Spring ThreadPoolTaskExecutor with Callable interface for concurrency programming

Callable is a Java interface that is an useful methodology for control multi-threading returns on concurrency development.

The tutorial will guide you to build a concurrency program with **ThreadPoolTaskExecut** of **Spring Framework** and **Callable interface**.

Contents [hide]

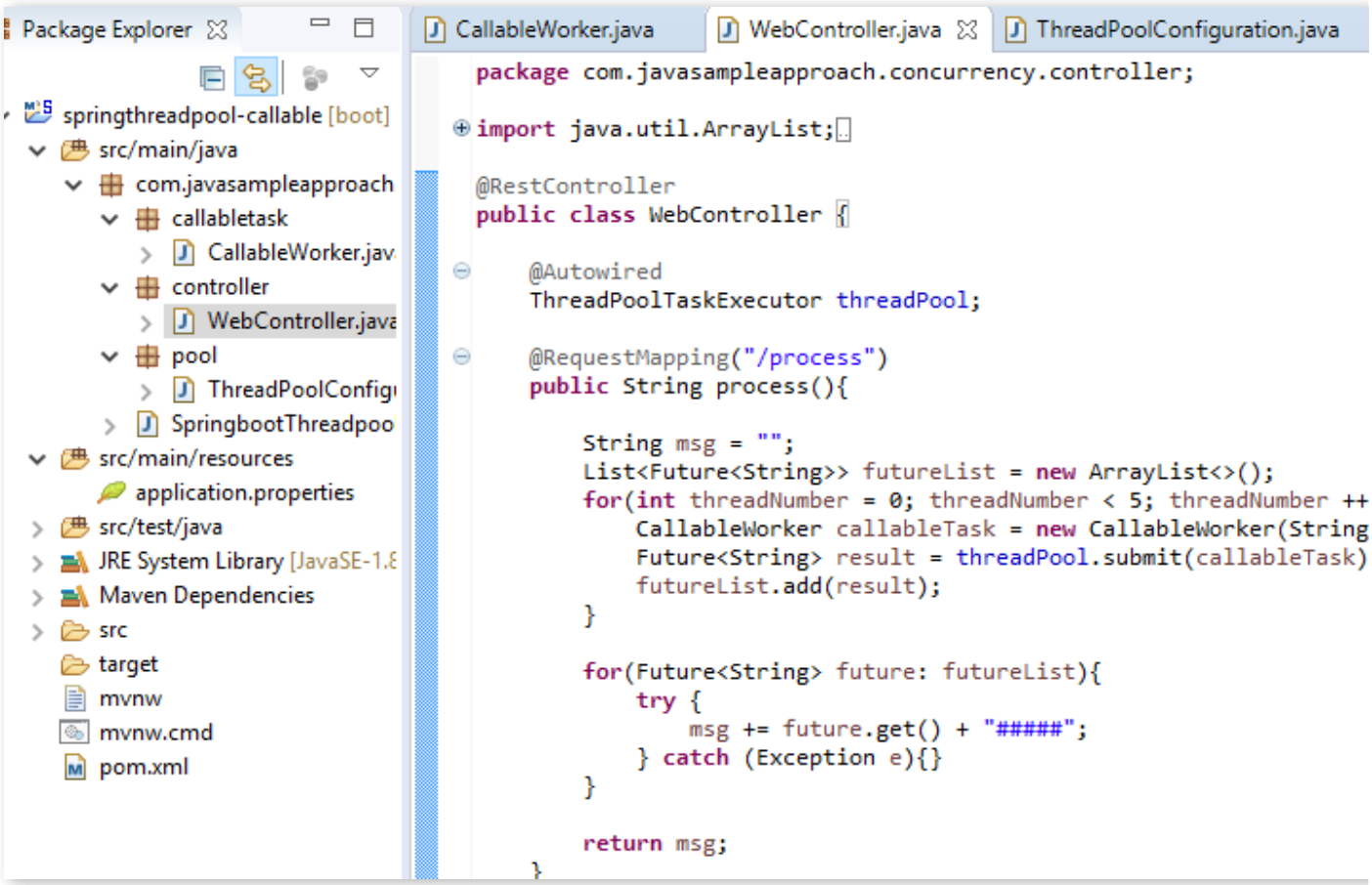
I. Technology

– Java 8
– Maven 3.3.9
– Editor: Spring Tool Suite – Version 3.7.3.RELEASE

II. Overview of Spring multi-thread with Callable project

– Program has **5 threads**, that works concurrently. At parent thread, for managing the returns of all child threads, We use **Callable interface**.

1. Structure of project



2. Step to do

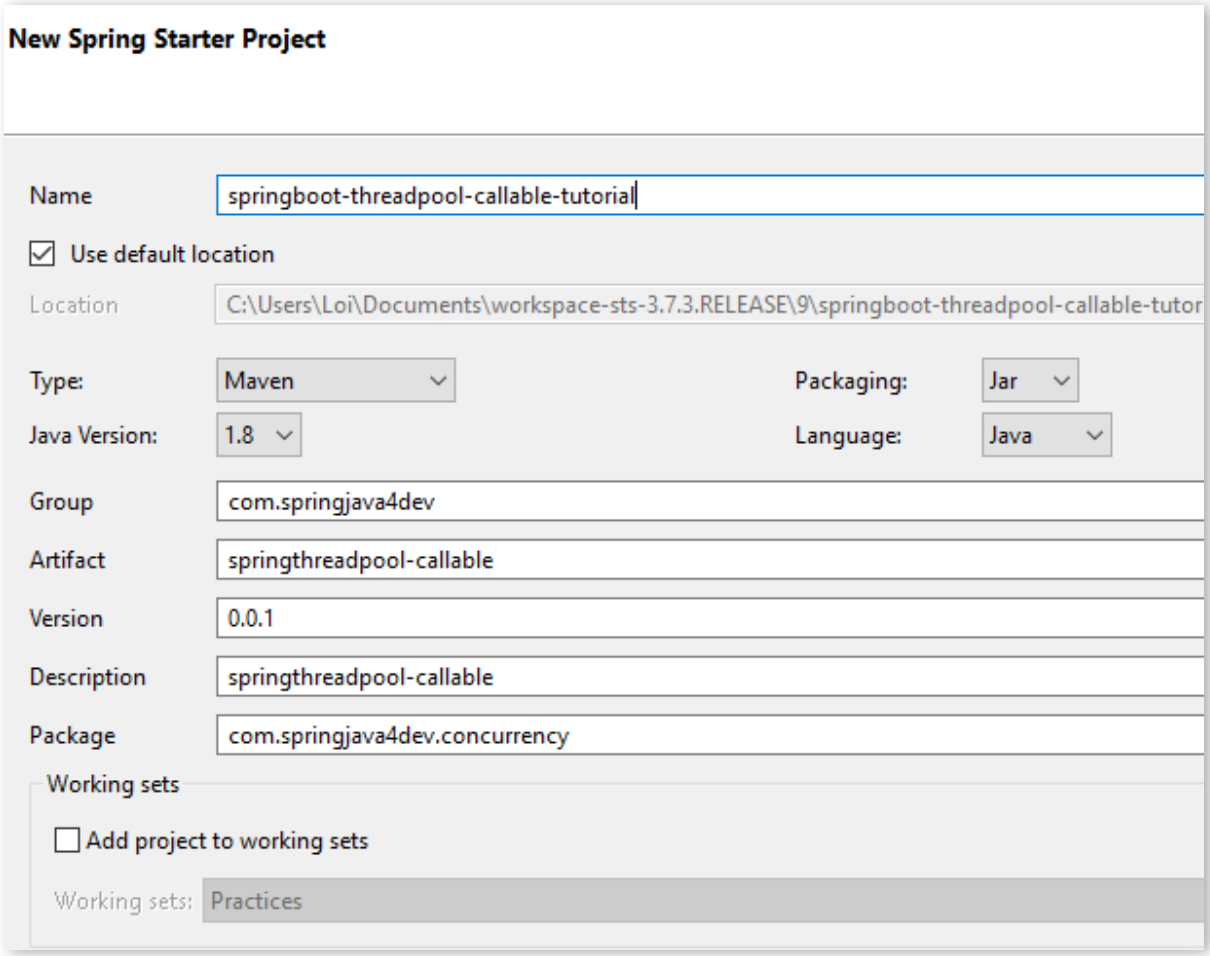
– Create Spring Boot project.
– Setup ThreadPoolTaskExecutor.

– Create CallableWorker that implements Callable interface.
– Create a Simple Web Controller

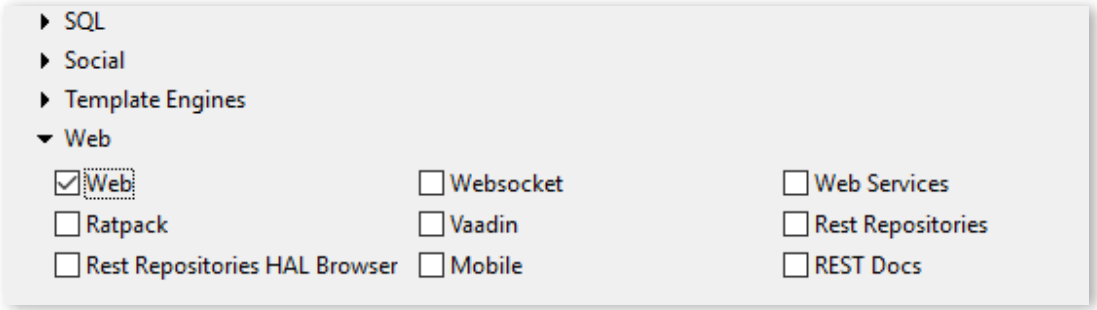
III. Practice

1. Create Spring Boot project

Open Spring Tool Suite, choose **File->New->Spring Starter Project**, input project information as below picture:



Press **Next** button, then add **needed dependencies**:
– For **Web MVC** dependency, choose **Web** then select **Web** as below:



Press **Finish** then **Spring Boot Project** will be created.

Check **pom.xml** dependency:

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

2. Setup ThreadPoolTaskExecutor

– Configure **application.properties**:

```
threadpool.corepoolsize=5
threadpool.maxpoolsize=10
```

– Configure a **Spring ThreadPool**:

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

@Configuration
public class ThreadPoolConfiguration {

    @Value("${threadpool.corepoolsize}")
    int corePoolSize;

    @Value("${threadpool.maxpoolsize}")
    int maxPoolSize;

    @Bean
    public ThreadPoolTaskExecutor taskExecutor() {
        ThreadPoolTaskExecutor pool = new ThreadPoolTaskExecutor();
        pool.setCorePoolSize(corePoolSize);
        pool.setMaxPoolSize(maxPoolSize);
        pool.setWaitForTasksToCompleteOnShutdown(true);
        return pool;
    }
}
```

3. Create Callable Worker

```
import java.util.concurrent.Callable;

public class CallableWorker implements Callable<String>{

    String name;

    public CallableWorker(String name) {
        this.name = name;
    }

    @Override
```

```
    public String call() throws Exception {
        process();
        String message = String.format("CallableWorker name: %s is Done", name);
        return message;
    }

    private void process(){
        for(int taskId=0; taskId < 10; taskId++){
            String message = String.format("CallableWorker name: %s is processing a taskId
            System.out.println(message);
        }
    }

}
```

4. Create a Simple Web Controller

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Future;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.springjava4dev.concurrency.callabletask.CallableWorker;

@RestController
public class WebController {

    @Autowired
    ThreadPoolTaskExecutor threadPool;

    @RequestMapping("/process")
    public String process(){

        String msg = "";
        List<Future<String>> futureList = new ArrayList<>();
        for(int threadNumber = 0; threadNumber < 5; threadNumber ++){
            CallableWorker callableTask = new CallableWorker(String.valueOf(threadNumber))
            Future<String> result = threadPool.submit(callableTask);
            futureList.add(result);
        }

        for(Future<String> future: futureList){
            try {
                msg += future.get() + "#####";
            } catch (Exception e){}
        }

        return msg;
```

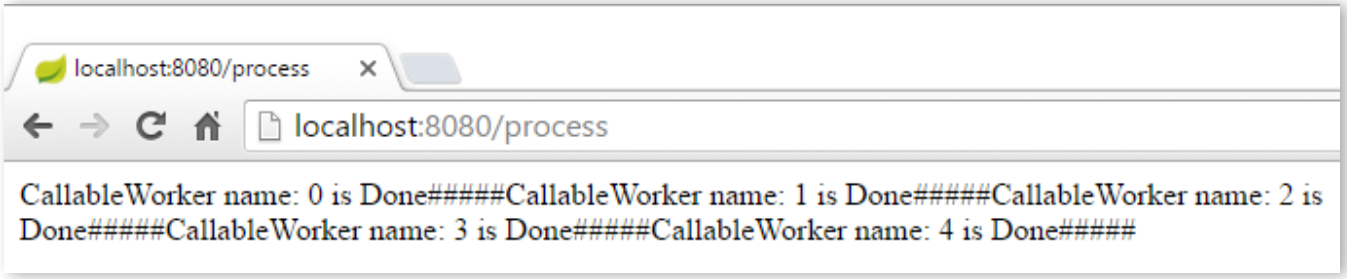
```
    }
}
```

IV. Run Project

– Build Spring Boot project

Set Goals: **clean install**

– Run Project:



– Logs

```
...
CallableWorker name: 4 is processing a taskId: 0
CallableWorker name: 2 is processing a taskId: 2
CallableWorker name: 4 is processing a taskId: 1
CallableWorker name: 2 is processing a taskId: 3
CallableWorker name: 4 is processing a taskId: 2
CallableWorker name: 2 is processing a taskId: 4
CallableWorker name: 4 is processing a taskId: 3
CallableWorker name: 2 is processing a taskId: 5
CallableWorker name: 4 is processing a taskId: 4
CallableWorker name: 2 is processing a taskId: 6
CallableWorker name: 4 is processing a taskId: 5
CallableWorker name: 2 is processing a taskId: 7
CallableWorker name: 4 is processing a taskId: 6
CallableWorker name: 2 is processing a taskId: 8
CallableWorker name: 4 is processing a taskId: 7
CallableWorker name: 2 is processing a taskId: 9
CallableWorker name: 4 is processing a taskId: 8
CallableWorker name: 4 is processing a taskId: 9
CallableWorker name: 1 is processing a taskId: 0
...
```

V. Source Code

springboot-threadpool-callable

By JavaSampleApproach | October 30, 2016.
Last updated on **June 4, 2017**.

Related Posts

- Java 9 CompletableFuture API Improvements – Delay and Timeout Support
- Spring Batch Partition for Scaling & Parallel Processing
- How to start Spring Async with Spring Boot
- How to create a Java Thread
- Java Thread Pool – ExecutorService
- Java Future
- Spring Boot + Angular 6 example | Spring Data JPA + REST + MySQL CRUD example
- Spring Boot + Angular 6 example | Spring Data JPA + REST + PostgreSQL CRUD example
- Angular 5 – Upload/Get Images to/from Spring Boot Server
- Angular 5 – Upload/Get MultipartFile to/from Spring Boot Server

Post Tags

callable multithreading runable spring boot

JavaSampleApproach

Home | Privacy Policy | Contact Us | Our Team

FOLLOW US

ABOUT US

We are passionate engineers in software development by Java Technology & Spring Framework. We believe that creating little good thing with specific orientation everyday can make great influence on the world someday.