Returns Textual driving directions for a particular maneuver.
      7. transportMode
         1. Type: string
         2. Permissions: Public
         3. This is a string indicating the mode of transportation used for the maneuver.
      8. formattedTime
         1. Type: string
         2. Permissions: Public
         3. Returns the calculated elapsed time as formatted text in HH:MM:SS format
      9. directionName
         1. Type: string
         2. Permissions: Public
         3. Returns the direction name
      10. turnType
          1. Type: Maneuver.TurnType
          2. Permissions: Public
          3. Returns the type associated to a particular maneuver
      11. Direction
          1. Type: Maneuver.Direction
          2. Permissions: Public
          3. Returns the direction associated to a particular maneuver.
      12. Direction
          1. Type: Maneuver.Attributes
          2. Permissions: Public
          3. Returns the attributes associated to a particular maneuver.
      13. startPoint
          1. Type: LatLng
          2. Permsions: Public
          3. Returns the 1st shape point latLng for a particular maneuver.
      14. Signs
          1. Type: List<Sign>
          2. Permissions: Public
          3. Returns text name, extra text, type(road shield) and direction present for a particular maneuver.
      15. linkIds
          1. Type : List<int>
          2. Permissions: Public
          3. Returns a list of linkIds
      16. Streets
          1. Type: List<string>
          2. Permissions: Public
          3. Returns the collection of sto names this maneuver applies streets.
      17. ManeuverNotes
          1. Type: List<string>
          2. Permissions: Public
          3. Returns the maneuver note for a particular maneuver.
7. Options.cs
   1. Public Class
   2. Options for routing
   3. Attributes:
      1. timeType
         1. Type: int
         2. Permissions: Public
         3. Returns the time Type
      2. maxLinkId
         1. Type: int
         2. Permissions: Public
         3. Returns maxLinkId
      3. Generalize
         1. Type: int
         2. Permissions: Public
         3. Returns generalization factor.
      4. routeNumber
         1. Type: int
         2. Permissions: Public
         3. Return route number
      5. drivingStyle
         1. Type: int
         2. Permissions: Public
         3. Return driving style to be used when calculating fuel usage.
      6. walkingSpeed
         1. Type: int
         2. Permissions: Public
         3. Returns speed allowed for walking in mph
      7. transferPenalty
         1. Type: int
         2. Permissions: Public
         3. Returns transfer penalty
      8. maneuverPenalty
         1. Type: int
         2. Permissions: Public
         3. Returns maneuver penalty
      9. urbanAvoidFactor
         1. Type: int
         2. Permissions: Public
         3. Returns avoid urban factor
      10. filterZoneFactor
          1. Type: int
          2. Permissions: Public
          3. Returns the response as a part of the options and represents an internal route option usd in alternate routes.
      11. highwayEfficiency
          1. Type: int
          2. Permissions: Public
          3. This is the fuel efficiency of your vehicle, given as miles per gallon.
      12. cyclingRoadFactor
          1. Type: int
          2. Permissions: Public
          3. Returns the cycling road favoring factor
      13. maxWalkingDistance
          1. Type: int
          2. Permissions: Public
          3. Returns maximum Walking distance.
      14. useTraffic
          1. Type: Bool
          2. Permissions: Public
          3. Returns traffic status
      15. enhancedNarrative
          1. Type: Bool
          2. Permissions: Public
          3. Enchanced Narrative option.
      16. sideOfStreetDisplay
          1. Type: Bool
          2. Permissions: Public
          3. Returns side of street to be displayed
      17. returnLinkDirections
          1. Type: Bool
          2. Permissions: Public
          3. Return Link Directions
      18. avoidTimedConditions
          1. Type: Bool
          2. Permissions: Public
          3. Returns avoid Timed Conditions
      19. stateBoundaryDisplay
          1. Type: Bool
          2. Permissions: Public
          3. Returns State boundary display option
      20. countryBoundaryDisplay
          1. Type: Bool
          2. Permissions: Public
          3. Returns Country boundary display option
      21. destinationManeuverDisplay
          1. Type: Bool
          2. Permissions: Public
          3. Returns the end at destination maneuver display option
      22. unit
          1. Type: string
          2. Permissions: Public
          3. Returns Specifies the type o units to use when calculating distance.
      23. Locale
          1. Type: string
          2. Permissions: Public
          3. Returns language to use in narrative
      24. Manmaps
          1. Type: Bool
          2. Permissions: Public
          3. Returns Maneuver maps display options
      25. routeType
          1. Type: Bool
          2. Permissions: Public
          3. Returns the Type of Route
      26. shapeFormat
          1. Type: string
          2. Permissions: Public
          3. Returns shape format option
      27. narrativeType
          1. Type: string
          2. Permissions: Public
          3. Specifies the type of narrative to generate
      28. avoidTripIds
          1. Type: List<int>
          2. Permissions: Public
          3. Returns attribute ids of roads to try and avoid
      29. mustAvoidLinkIds
          1. Type: List<int>
          2. Permissions: Public
          3. Returns attribute ids of roads to try and avoid
      30. tryAvoidLinkId
          1. Type: List<int>
          2. Permissions: Public
          3. Returns link Ids of roads that we will try to avoid during route calculation.
      31. arteryWeights
          1. Type: List<string>
          2. Permissions: Public
          3. Returns attribute ids of roads to try and avoid
   4. Constructors:
      1. Options()
         1. timeType set to 0
         2. maxLinkId set to 0
         3. generalize set to 0
         4. routeNumber set to 0
         5. drivingStyle set to 0
         6. walkingSpeed set to 0
         7. transferPenalty set to 0
         8. maneuverPenalty set to 0
         9. urbanAvoidFactor set to 0
         10. filterZoneFactor set to 0
         11. highwayEfficiency set to 0
         12. cyclingRoadFactor set to 0
         13. maxWalkingDistance set to 0
         14. useTraffic set to false
         15. manmaps set to false
         16. enhancedNarrative set to false
         17. sideOfStreetDisplay set to false
         18. returnLinkDirections set to false
         19. avoidTimedConditions set to false
         20. stateBoundaryDisplay set to false
         21. countryBoundaryDisplay set to false
         22. destinationManeuverDisplay set to false
         23. unit set to String.Empty
         24. locale set to String.Empty
         25. routeType set to String.Empty
         26. shapeFormat set to String.Empty
         27. narrativeType set to String.Empty
         28. avoidTripIds set to new List<int>()
         29. mustAvoidLinkIds set to new List<int>()
         30. tryAvoidLinkIds set to new List<int>()
         31. arteryWeights set to new List<string>()
   5. Functions: No other functions
8. Result.cs
   1. Public Class
   2. Base class required for newtonsoft json parsing.
   3. Attributes:
      1. locations:
         1. Type: List<Locations>
         2. Permissions: Public
         3. Returns a list of results.
   4. Constructors:
      1. LocationsRootObject()
         1. locations set to new instance of List<Locations>
   5. Functions: No Other functions
9. Shape.cs
   1. Public Class
   2. A collection of latitude/longitude coordinates or shape points for the entire route highlight based on original mapstate and/or the generalize option. Shape is an alternated array of lat/lngs. Evens are lat and odds are lng.
   3. Enums:
      1. SFormat
         1. delta: First shape point is represented in mutliples of 106 and subsequent points is given as difference from previous point.
         2. raw: Shape is represented as latitude/longitude pairs.
         3. cmp: Shape is represented as a compressed path string with 5 digits of precision.
         4. cmp6: Same as cmp, but uses 6 digits of precision.
   4. Attributes:
      1. format
         1. Type: SFormat
         2. Permissions: Public
         3. The Shape format options
      2. maneuverIndexes
         1. Type: List<int>
         2. Permissions: Public
         3. An array of the starting index for each maneuver.
      3. shapePoints
         1. Type: List<double>
         2. Permissions: Public
         3. Shape points for the route (clipped and generalized) will be returned if mapState is given. Shape points is an alternated array of lat/lngs. Evens are lat and odds are lng. The index of a specific shape point is i/2.
      4. legIndexes
         1. Type: List<int>
         2. Permissions: Public
         3. The shape point index which starts a specific route segment. The shape point index of the end of the segment is legIndex-1 of the next legIndex. Note: that there is always one extra legIndex (the number of legIndexes = number of legs +1) to account for the last shape point of the final segment.
      5. encodedShape\_
         1. Type: string
         2. Permissions: Private
         3. An option to encode shape to send to api
      6. encodedShape
         1. Type: string
         2. Permissions: Public
         3. An Option to encode shape to send to api
   5. Constructors:
      1. Shape()
         1. format set to SFormat.cmp6
         2. maneuverIndexes set to new instance of List<int>()
         3. shapePoints set to new instance of List<double>()
         4. legIndexes set to new instance of List<int>()
         5. encodedShape\_ set to null
   6. Functions:
      1. returnCompressedPairs
         1. Part of the encoding process.
         2. Parameters:
            1. toBeCompressed: The array of doubles to be compressed
            2. precision: How many digits to compress to.
         3. Returns:
            1. String Encoded Value
      2. encodeNumber
         1. Encodes the number for making the string size smaller.
         2. Parameters:
            1. num: value to encode
         3. Returns:
            1. string encoded value
      3. getString()
         1. This function returns the string nessessary for the static maps API
         2. Returns: String format
10. Sign.cs
    1. Public Class
    2. Sign Class is used in storing data related to directional signs for a particular maneuver. Not peace signs, just road signs and related signs.
    3. Attributes:
       1. type
          1. Type: int
          2. Permissions: Public
          3. Returns the integer value for current said sign.
       2. direction
          1. Type: int
          2. Permissions: Public
          3. Returns the directional integer value for current sign.

Values:

* + - * 1. None = 0
        2. North = 1
        3. Northwest = 2
        4. Northeast = 3
        5. South = 4
        6. Southeast = 5
        7. Southwest = 6
        8. West 7
        9. East = 8
    1. url
       1. Type: string
       2. Permissions: Public
       3. Returns the url of the sign image
    2. directionImage
       1. Type: Image
       2. Permissions: Public
       3. Returns the direction image.
    3. Sign\_
       1. Type: Image
       2. Permissions: private
       3. Used for directionsImage
    4. Text
       1. Type: string
       2. Permissions: Public
       3. Returns text name
    5. extraText
       1. Type: string
       2. Permissions: Public
       3. Returns extra provided text
  1. Constructors:
     1. sign\_ set to null
     2. type set to 0
     3. direction set to 0
     4. url set to String.Empty
     5. text set to String.Empty
     6. extraText set to String.Empty
  2. Functions: No additional functions