Newswhere Clarifications

<u>Client-side GUI</u>

- **Minimal requirements for full credit**
    - Textbox in the frame GUI (which will hold the MapWidget component) where you can enter <u>one</u> URL for a request. Client forms XML out of this single request and sends that to server.
    - Textbox to input an already formed XML file on the disk. The client simply sends that file to the server
- **Extra Credit**
    - Format your GUI so that you can dynamically add or remove multiple URLs to the request.
    - Have the client do a crawl on the given URLs. That means the client will load up the page, search for links there, and accumulate them to the XML request.
- **Other notes**
    - MapWidget.update() will *overwrite* the current data in the widget.
    - You have to accumulate probabilities from multiple responses by adding the probabilities for each repeating country and normalizing it (the total should always be 1.0)

<u>Server-side</u>

- Thread pools should be used to get full credit, even if you're not multi-threading page requests (discussed below).
- Make sure request time-outs work with the thread pools correctly. When the time-out expires, the server should send the result (whatever it has at that point), and not wait for any other processes to finish. If a thread is in the middle of processing and the time-out expires, you do not have to worry about interrupting that thread, as long as you send the response immediately. Also, make sure that references to threads are removed, so that they **are** garbage-collected.
- HTML parsing is <u>not</u> necessary. You can just treat the whole HTML page as a long string.
- It's acceptable to do brute-force string comparison in your AI. However, if it takes over 30 seconds to process a single page, then you're doing something wrong.
- **Extra Credit**
    - Multi-threading each page request per client. If you do this, definitely use thread pools as well. If the time-out expires, that connection should send back whatever response it has immediately.