FLOWR with Hoax data

Day 02 Session 02 slot 03 Highlevel overview of FLWOR with Hoax data.

Example 1: Introducing places making a fragment with FL(WO)R

```
xquery version "3.1";

(:
 : Introducing places we make a framgment creation with FL(WO)R
 :
 :)

declare namespace hoax = "http://obdurodon.org/hoax";
declare namespace hoax-model = "http://www.obdurodon.org/model";
declare namespace tei = "http://www.tei-c.org/ns/1.0";

declare variable $places-doc :=doc('/db/data/hoax_aux_xml/places.xml');

let $places as element(tei:place)+ := $places-doc//tei:place
return
<hoax-model:places>
{
  for $place in $places
    let $place-name as element(tei:placeName)+ := $place/tei:placeName
    let $geo :=$place/tei:location/tei:geo
    let $lat as xs:string := substring-before($geo, " ")
    let $long as xs:string := substring-after($geo, " ")
    let $parent as element(tei:placeName)? := $place/parent::tei:place/tei:plac

    return
    <placeEntry>
        <placeName>{$place-name ! string()}</placeName>
        <geo>
        <lat>{$lat}</lat>
        <long>{$long}</long>
        </geo>
    <parentPlace>{$parent}</parentPlace>
    </placeEntry>
    }
</hoax-model:places>
```

Example 2: Introducing places making a fragment with FL(WO)RList the URIs of documents in a collection

```
xquery version "3.1";

(:
 : Introducing persons we make a similar framgment creation with FL(WO)R
 :
 :)

declare namespace hoax = "http://obdurodon.org/hoax";
declare namespace hoax-model = "http://www.obdurodon.org/model";
declare namespace tei = "http://www.tei-c.org/ns/1.0";

declare variable $persons-doc := doc('/db/data/hoax_aux_xml/persons.xml');

<hoax-model:persons>
{
for $person in $persons-doc//tei:listPerson/*
  let $surname := $person/tei:persName/tei:surname
  let $forename := $person/tei:persName/tei:forename => string-join(' ')
  let $abt := $person//tei:bibl ! normalize-space(.)
  let $occupation := $person//tei:occupation ! normalize-space(.)
  let $role := $person/@role ! string()
  let $sex := $person/@sex ! string()
  return

    <entry>
        <name>{string-join(($surname, $forename), ', ')}</name>
        <about>{$abt}</about>
        <job>{$occupation}</job>
        <role>{$role}</role>
        <gm>{$sex}</gm>
    </entry>

}
</hoax-model:persons>
```

Example 3: Continuing with places we add a where clause to our FLW(O)R

```
xquery version "3.1";

(:
 : Continuing with places we add a where clause to our FLW(O)R
 :
 :)

declare namespace hoax = "http://obdurodon.org/hoax";
```

```
declare namespace hoax-model = "http://www.obdurodon.org/model";
declare namespace tei = "http://www.tei-c.org/ns/1.0";

declare variable $path-to-data as xs:string := '/db/data';
declare variable $places-doc := doc($path-to-data || '/hoax_aux_xml/places.xml'
declare variable $articles-coll := collection($path-to-data || '/hoax_xml');
declare variable $articles as element(tei:TEI)+ := $articles-coll/tei:TEI;

let $places as element(tei:place)+ := $places-doc/descendant::tei:place
return
<hoax-model:geo-places>
{
  for $entry in $places
    let $place-name as xs:string := $entry/tei:placeName => string-join('; ')
    let $geo :=$entry/tei:location/tei:geo
    let $lat as xs:string := substring-before($geo, " ")
    let $long as xs:string := substring-after($geo, " ")
    let $articles as element(tei:TEI)* :=  $articles[descendant::tei:placeName
                                    [substring-after(@ref, '#') eq $entry/@xml:
                                    (: if using the index in a predicate, you s

    where exists($geo)

    return
      <hoax-model:place>
      <hoax-model:name>{$place-name}</hoax-model:name>
          <hoax-model:geo>
          <hoax-model:lat>{$lat}</hoax-model:lat>
          <hoax-model:long>{$long}</hoax-model:long>
          <hoax-model:articles>{for $article in $articles
                      return <hoax-model:article>{$article/@xml:id, $article/de
          }</hoax-model:articles>
          </hoax-model:geo>
      </hoax-model:place>
}
</hoax-model:geo-places>
```

Example 4: Building on query 03 we add order by to complete a full FLWOR

```
xquery version "3.1";

(:
 : Building on query 03 we add order by to complete a full FLWOR
 :
 :)
```

```
declare namespace hoax = "http://obdurodon.org/hoax";
declare namespace hoax-model = "http://www.obdurodon.org/model";
declare namespace tei = "http://www.tei-c.org/ns/1.0";

declare variable $path-to-data as xs:string := '/db/data';
declare variable $places-doc := doc($path-to-data || '/hoax_aux_xml/places.xml'
declare variable $articles-coll := collection($path-to-data || '/hoax_xml');
declare variable $articles as element(tei:TEI)+ := $articles-coll/tei:TEI;

let $places as element(tei:place)+ := $places-doc/descendant::tei:place
return
<hoax-model:geo-places>
{
  for $entry in $places
    let $place-name as xs:string := $entry/tei:placeName => string-join('; ')
    let $geo :=$entry/tei:location/tei:geo
    let $lat as xs:string := substring-before($geo, " ")
    let $long as xs:string := substring-after($geo, " ")
    let $articles as element(tei:TEI)* :=  $articles[descendant::tei:placeName
                                    [substring-after(@ref, '#') eq $entry/@xml:
                                    (: if using the index in a predicate, you s

    where exists($geo)

    order by $long descending, $place-name

    return
      <hoax-model:place>
      <hoax-model:name>{$place-name}</hoax-model:name>
          <hoax-model:geo>
          <hoax-model:lat>{$lat}</hoax-model:lat>
          <hoax-model:long>{$long}</hoax-model:long>
          <hoax-model:articles>{for $article in $articles
                      return <hoax-model:article>{$article/@xml:id, $article/de
          }</hoax-model:articles>
          </hoax-model:geo>
      </hoax-model:place>
}
</hoax-model:geo-places>
```