Midterm – Fall 2016 solution key

Highest: 97
Lowest: 34
Average: 76

Question 1 - i

As a developer of a web application client, which of the following header combinations can you use to ensure you never have to work with stale data from the server, along with ensuring that the client needs to communicate with the server only when required?

(a) If-Match and If-None-Match

(b) Expires, Last-Modified

(c) If-Modified-Since, Last-Modified

(d) Expires, If-Modified-Since, Last-Modified

Correct answer: d

The answer needs to consider two aspects – never work with stale data and server bandwidth is preserved

Never working with stale data will be achieved by combination of If-Modified-Since, and Last-Modified headers.

Preserving server bandwidth will be achieved by requesting data from server opportunistically when the cached data has expired. So, the Expires header is also important.

Question 1 - ii

Suppose the server is maintaining a resource called "index.html" accessible at http://localhost:8080/assignment2/index.html. The contents of index.html is the following line "This is index.html file". The server maintains a ETag for this resource. Suppose the current value of this ETag is “12345”. From the client side suppose following action is executed:

GET http://localhost:8080/assignment2/index.html HTTP/1.1

If-None-Match: “12344”

What will be the output of this call?

This is index.html

or

HTTP/1.1 200 OK

This is index.html

Reason: ETag on server does not match the ETag sent in the If-None-Match header

Question 1 - iii

Consider a University network where all the network traffic goes through a University-wide proxy server which implements caching according to HTTP 1.0 and 1.1. Suppose that a user makes request to resource (r1) and the response sent by the origin server for that resource has a Cache-control header with max-age: 10. Assuming that the proxy server works correctly, answer the following questions

(3 points) Suppose another user makes request to the same resource after two minutes, what will happen?

Because the max-age header value is the duration in seconds for which the response should be cached, the request sent after two minutes would not find the resource in the cache. The resource would need to be fetched from the origin server again.

Question 1 - iii

Suppose a user wants to ensure that the response is only received from the origin server.

How can this be achieved?

Include Cache-control: max-age=0 or pragma: no-cache header in the request

Question 2

What are the difference between getClasses() and getSingletons() methods in Application class in a RESTEasy application?

getClasses() is used to specify the resource and service classes that need to instantiated for every request.

getSingletons() is used to specify the instances/objects of classes whose only single instance needs to be present in a RESTEasy web app.

Question 3

- Unit test 1:

```
@Test
public void testDataExistsInCache() {
 CachingManager mgr = mock(CachingManager.class)
         CachingHTTPClient client = new CachingHTTPClient(mgr);
         when(mgr.checkIfExistsInCache(“url1”)).thenReturn(true);
         when(mgr.getDataFromCache(“url1”)).thenReturn(“ur1data”);
        String response = client.getData(“url1”);
        assertEquals(“url1data”, response);
        verify(mgr).checkIfExistsInCache(true);
        verify(mgr).getDataFromCache(“url1”);
}
```

Question 3

- Unit test 2

```
@Test
public void testDataDoesNotExistInCache () {
 CachingManager mgr = mock(CachingManager.class)
         CachingHTTPClient client = new CachingHTTPClient(mgr);
         when(mgr.checkIfExistsInCache(“url1”)).thenReturn(false);
        String response = client.getData(“url1”);
        assertEquals(“”, response);
        verify(mgr).checkIfExistsInCache(false);
}
```

Question 4 - Servlets

There is problem with the code.

The problem is that the updateCount method is prone to race condition.

Question 5 – REST API

-

Question 6 – Spring framework

```
public class CachingHTTPClient {
   CachingManager mgr;
   public void setMgr(CachingManager mgrRef) {
         this.mgr = mgrRef;
    }
}
<beans>
  <bean name=“cachingClient” class=“CachingHTTPClient”>
     <property name=“mgr” ref=“mgrImpl”/>
  </bean>
  <bean name=“mgrImpl” class=“CachingManager”>
  </bean>
</beans>
```

Question 7 – Functional testing

```
@Test
public void testOutputIsSorted() {
    String method = “GET”;
    String url = “http://localhost:8080/assignment3/myeavesdrop/projects/”
    InputStream is = RESTHelper.getInputStream(method, url);
    List<String> projectList = RESTHelper.getProjectList(is);
    assertTrue(RESTHelper.isSorted(projectList));
}
```

Question 8 – True/False

1. Range HTTP request header allows retrieving a REST resource in parts from a web application/web server.

True

2. The addressability property of HTTP is essential for implementing discoverability (HATEOS) property for REST resources.

True

3. HTTP response codes of 400, 404, 403 can be used interchangeably when designing a REST API.

False

4. When using Spring framework, no Servlet of any kind is involved in handling requests.

False

5. When internal code structure is changed, functional tests need to be updated.

False

Question 8 – True/False

6. The main idea behind cookies is to delegate storage of session state away from server as the server needs to be stateless.

True

7. It is not possible to implement a web application that uses query parameters using Servlets.

False

8. Functional testing is focused on testing the functional behavior of methods within the code.

False

9. Writing a functional test involves unmarshalling the response received from REST API.

True

10. Dividing code into different layers (resource/controller and service) is important from unit testing point-of-view, whereas defining a stable REST API is important from functional testing point-of-view.

True