HowToDoInJava

Java 8

Regex

Concurrency

Best Practices

Spring Boot

JUnit5

Interview Questions



Spring Boot REST

SB REST – Hello World

SB REST – @RestController

SB REST - JSON

SB REST - POST with

Headers

SB REST - HATEOAS

SB REST - @Async

SB REST - SseEmitter

SB REST -

ResponseBodyEmitter

SB REST - Callable

SB REST - Unit Testing

SB REST – Gzip Compression

SB REST - i18n

Spring Boot 2 Tutorial

Spring Boot – Introduction

Spring Boot – Starter parent

Spring Boot - Starter

templates

Spring Boot – Multi-module project

Spring @Async rest controller example – Spring @EnableAsync

By Sudip Roy Chowdhury | Filed Under: Spring Boot REST

Learn to create asynchronous controller methods in Spring framework with the help of @Async and @EnableAsync annotations, async thread pool on top of Java ExecutorService framework.

1. Spring @Async rest controller

Spring comes with <code>@EnableAsync</code> annotation and can be applied on application classes for asynchronous behavior. This annotation will look for methods marked with <code>@Async</code> annotation and run in background thread pools. The <code>@Async</code> annotated methods can return <code>CompletableFuture</code> to hold the result of an asynchronous computation.

To enable async configuration in spring, follow these steps:

1. Create async thread pool

Search Tutorials

Type and Press ENTER...

```
Spring Boot – Annotations
Spring Boot - Auto
configuration
Spring Boot – AOP
Spring Boot – Logging
Spring Boot – DevTools
Spring Boot – WAR
Packaging
Spring Boot – REST API
Spring Boot - CRUD
Spring Boot – OAuth2
Spring Boot – Testing
Spring Boot – RestTemplate
Spring Boot – Thymeleaf
Spring Boot – Hibernate
Spring Boot – DataSource
Spring Boot – Error Handling
Spring Boot – Caching
Spring Boot – Retry
Spring Boot – BasicAuth
Spring Boot – H2 Database
Spring Boot – Ehcache 3.x
Spring Boot – Gson
Spring Boot – RMI
Spring Boot - Send Email
Spring Boot – Interview
Questions
```

Spring Boot Tutorial

```
AsyncConfiguration.java

@Configuration
@EnableAsync
public class AsynchConfiguration
{
    @Bean(name = "asyncExecutor")
    public Executor asyncExecutor()
    {
        ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecuto
            executor.setCorePoolSize(3);
            executor.setMaxPoolSize(3);
            executor.setQueueCapacity(100);
            executor.setThreadNamePrefix("AsynchThread-");
            executor.initialize();
            return executor;
        }
}
```

2. @Async controller methods

Methods which shall run asynchronously, annotate them with <code>@Async</code> annotation and method return type should return

```
@Async("asyncExecutor")
public CompletableFuture<EmployeeNames> methodOne() throws Interrup
    //code
}
```

3. Combine async method results

```
Spring Boot -
CommandLineRunner
Spring Boot – Configure Jetty
Spring Boot – Tomcat Default
Port
Spring Boot – Context Root
Spring Boot – SSL [https]
Spring Boot – Get all loaded
beans
Spring Boot – PropertyEditor
Spring Boot -
@EnableScheduling
Spring Boot - Jersey
Spring Boot - SOAP
Webservice
Spring Boot – SOAP Client
Spring Boot – JMSTemplate
Spring Boot - REST APIs
Spring Boot – JSP View
Spring Boot - Actuator
endpoints
Spring Boot - Role Based
Security
Spring Boot – RSS / ATOM
Feed
Spring Boot – Ehcache 2.x
```

Popular Tutorials

Java 8 Tutorial
Core Java Tutorial
Collections in Java

```
Inside REST Controller

CompletableFuture.allOf(methodOne, methodTwo, methodThree).join();
```

2. Spring @Async rest controller example

In this demo, we will create an REST API which will fetch data from three (3) remote services asynchronously and when responses from all 3 services is available then aggregate the responses. e.g.

- 1. Invoke EmployeeName API
- 2. Invoke EmployeeAddress API
- 3. Invoke EmployeePhone API
- 4. Wait for responses from above services
- 5. Aggregate all three API responses and build final response to send back to client

2.1. EmployeeName, EmployeeAddress and EmployeePhone APIs to be accessed async way

```
package com.howtodoinjava.example.sampleservice.controller;

import java.util.ArrayList;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

Java Concurrency
Java Date and Time
Spring Boot Tutorial
Spring AOP Tutorial
Spring MVC Tutorial
Spring Security Tutorial
Hibernate Tutorial
Python Tutorial
Jersey Tutorial
Maven Tutorial
Log4j Tutorial
Regex Tutorial

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.howtodoinjava.example.sampleservice.model.EmployeeAddress;
import com.howtodoinjava.example.sampleservice.model.EmployeeAddresses;
import com.howtodoinjava.example.sampleservice.model.EmployeeName;
import com.howtodoinjava.example.sampleservice.model.EmployeeNames;
import com.howtodoinjava.example.sampleservice.model.EmployeePhone;
@RestController
public class EmployeeDataController
    private static Logger log = LoggerFactory.getLogger(EmployeeDataCont)
    @RequestMapping(value = "/addresses", method = RequestMethod.GET)
    public EmployeeAddresses getAddresses()
        log.info("get addresses Start");
        EmployeeAddresses employeeAddressesList = new EmployeeAddresses(
        EmployeeAddress employeeAddress1 = new EmployeeAddress();
        EmployeeAddress employeeAddress2 = new EmployeeAddress();
        List<EmployeeAddress> addressList = new ArrayList<EmployeeAddress</pre>
            employeeAddress1.setHouseNo("1111");
            employeeAddress1.setStreetNo("111");
            employeeAddress1.setZipCode("111111");
            employeeAddress2.setHouseNo("222");
            employeeAddress2.setStreetNo("222");
            employeeAddress2.setZipCode("222222");
            addressList.add(employeeAddress1);
            addressList.add(employeeAddress2);
```

```
employeeAddressesList.setEmployeeAddressList(addressList);
    }
    return employeeAddressesList;
@RequestMapping(value = "/phones", method = RequestMethod.GET)
public EmployeePhone getPhoneNumbers()
    log.info("get phones Start");
    EmployeePhone employeePhone = new EmployeePhone();
        ArrayList<String> phoneNumberList = new ArrayList<String>();
        phoneNumberList.add("100000");
        phoneNumberList.add("200000");
        employeePhone.setPhoneNumbers(phoneNumberList);
    return employeePhone;
@RequestMapping(value = "/names", method = RequestMethod.GET)
public EmployeeNames getEmployeeName()
    log.info("get names Start");
    EmployeeNames employeeNamesList = new EmployeeNames();
    EmployeeName employeeName1 = new EmployeeName();
    EmployeeName employeeName2 = new EmployeeName();
    List<EmployeeName> employeeList = new ArrayList<EmployeeName>();
        employeeName1.setFirstName("Santa");
        employeeName1.setLastName("Singh");
```

```
{
    employeeName2.setFirstName("Banta");
    employeeName2.setLastName("Singh");
}

employeeList.add(employeeName1);
    employeeList.add(employeeName2);

employeeNamesList.setEmployeeNameList(employeeList);

return employeeNamesList;
}
}
```

2.2. Async thread pool configuration

```
import java.util.concurrent.Executor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

@Configuration
@EnableAsync
public class AsyncConfiguration
{
    @Bean(name = "asyncExecutor")
    public Executor asyncExecutor() {
        ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
        executor.setCorePoolSize(3);
        executor.setQueueCapacity(100);
```

```
executor.setThreadNamePrefix("AsynchThread-");
    executor.initialize();
    return executor;
}
```

2.3. Spring @Async controller methods

```
AsyncService.java
package com.howtodoinjava.example.async.service;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import com.howtodoinjava.example.async.model.EmployeeAddresses;
import com.howtodoinjava.example.async.model.EmployeeNames;
import com.howtodoinjava.example.async.model.EmployeePhone;
@Service
public class AsyncService {
    private static Logger log = LoggerFactory.getLogger(AsyncService.clas
    @Autowired
    private RestTemplate restTemplate;
    @Bean
```

```
public RestTemplate restTemplate() {
    return new RestTemplate();
@Async("asyncExecutor")
public CompletableFuture<EmployeeNames> getEmployeeName() throws Intelligence
    log.info("getEmployeeName starts");
    EmployeeNames employeeNameData = restTemplate.getForObject("http
    log.info("employeeNameData, {}", employeeNameData);
                         //Intentional delay
    Thread.sleep(1000L);
    log.info("employeeNameData completed");
    return CompletableFuture.completedFuture(employeeNameData);
@Async("asyncExecutor")
public CompletableFuture<EmployeeAddresses> getEmployeeAddress() threat
    log.info("getEmployeeAddress starts");
    EmployeeAddresses employeeAddressData = restTemplate.getForObject
    log.info("employeeAddressData, {}", employeeAddressData);
    Thread.sleep(1000L);
                         //Intentional delay
    log.info("employeeAddressData completed");
    return CompletableFuture.completedFuture(employeeAddressData);
@Async("asyncExecutor")
public CompletableFuture<EmployeePhone> getEmployeePhone() throws In:
    log.info("getEmployeePhone starts");
    EmployeePhone employeePhoneData = restTemplate.getForObject("http
    log.info("employeePhoneData, {}", employeePhoneData);
    Thread.sleep(1000L);
                          //Intentional delay
```

```
log.info("employeePhoneData completed");
    return CompletableFuture.completedFuture(employeePhoneData);
}
```

2.4. Call async methods and aggregate results

```
package com.howtodoinjava.example.async.controller;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.howtodoinjava.example.async.model.EmployeeAddresses;
import com.howtodoinjava.example.async.model.EmployeeNames;
import com.howtodoinjava.example.async.model.EmployeePhone;
import com.howtodoinjava.example.async.service.AsyncService;
@RestController
public class AsyncController {
    private static Logger log = LoggerFactory.getLogger(AsyncController.
    @Autowired
    private AsyncService service;
    @RequestMapping(value = "/testAsynch", method = RequestMethod.GET)
    public void testAsynch() throws InterruptedException, ExecutionExcept
```

2.5. How to run the demo

Download and start both the applications.

Hit the API: http://localhost:8081/testAsynch.

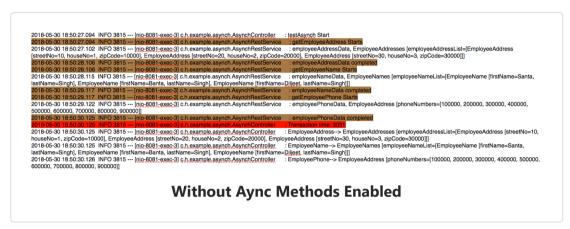
Observe the output in console.

2.5.1. With @Aync Enabled

```
2018-05-30 18.37:38.783 INFO 3802 --- [nio-8081-exec-3] c.h. example.asynch.AsynchController 2018-05-30 18.37:38.783 INFO 3802 --- [AsynchThread-1] c.h. example.asynch.AsynchRestService 2018-05-30 18.37:38.784 INFO 3802 --- [AsynchThread-3] c.h. example.asynch.AsynchRestService 2018-05-30 18.37:38.784 INFO 3802 --- [AsynchThread-2] c.h. example.asynch.AsynchRestService 2018-05-30 18.37:38.00 INFO 3802 --- [AsynchThread-2] c.h. example.asynch.AsynchRestService 2018-05-30 18.37:38.00 INFO 3802 --- [AsynchThread-3] c.h. example.asynch.AsynchRestService 2018-05-30 18.37:39.80 INFO 3802 --- [Inio-9801-exac-3] c.h. example.asynch.AsynchRestService 2018-05-30 18.37:39.80 INFO 3802 --- [Inio-9801-exac-3] c.h. example.asynch.AsynchController 2018-05-30 18.37:39.80 INFO 3802 --- [Inio-9801-exac-3] c.h. example.asynch.AsynchController 2018-05-30 18.37:39.80 INFO 3802 --- [Inio-9801-exac-3] c.h. example.asynch.AsynchController 2018-05-30 18.37:39.80 INFO 3802 -- [Inio-9801-exac-3] c.h. example.asynch.AsynchController 2018-05-30 18
```

With Aync Methods Enabled

2.5.2. Without Aync Enabled



Spring Async Service Sourcecode

Employee Data Service Sourcecode

Drop me your questions related to creating spring boot non blocking rest api.

Happy Learning!!

References:

https://spring.io/guides/gs/async-method/

⁹ Leave a Reply

Join the discussion...

=6 **≥**3 **3**0 **→ ♦**

9

☑ Subscribe ▼

▲ newest ▲ oldest ▲ most voted

sagar gaikwad



If we uses same port number for /testAsynch as well as for /address . i have websphere application server and i am using portnumber 9080 for testAsynch and for address what should i used

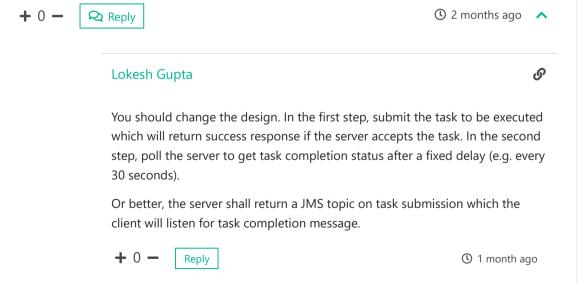


Deepak Kumar



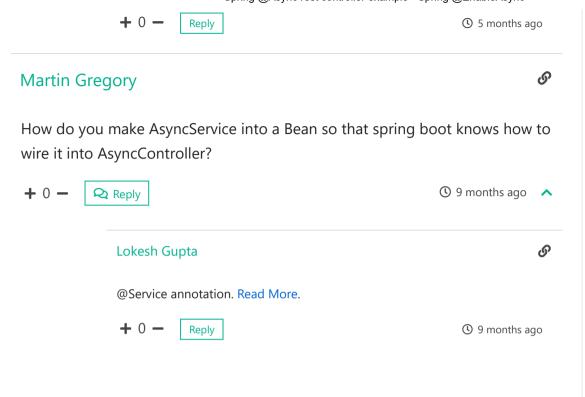
டு

Can you tell me how to extend Session timed out. i have one API it is working fine when it in local Machine but after deployment i am getting error Session and connection timeout it is happening because if you hit any https request the api should return response within 60 seconds but my Api is taking 5-10 minutes to execute task. so could you guide me how to increase this time limit. I am using Spring Boot 2.0 and Java 8



Hema

Is it efficient to use the default task executor without having to configure one? You have asyncExecutor with certain config on thread pool numbers and so. I was not sure what values to specify that works best for our app. **+** 0 **-**3 months ago **Reply** 6 Sumanth Sir, What if one of the async call throws Exception? © 5 months ago Reply Reply G Jagadish Kurli Hi, Was Wondering when exactly we should use Async? If i understand correctly, this async is used in use cases where the user doesn't want to wait for the response or the user wants quick response, by making synchronous calls in order to achieve one big response. But i was wondering if we can achieve retrying mechanism like with we do with JMS through this? ⑤ 5 months ago ∧ **Reply** G Lokesh Gupta Async is useful when there are multiple UI components in screen which can update separately. For example, admin dashboards.



Meta Links

Advertise Contact Us Privacy policy About Me

Recommended Reading

10 Life Lessons
Secure Hash Algorithms
How Web Servers work?
How Java I/O Works Internally?
Best Way to Learn Java
Java Best Practices Guide
Microservices Tutorial
REST API Tutorial
How to Start New Blog

Copyright © 2016 · HowToDoInjava.com · All Rights Reserved. | Sitemap