ASYNC RESTFUL API CALL, JAVA, NOTES:

|  |
| --- |
| <http://stackoverflow.com/questions/3142915/how-do-you-create-an-asynchronous-http-request-in-java> |
| Summary: HTTP calls are SYNCRONOUS by nature, you have to build the ASYNC on top of that. You can do that by opening a new thread for the call. There are libraries that handle async HTTP requests. However, if they use this implementation, or something else, I do not know. |
| JAXRS, RxJava, HyStrix, WebHooks |

|  |
| --- |
| Concurrency Tutorial: http://docs.oracle.com/javase/tutorial/essential/concurrency/ |
| If you read it, you will know how to make an asynchronous HTTP GET call. |

Sprig Async Notes:

|  |
| --- |
| SOURCE:  https://spring.io/guides/gs/async-method/  JMIM NOTE:  Great that we have multiple tasks running on sub-threads.  HOWEVER, sleeping in a while loop and periodically checking to  see if they are done? That is some moronic behaviour there. |
| |  |  | | --- | --- | | Cannot use a promise design pattern:  // Callback approach  async1(function(){  async2(function(){  async3(function(){  ....  });  });  }); | Because Java has **no function types** like JavaScript/AS3. However, maybe I could make an "**Async Check List**" Object. An array of <**Future**> type objects. And each time a future Object completes, it **marks itself off the check-list**, and then the **current state of the check-list is evaluated.** | | //T == true, F== false.  private Boolean areProcessesDone()  {  if(fn1.get() != T){ return F;}  if(fn2.get() != T){ return F;}  if(fn3.get() != T){ return F;}  return T;  } | //This would make more sense than SLEEPING THE MAIN THREAD!  //WHAT THE HELL!  The spring example code sleeps the main thread until all these processes are done. | |

|  |
| --- |
| Spring Async Example with idoitic choice of sleeping the main thread: Unless sleeping the main thread does NOT lead to an unresponsive program... This is just ridiculous. We need callbacks. |
| package hello;  import java.util.concurrent.Future;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.boot.CommandLineRunner;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.scheduling.annotation.EnableAsync;  @SpringBootApplication  @EnableAsync  public class Application implements CommandLineRunner {  @Autowired  FacebookLookupService facebookLookupService;  @Override  public void run(String... args) throws Exception {  // Start the clock  long start = System.currentTimeMillis();  // Kick of multiple, asynchronous lookups  Future<Page> page1 = facebookLookupService.findPage("PivotalSoftware");  Future<Page> page2 = facebookLookupService.findPage("CloudFoundry");  Future<Page> page3 = facebookLookupService.findPage("SpringFramework");  // Wait until they are all done  while (!(page1.isDone() && page2.isDone() && page3.isDone())) {  **Thread.sleep(10);** //10-millisecond pause between each check  }  // Print results, including elapsed time  System.out.println("Elapsed time: " + (System.currentTimeMillis() - start));  System.out.println(page1.get());  System.out.println(page2.get());  System.out.println(page3.get());  }  public static void main(String[] args) {  SpringApplication.run(Application.class, args);  }  } |

According to this article: Future == Promise, approximately synonamous.  
http://blog.ometer.com/2011/07/24/callbacks-synchronous-and-asynchronous/  
  
Read up more on “future” object. Get your async ping-pong working sunday.  
http://www.nurkiewicz.com/2013/02/javautilconcurrentfuture-basics.html