Computer Science 571 2nd Exam
Prof. Papa
Thursday, December 6, 2009, 7:00pm – 8:20pm

Name: **Student ID Number:**

1. **This is a closed book exam.**
2. **Please answer all questions on the test**

JSON Question [20 pts]

The REST Flickr Service includes a "getRecent" API, flickr.photos.getRecent, which returns a list of the latest public photos uploaded to flickr. It is defines as follows:

flickr.photos.getRecent

Returns a list of the latest public photos uploaded to flickr.

Authentication

This method does not require authentication.

Arguments

`api_key` (Required)
Your API application key.

`extras` (Optional)
A comma-delimited list of extra information to fetch for each returned record. Currently supported fields are: `license`, `date_upload`, `date_taken`, `owner_name`, `icon_server`, `original_format`, `last_update`, `geo`, `tags`, `machine_tags`, `o_dims`, `views`, `media`, `path_alias`, `url_sq`, `url_t`, `url_s`, `url_m`, `url_o`

`per_page` (Optional)
Number of photos to return per page. If this argument is omitted, it defaults to 100. The maximum allowed value is 500.

`page` (Optional)
The page of results to return. If this argument is omitted, it defaults to 1.

A sample XML REST call is shown below:

http://api.flickr.com/services/rest/?method=flickr.photos.getRecent&api_key=626cf9c993df85b49d193b9645fd2c0d

When the format is XML (the default), the following is an example of the data returned:

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
<photos page="1" pages="100" perpage="2" total="200">
     <photo id="4144809437" owner="9755447@N04"
secret="5d5e5dc80e" server="2582" farm="3" title="Big
Springs Monument Talladega" ispublic="1" isfriend="0"
isfamily="0" />
     <photo id="4144809495" owner="40432260@N04"
secret="2f3880845a" server="2689" farm="3"
title="IMG_1247" ispublic="1" isfriend="0"
isfamily="0" />

</photos>
</rsp>
```

When the “format=JSON”, a JSONP response is returned. A sample JSON REST call is shown below:

A sample XML REST call is shown below:
http://api.flickr.com/services/rest/?method=flickr.photos.getRecent&api_key=626cf9c993df85b49d193b9645fd2c0d&format=json

Please fill in the missing JSON code that duplicates the XML result above:

```
jsonFlickrApi({"photos":{"page":1, "pages":100,
"perpage":2, "total":200,
}, "stat":"ok"})
```

XML Schema Questions [20 pts]

The Yahoo Maps Web Service includes a “Geocoding API” that allows to find the specific longitude and latitude for a given address. The XML Schema for this service response includes the following fields, with the noted descriptions:

Field	Description
ResultSet	Contains all of the query responses.
Result	Contains each individual response. More than one result, **with a maximum of 10**, may be returned if the given address is ambiguous. Has attributes: • **precision**: The precision of the address used for geocoding, from specific street address all the way up to country, depending on the precision of the address that could be extracted. Possible values, from most specific to most general are: ○ address ○ street ○ zip+4 ○ zip+2 ○ zip ○ city ○ state ○ country • **warning**: If the exact address was not found, the closest available match will be noted here. This attribute is **optional.**
Latitude	The latitude of the location.
Longitude	The longitude of the location.
Address	Street address of the result, if a specific location could be determined.
City	City in which the result is located.
State	State in which the result is located.
Zip	Zip code, if known.
Country	Country in which the result is located.

The following is a sample response for the Geocode of the Yahoo! Headquarters:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet          xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="urn:yahoo:maps"
xsi:schemaLocation="urn:yahoo:maps
http://api.local.yahoo.com/MapsService/V1/GeocodeResponse.xsd">
  <Result precision="address">
    <Latitude>37.416384</Latitude>
    <Longitude>-122.024853</Longitude>
    <Address>701 FIRST AVE</Address>
    <City>SUNNYVALE</City>
    <State>CA</State>
    <Zip>94089-1019</Zip>
    <Country>US</Country>
  </Result>
</ResultSet>
```

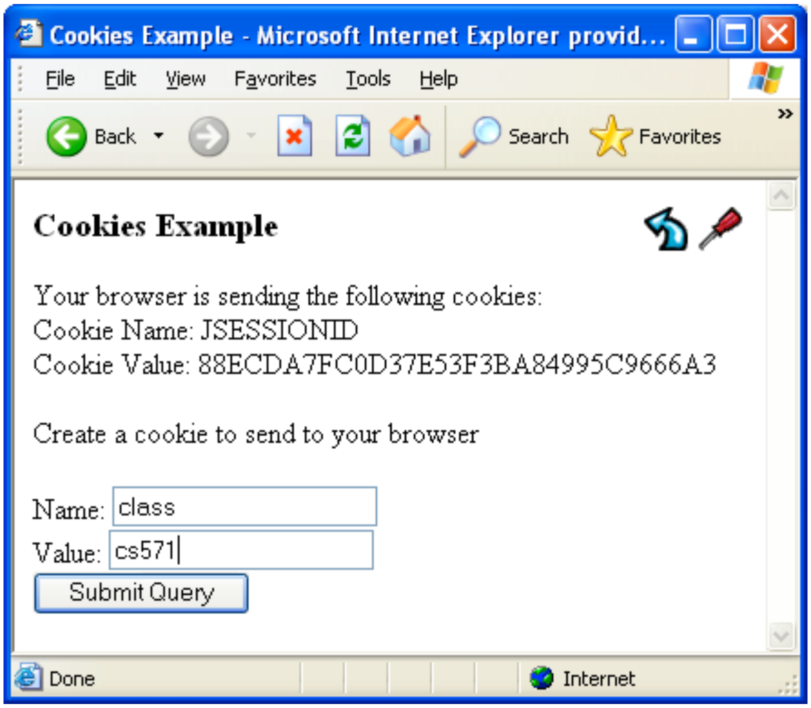
Complete the XML Schema that corresponds to the above description and sample. Fill in the missing parts.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:yahoo:maps"
  xmlns="urn:yahoo:maps"
  elementFormDefault="qualified">

     <xs:element name="ResultSet">



     </xs:element>

     <xs:complexType name="ResultType">
```

```
    </xs:complexType>

</xs:schema>
```

Java Servlet Questions [20 pts]

Below is the initial screen produced by a Java servlet. Its two text fields, representing a cookie name and value to be set, have been filled in. The <form> tag surrounding the two text fields is <form action="CookieExample" method=POST>



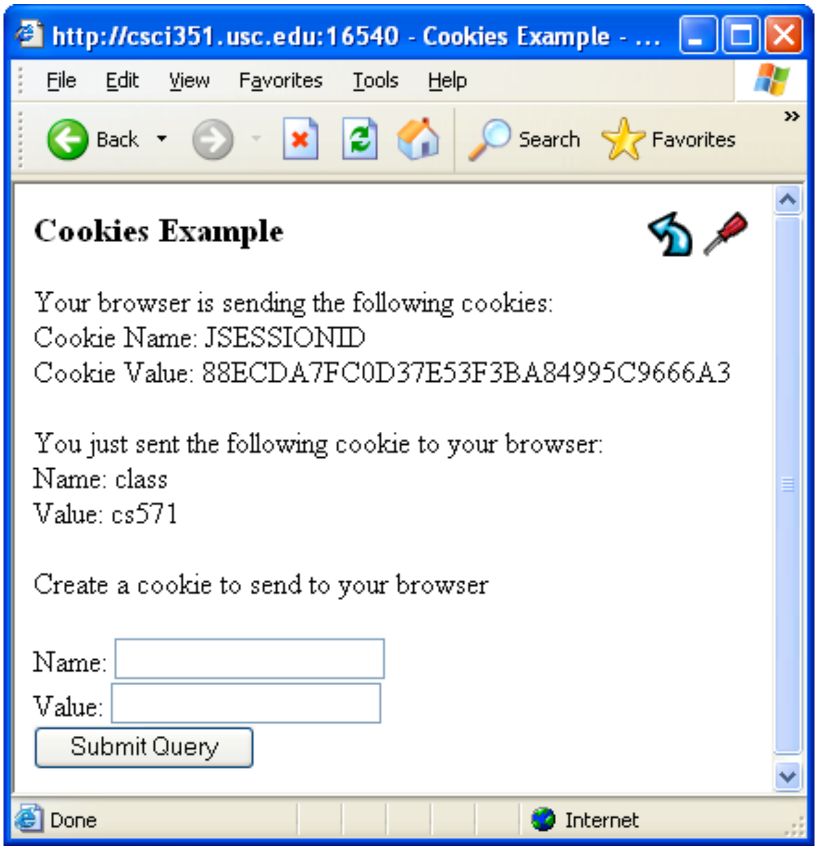
Cookies Example

Your browser is sending the following cookies:
Cookie Name: JSESSIONID
Cookie Value: 88ECDA7FC0D37E53F3BA84995C9666A3

Create a cookie to send to your browser

Name: class
Value: cs571

After clicking on Submit Query the result in the browser is shown below. The parameters that were entered are displayed and a cookie has not been added to the user's browser environment. Two empty text fields are also displayed.



Cookies Example

Your browser is sending the following cookies:
Cookie Name: JSESSIONID
Cookie Value: 88ECDA7FC0D37E53F3BA84995C9666A3

You just sent the following cookie to your browser:
Name: class
Value: cs571

Create a cookie to send to your browser

Name:
Value:
Submit Query

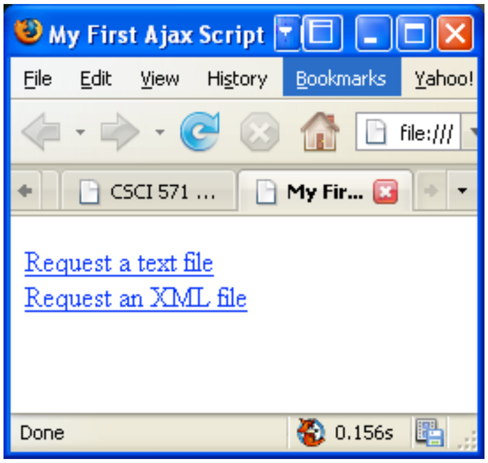
Below is the beginning of the source code that implements the above example. Your task is to add the lines that will complete the program.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class CookieExample extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws IOException, ServletException
    {
        Question 1 [3 pts]                    One line is missing here


        PrintWriter out = response.getWriter();

        // print out cookies
        Question 2 [3 pts]                    One line is missing here


        for (int i = 0; i < cookies.length; i++) {
            Cookie c = cookies[i];
            Question 3 [6 pts]                Two lines are missing here
            out.println(name + " = " + value);
        }

        // set a cookie

        String name = request.getParameter("cookieName");
        if (name != null && name.length() > 0) {
          Question 4 [8 pts]  These 3 missing lines set the cookie




        }
    }
}
```

Web Performance Questions [20 pts]

List 10 out of the 14 rules for faster Web pages from Yahoo's Steve Souders:

JavaScript + AJAX Questions [20 pts]



Below is the source code that generated the web page above. There are two links on the page. The first one causes a text file to be displayed in the page beneath the links. The second link causes an XML file to be displayed in the same place.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html><head>
      <title>First Ajax Script</title>
      <script src="script01.js" type="text/javascript"
language="Javascript">
      </script>
</head><body>
      <p><a id="makeTextRequest" href="gAddress.txt">Request a
text file</a><br />
      <a id="makeXMLRequest" href="us-states.xml">
Request an XML file</a></p>
      <div id="updateArea">&nbsp;</div>
</body>
</html>
```

Below is the JavaScript source that was imported into the HTML above, but some of the lines are missing, replaced by XXXXXXXs. Fill in the missing lines on the answer sheet.

```
window.onload = XXXXXXX1;
                              ______________________________

var xhr = false;
function initAll() {
document.getElementById("makeTextRequest").onclick = getNewFile;
document.getElementById("makeXMLRequest").onclick = getNewFile;}
function getNewFile() {
      makeRequest(this.href);
      return false;}
```

```
function makeRequest(url) {
      if (window. XXXXXXX2) {
                              ______________________________

            xhr = new XXXXXXX3 ();}
                              ______________________________

      else { if (window.ActiveXObject) {
            try { xhr = new ActiveXObject("XXXXXXX4"); }
                              ______________________________

                  catch (e) { }
            }   }
      if (xhr) {
            xhr.onreadystatechange = showContents;
            xhr.open("XXXXXXX5", url, true);
                              ______________________________

            xhr.send(null);  }
      else {
document.getElementById("updateArea").innerHTML = "Sorry, but I
couldn't create an XMLHttpRequest";
    }  }

function XXXXXXX 6() {
                              ______________________________

      if (xhr.readyState == XXXXXXX7) {
                              ______________________________

            if (xhr.status == XXXXXXX8) {
                              ______________________________

                  var outMsg = (
xhr.responseXML && xhr.responseXML.contentType=="text/xml") ?
xhr.responseXML.getElementsByTagName("choices")[0].textContent :
xhr.responseText;
      } else {
           var outMsg = "There was a problem with the request " +
xhr.status; }

document.getElementById("XXXXXXX9").innerHTML = XXXXXXX10;

                              ______________________________

                              ______________________________

} }
```