

Industrial strength Java/JEE Career Companion to open more doors

search here ...

Go

[Home](#)[Java FAQs](#)[600+ Java Q&As](#)[Career](#)[Tutorials](#)[Member](#)[Why?](#)[Can u Debug?](#)[Java 8 ready?](#)[Top X](#)[Productivity Tools](#)[Judging Experience?](#)[Home](#) > [Interview](#) > [Java Architecture Interview Q&A](#) > 1. Asynchronous processing in Java real life examples – part-1

1. Asynchronous processing in Java real life examples – part-1

Posted on [September 2, 2015](#) by [Arulkumaran Kumaraswamipillai](#)

Tweet



Share

The scenario based questions are very popular with the job interviewers, and some scenario based questions are related to decoupling and **asynchronous** (aka **non-blocking**) processing in Java with message queues/topics, database tables where one process inserts data and another process consumes the inserted data, and **Future** objects in Java multi-threading.

Q1. Can you describe instances where you used asynchronous processing in your application?

A1. The asynchronous processing is very handy for decoupling systems and for boosting performance and scalability. Here are a few examples.

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)[Ice Breaker Interview](#)[Core Java Interview C](#)[JEE Interview Q&A \(3](#)[Pressed for time? Jav](#)[SQL, XML, UML, JSC](#)[Hadoop & BigData Int](#)[Java Architecture Inte](#)[♥♦ 01: 30+ Writing](#)[001A: ♦ 7+ Java int](#)[001B: ♦ Java archil](#)[01: ♥♦ 40+ Java W](#)[02: ♥♦ 13 Tips to w](#)[03: ♦ What should l](#)[04: ♦ How to go ab](#)[05: ETL architectur](#)[1. Asynchronous pi](#)[2. Asynchronous pi](#)[Scala Interview Q&As](#)[Spring, Hibernate, & I](#)[Testing & Profiling/Sa](#)[Other Interview Q&A 1](#)

Example 1: Performing time consuming tasks asynchronously

















An online application with a requirement to produce time consuming reports or a business process (e.g. rebalancing accounts, aggregating hierarchical information, producing compliance reports that runs through hundreds of rules, etc) could benefit from making these long running operations asynchronous. Once the reports or the long running business process is completed, the outcome can be communicated to the user via emails or asynchronously refreshing the web page via techniques known as “**server push**” (i.e via Servlet 3.0 asynchronous) or “**client pull**” (i.e. meta refresh tag). A typical example would be

- a) A user makes a request for an aggregate report or a business process like rebalancing his/her portfolios.
- b) The user input can be saved to a database table for a separate process to periodically pick it up and process it asynchronously.
- c) The user could now continue to perform other functionalities of the website without being blocked.
- d) A separate process running on the same machine or different machine can periodically scan the table for any entries and produce the necessary reports or execute the relevant business process. This could be a scheduled job that runs once during off-peak or every 10 minutes. This depends on the business requirement.
- e) Once the report or the process is completed, notify the user via emails or making the report available online to be downloaded.

Example 2: RESTFul Web Service utilizing store, play and replay asynchronously mechanism








16 Technical Key Areas

[open all](#) | [close all](#)

-  [Best Practice \(6\)](#)
-  [Coding \(26\)](#)
-  [Concurrency \(6\)](#)
-  [Design Concepts \(7\)](#)
-  [Design Patterns \(11\)](#)
-  [Exception Handling \(3\)](#)
-  [Java Debugging \(21\)](#)
-  [Judging Experience \(1\)](#)
-  [Low Latency \(7\)](#)
-  [Memory Management \(1\)](#)
-  [Performance \(13\)](#)
-  [QoS \(8\)](#)
-  [Scalability \(4\)](#)
-  [SDLC \(6\)](#)
-  [Security \(13\)](#)
-  [Transaction Management \(1\)](#)

80+ step by step Java Tutorials

[open all](#) | [close all](#)

-  [Setting up Tutorial \(6\)](#)
-  [Tutorial - Diagnosis \(2\)](#)
-  [Akka Tutorial \(9\)](#)
-  [Core Java Tutorials \(2\)](#)
-  [Hadoop & Spark Tutorial \(1\)](#)
-  [JEE Tutorials \(19\)](#)
-  [Scala Tutorials \(1\)](#)
-  [Spring & Hibernate Tutorials \(1\)](#)
-  [Tools Tutorials \(19\)](#)
-  [Other Tutorials \(45\)](#)

Define two RESTful web services that take payloads and process them asynchronously. This means the web service requests are not blocked.

1) The “submitCreateCase” stores the payload in a database table for auditing and replaying on failure purposes. This service is invoked by a user via a GUI when submitting case details (i.e. submit some data) that needs to be posted via a third-party web service. If invocation of the third-party web service fails, the status gets updated on the table with the payload as “FAILED”, and can be retried via

“replayFailedRequests” service.

2) The “replayFailedRequests” service to retry the payload with status being “FAILED” in the database table. This service gets periodically invoked by a scheduled job running say every 15 minutes to re-invoke the third-party RESTful web service by retrieving the saved payload from the database table.

Here is some sample code snippets:

```

1  //...package & imports
2
3  @Controller
4  @RequestMapping(value = "/v1/myapp/", produces =
5  public class MyAppEndpointControllerImpl implements
6
7      @Inject
8      @Named("createCaseExecutor")
9      private Executor createCaseExecutor;
10
11     @Inject
12     @Named("replayExecutor")
13     private Executor replayExecutor;
14
15     @Inject
16     private MyAppReplayService replayService;
17
18     @Inject
19     private MyAppClientService myAppClientService;
20
21     @Inject
22     private UniqueIdentifierGeneratorService uidGeneratorService;
23
24     private static final Logger LOG = LoggerFactory.getLogger(MyAppEndpointControllerImpl.class);
25
26     //invoked via a User Interface by a user submit case
27     @Override
28     @ResponseStatus(HttpStatus.ACCEPTED)
29     @RequestMapping(value = "/submitCase", method = RequestMethod.POST)

```

100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(1\)](#)
- [Complete the given code](#)
- [Converting from A to B](#)
- [Designing your classes](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with this code?](#)
- [Writing Code Home Assignment](#)
- [Written Test Core Java](#)
- [Written Test JEE \(1\)](#)

How good are you ..?

[open all](#) | [close all](#)

- [Career Making Knowledge](#)
- [Job Hunting & Resumes](#)

```

30 @ResponseBody public CreateCaseResponseAsync
31 @RequestBody final CreateCaseRequestAsync
32
33     final String auditTrackingId = uidGenera
34
35     //spawn a new thread
36     createCaseExecutor.execute(new Runnable(
37         @Override
38         public void run() {
39             LOG.debug("submitCreateCase() in
40             final CreateCaseRequest createCa
41             //store request payload, status,
42             //invoke third-party service asy
43             myAppClientService.savePayload(c
44             callbackProcessor.processCallbac
45         });
46     });
47
48     // return the DTO including the auditTra
49     final CreateCaseResponseAsync createCase
50     createCaseResponseAsync.setAuditTracking
51     return createCaseResponseAsync;
52 }
53
54 //invoked by a scheduled job every 15 minute
55 @Override
56 @ResponseStatus(HttpStatus.ACCEPTED)
57 @RequestMapping(value = "/replay", method =
58 public void replayFailedRequests(
59     @RequestParam(required = false) final In
60
61
62     //the request payload and statuses are s
63     //spawn a new thread
64     replayExecutor.execute(new Runnable() {
65         @Override
66         public void run() {
67             LOG.info("replayFailedRequests()
68             replayService.replayFailedReques
69         });
70     });
71 }
72 }

```

The executors can be configured via Java based Spring configs.

```

1
2 @Bean
3 public ExecutorService replayExecutor() {
4     return Executors.newSingleThreadExecutor
5 }
6
7 @Bean
8 public ExecutorService createCaseExecutor()
9     return Executors.newCachedThreadPool();
10 }
11

```

The callbackProcessor that creates a new nested transaction, and makes the actual third-party REST call.

```
1 @Named
2 @Transactional
3 public class MyAppCallbackProcessorImpl implements
4     private static final Logger LOG = LoggerFactory
5
6     private final RestTemplate restTemplate;
7     private final TransactionService transactionService;
8     private HttpHeaders httpHeaders;
9
10
11 @Inject
12 public MyAppCallbackProcessorImpl(final RestTemplate restTemplate,
13     this.restTemplate = restTemplate;
14     this.transactionService = transactionService;
15 }
16
17 @Override
18 public void processCallback(final String callbackUrl) {
19     try {
20         transactionService.executeInNewTransaction(() -> {
21             @Override
22             public Void doInTransaction() throws Exception {
23                 sendCreateCaseRequest(callbackUrl);
24                 return null;
25             }
26         });
27     } catch (final Exception ex) {
28         LOG.error(String.format("An error has occurred while processing callback: %s", callbackUrl), ex);
29     }
30 }
31
32 private void sendCreateCaseRequest(final String callbackUrl) {
33     if (StringUtils.isNotBlank(callbackUrl)) {
34         LOG.info("Sending PUT request to MyAPI at " + callbackUrl);
35         final HttpEntity<CreateCaseRequest> request = new HttpEntity<CreateCaseRequest>(
36             new CreateCaseRequest(), httpHeaders);
37         restTemplate.put(URI.create(callbackUrl), request);
38     }
39 }
40
41 @Override
42 public void afterPropertiesSet() throws Exception {
43     httpHeaders = new HttpHeaders();
44     httpHeaders.setAccept(ImmutableList.of(MediaType.APPLICATION_JSON));
45     httpHeaders.setContentType(MediaType.APPLICATION_JSON);
46 }
47 }
48 }
```

More asynchronous processing in Java examples part 2.

Popular Member Posts

◆ 11 Spring boot interview questions & answers

850 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

768 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

399 views

18 Java scenarios based interview Questions and Answers

387 views

♦ 7 Java debugging interview questions & answers

308 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

305 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

297 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

293 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

246 views

001B: ♦ Java architecture & design concepts interview questions & answers

204 views

Bio

Latest Posts



Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



**About** [Arulkumaran Kumaraswamipillai](#)

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](#) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ 03: Identifying and fixing NonUniqueObjectException in Hibernate

04: Identifying and fixing LazyInitializationException in Hibernate ▶

Posted in Java Architecture Interview Q&A, Judging Experience Interview Q&A, member-paid

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)

☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

Prepare to succeed

☀ [Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ \[How to prepare for Java job interviews?\]\(#\) ☀ \[16 Technical Key Areas\]\(#\) ☀ \[How to choose from multiple Java job offers?\]\(#\)](#)

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.