Plotting TEI data on a map with Leaflet.js

Magdalena Turska *February 2015*

1 Handy oXygen tricks

Enclose the selection with a tag:

- highlight the characters which you want to tag
- type CTRL+E to display the menu of available tags
- type 'pe' (for `<persName>`) or 'pl' (for `<placeName>`) and then press RETURN

Split the long chunk of text into a sequence of elements of the same kind:

- highlight the long chunk and wrap it with your desired element (say `<p>`)
- move the cursor to a place within that chunk that is the start of next element of the same kind
- type ALT+SHIFT+D to split elements (it inserts closing and starting tags at the cursor position)
- repeat as many times as needed

If you forget the key combination to perform the trick try the right-click and see what's in the Refactoring section

It can help a lot to format and indent your work automatically via CTRL+SHIFT+P or clicking the Format and indent icon.

2 What shall we do today?

In this exercise, we will work with the data from Hamlet encoded in TEI P5 and available from Bodleian First Folio project.

The aim is to:

- extract data in JSON format from TEI file
- learn how to plug in leaflet.js library into html file
- how to plot markers, lines and circles on a map
- load data for leaflet from external file
- create map layers that can be turned on/off separately

3 Places in TEI files

Typically references to places in TEI files are marked as `<name>` or more specifically `<placeName>` elements. In *Hamlet* the latter method was used. The *@ref* on a `<placeName>` points to the `<place>` record in `<teiHeader>`/`<profileDesc>`/`<settingDesc>`/`<listPlace>` that contains `<location>` and its `<geo>` child that wraps geo-coordinates: latitude and longitude of a place.

```
<settingDesc>
 <listPlace>
  <place xml:id="Elsinore">
   <placeName>Elsinore</placeName>
   <location>
    <geo>56.038103 12.620845</geo>
   </location>
  </place>
<!-- ... -->
 </listPlace>
</settingDesc>
```

4 Warm-up with leaflet

If you remember your HTML you can surely create very simple web page that has just one div in its body

```
<html>
 <head>
  <title>Hello map!</title>
 </head>
 <body>
  <div/>
 </body>
</html>
```

To create our first map we need to load Leaflet library first

```
<!-- load leaflet library and corresponding css files --><link rel="stylesheet"
 href="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css"/>
<script src="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.js"/>
```

and define the height and assign an ID to the div that will contain our map to be able to point to it later from leaflet

```
<!-- set id and height for the map container --><div id="map" style="height: 500px;"/>
```

This sort of works and you can try to render your webpage in web browser, but it definitely lacks the map! Leaflet needs so called *tileLayer* or set of graphic files that represent map's background at different zoom levels.

```
<script> var map = new L.Map('map', {
center: new L.LatLng(52.23,21.06),
zoom: 13,
minZoom: 3,
maxZoom: 18
});
L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {}).addTo(map);
</script>
```

Key things to understand here are: the initial center point (center), initial, minimum and maximum zoom levels (zoom, minZoom, maxZoom) and tile setup (L.tileLayer). The latter points to the URL of a set of tile images.

5 Now you try!

Try to play around and center the map on your place of birth. Then how about changing the initial zoom level?
And maybe different tiles?
`http://a.tiles.mapbox.com/v3/isawnyu.map-knmctlkh/{z}/{x}/{y}.png`
`http://services.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/ti`

`http://services.arcgisonline.com/ArcGIS/rest/services/World_Physical_Map/MapServ`

6 Markers

Most probably you'd want to show something more than just a map. So let's add a marker located on Dobra street in Warsaw.

```
 L.marker([52.23,21.06]).addTo(map)
.bindPopup("<b>Oupss!</b>
<br/>Wrong address!").openPopup();
```

This was not too bad, except this is not Dobra! We need better geo-coordinates! How would you go about determining them? Try to add another marker in correct place.

7 Circles and polygons

Adding circles is similarly easy, just note that as a second parameter we need to add radius of the circle

```
var circle = L.circle([52.242,21.024], 500, {
color: 'red',
fillColor: '#f03',
fillOpacity: 0.5
}).addTo(map);
```

Good thing is we have control over color and other format properties of the circle.
We can have this for polygons as well but they need whole series of geocoordinates, not just one poing.

```
var polygon = L.polygon([
[52.242,21.025],
[52.242,21.037],
[52.232,21.024]
], {
color: 'green',
weight: 1,
fillColor: '#0f3',
fillOpacity: 0.5
}).addTo(map);
```

8 Importing data from external file

This is all very well but finding and copying latitudes and longitudes can be quite tiresome... Let's try to store our coordinates in external file and see if this helps.

The file hamlet.js contains just one variable - with table of place objects. It looks like this:

```
var places = { data:[ {"lat": "56.038103", "lon": "12.620845", "title": "Elsinore"
{"lat": "55.9396761", "lon": "9.5155848", "title": "Denmark", "number": "22"},
{"lat": "64.5783089", "lon": "17.888237", "title": "Norway", "number": "10"} ]}
```

Let's worry how this data got there and see how we could plot it on our map. We need to add a piece of code that goes through all elements in places table. Like this:

```
 $.each(places.data, function(idx, place) {
L.marker([place.lat, place.lon], {
color: '#118128', fillColor: '#1bcc3f', weight: 1, opacity: 0.6, fillOpacity: 1
}).addTo(map)
.bindPopup('<strong>' + place.title + ' ' + ' ' + place.number + '</strong>')
.openPopup();
});
```

and one extra javascript library in our header

```
<script type="text/javascript"
 src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min.js"/>
```

9 Extracting data from TEI

Now, how did we get that data from Hamlet?

As (at least some) places referenced in Hamlet were encoded with `<placeName>` elements we can use XSLT to extract this information and convert it to JavaScript table.

First we need to know precisely what we want to extract: we need a list of all places that _have_geocoordinates, for each place we need latitude, longitude, place name and number of marked occurrences of that place in Hamlet.

Second, we need to know where that information is stored in our TEI P5 encoding. As mentioned earlier, all place records go into dedicated section of teiHeader and thus can be retrieved with following XPath expression `/TEI/teiHeader/profileDesc/settingDesc/listPlace/place`. Place name is in `<placeName>` child of a `<place>`. Geocoordinates (if given) can be found in `<location>`/`<geo>`.

Now we need a bit of XSLT to put this all together and in addition to wrap everything as JavaScript objects and tables. Luckily all this can be achieved with relatively simple template.

```
<xsl:template match="/"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> var places = { data:[ <xsl:for-
each select="//tei:place">
  <xsl:variable name="lat"
   select="substring-before(tei:location/tei:geo, ' ')"/>
  <xsl:variable name="lon"
   select="substring-after(tei:location/tei:geo, ' ')"/>
  <xsl:variable name="id"
   select="concat('#',@xml:id)"/>
  <xsl:variable name="entry">
   <xsl:text>{"lat": "</xsl:text>
   <xsl:value-of select="$lat"/>
   <xsl:text>", "lon": "</xsl:text>
   <xsl:value-of select="$lon"/>
```

```
   <xsl:text>", "title": "</xsl:text>
   <xsl:value-of select="tei:placeName"/>
   <xsl:text>", "number": "</xsl:text>
   <xsl:value-of select="count(//tei:placeName[@ref=$id])"/>
   <xsl:text>"}</xsl:text>
   <xsl:if test="position()!=last()">
    <xsl:text>,
    </xsl:text>
   </xsl:if>
  </xsl:variable>
  <xsl:value-of select="$entry"/>
 </xsl:for-each>
 <xsl:text> ]} </xsl:text>
</xsl:template>
```

10 Applying XSLT

You can download complete XSLT stylesheet and run it in oXygen on TEI source file. The output file of the transformation can be saved as hamlet.js we used earlier to feed data into Leaflet.js

For more information about configuring and running transformation scenarios in oXygen try this exercise.

11 Layered map

When map gets too busy, you might want to put different information into separate layer to be turned on/off by the user. Our next example does exactly that and in addition covers also few extra tricks.

Have a look at the source of layeredMap.html. It should look quite familiar but there are differences as well.

- we have more JavaScript libraries and css files plugged in - this is because we will use extra functionality of leaflet.label.js to have hovering labels
- all scripting now goes into html header
- instead of starting with a map (`L.map`) object, we create layers (`L.layerGroup`) and add markers to the layer `placeLayer.addLayer(marker)`
- and map is created last to use previously constructed layers
- special controls need to be added to allow user to turn layers on and off `L.control.layers(...).addTo(map);`

12 Where do you want to go next?

Leaflet website http://leafletjs.com/ especially the examples http://leafletjs.com/examples.html

Open source maps at https://www.mapbox.com/ and their own JavaScript library built on top of Leaflet https://www.mapbox.com/mapbox.js/api/v2.1.5/