**Archius**

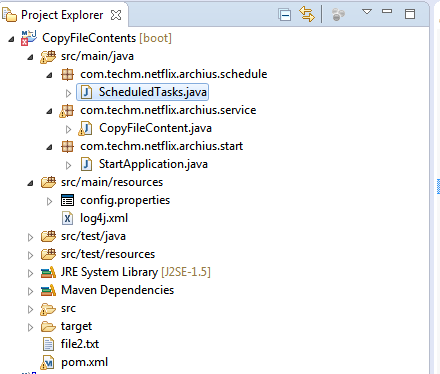
**1.Feature : Dynamic property Change**

**Prerequisites**

We must have installed the following

* JDK 1.7 or later
* Maven 3 or later

**Project Structure**



**Project dependencies**

|  |
| --- |
| **Pom file to generate war file** |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.netflix.archius.copyFile</groupId>  <artifactId>CopyFileContents</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>CopyFileContents</name>  <description>CopyFileContents</description>    <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>    <dependencies>  <dependency>  <groupId>com.netflix.archaius</groupId>  <artifactId>archaius-core</artifactId>  <version>0.7.4</version>  </dependency>    <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actuator</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <version>1.2.5.RELEASE</version>  <executions>  <execution>  <goals>  <goal>repackage</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build>  </project> |

**SpringBootInitializer File**

|  |
| --- |
| **StartApplication.java** |
| **Path 🡪 src/main/java/com/**techm **/netflix/archius/start/ StartApplication.java** |
| **package** com.techm.netflix.archius.start;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.context.annotation.ComponentScan;  **import** org.springframework.scheduling.annotation.EnableScheduling;  **import** org.springframework.stereotype.Component;  @SpringBootApplication  @Component  @ComponentScan("com.techm.netflix.archius")  @EnableScheduling  **public** **class** StartApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(StartApplication.**class**, args);  **final** Logger LOGGER = LoggerFactory.*getLogger*(StartApplication.**class**);  LOGGER.info("Application Started");  }  } |

**Scheduler File**

|  |
| --- |
| **ScheduledTasks.java** |
| **Path 🡪 src/main/java/com/techm/netflix/archius/schedule/ScheduledTasks.java** |
| **package** com.techm.netflix.archius.schedule;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.scheduling.annotation.Scheduled;  **import** org.springframework.stereotype.Component;  **import** com.techm.netflix.archius.service.CopyFileContent;  @Component  **public** **class** ScheduledTasks {  **private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(ScheduledTasks.**class**);  **private** **static** **final** SimpleDateFormat ***dateFormat*** = **new** SimpleDateFormat("HH:mm:ss");  @Scheduled(fixedRate = 30000)  **public** **void** copyFileContents() **throws** Exception {  ***log***.info("The time is now {}", ***dateFormat***.format(**new** Date()));  CopyFileContent fileOpe = **new** CopyFileContent();  fileOpe.CopyFileOperation();  }  } |

**Service File**

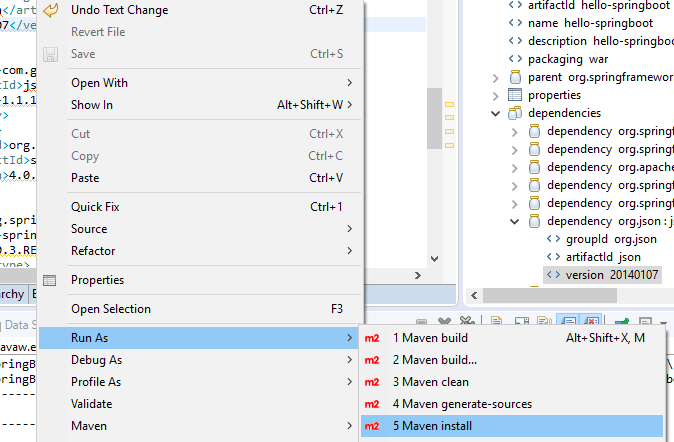
|  |
| --- |
| **CopyFileContent.java** |
| **Path 🡪 src/main/java/com/techm/netflix/archius/service/ CopyFileContent.java** |
| **package** com.techm.netflix.archius.service;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.FileNotFoundException;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** org.springframework.stereotype.Component;  **import** com.netflix.config.ConfigurationManager;  **import** com.netflix.config.DynamicPropertyFactory;  **import** com.netflix.config.DynamicStringProperty;  @Component  **public** **class** CopyFileContent {  **public** **void** CopyFileOperation() **throws** Exception {  ConfigurationManager.*loadCascadedPropertiesFromResources*("config");  DynamicStringProperty sourcedir = DynamicPropertyFactory.*getInstance*()  .getStringProperty("sourcedir", "");  DynamicStringProperty targetdir = DynamicPropertyFactory.*getInstance*()  .getStringProperty("targetdir", "");  DynamicStringProperty targetfileName = DynamicPropertyFactory.*getInstance*()  .getStringProperty("targetFilename", "");    String completepath = targetdir.get()+"/"+targetfileName.get();    System.***out***.println("source Directory ::" + sourcedir.get());  System.***out***.println("target Directory ::" + targetdir.get());  InputStream input = **null**;  OutputStream output = **null**;  File sourceFile = **null**;  File destFile,outputFile = **null**;  **try** {  /\* Source file, from which content will be copied \*/  sourceFile = **new** File(sourcedir.get());  } **catch** (Exception e) {  System.***err***.println("Excception occuredd in with source file :" + e);  }  **try** {  /\* destination file, where the content to be pasted \*/  destFile = **new** File(targetdir.get());  **if** (!destFile.exists()) {  destFile.mkdirs();  outputFile = **new** File(completepath);  System.***out***.println("DIR created");  }  } **catch** (Exception e) {  System.***out***.println("Exception occured with target file::" + e);  }  **try** {  /\* FileInputStream to read streams \*/  input = **new** FileInputStream(sourceFile);  /\* FileOutputStream to write streams \*/  output = **new** FileOutputStream(completepath);  **byte**[] buf = **new** **byte**[1024];  **int** bytesRead;  **while** ((bytesRead = input.read(buf)) > 0) {  output.write(buf, 0, bytesRead);  }  System.***out***.println("File Copied Sucessfully...");  } **catch** (FileNotFoundException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }  **finally** {  **try** {  **if** (**null** != input) {  input.close();  }  **if** (**null** != output) {  output.close();  }  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  } |

**config.properties**

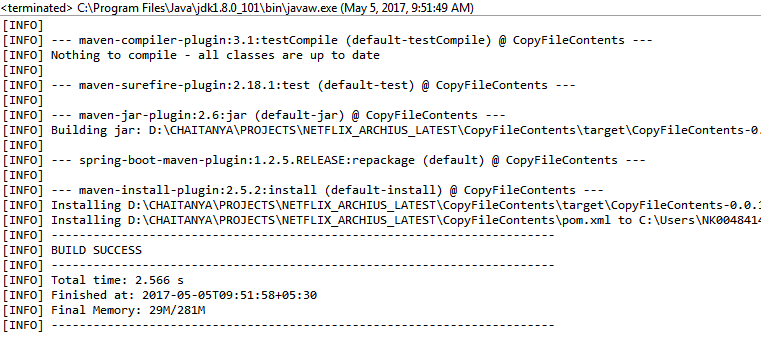
|  |
| --- |
| **config.properties** |
| **Path 🡪 src/main/resources/config.properties** |
| sourcedir = D:/CHAITANYA/CopiledFiles/input.txt  targetdir = D:/CHAITANYA/CopiledFiles/outputnew  targetFilename = output.txt |

**Steps to resolve Dependencies**

1. Right click on the pom file in eclipse. Run As 🡪 Maven install



This will resolves the dependency Issues.



Once Build Success execute the Application.

**Steps to Execute the application**

Execute the below java File and it will starts the Spring Boot Application

**src/main/java/com/**techm **/netflix/archius/start/ StartApplication.java**

Once the application has been started .As per the Scheduler Time period configured in the scheduler File .application executes the respective task configured .

NOTE : I have set to schedule time for 30 seconds.

@Scheduled(fixedRate = 30000)

**public** **void** copyFileContents() **throws** Exception {

***log***.info("The time is now {}", ***dateFormat***.format(**new** Date()));

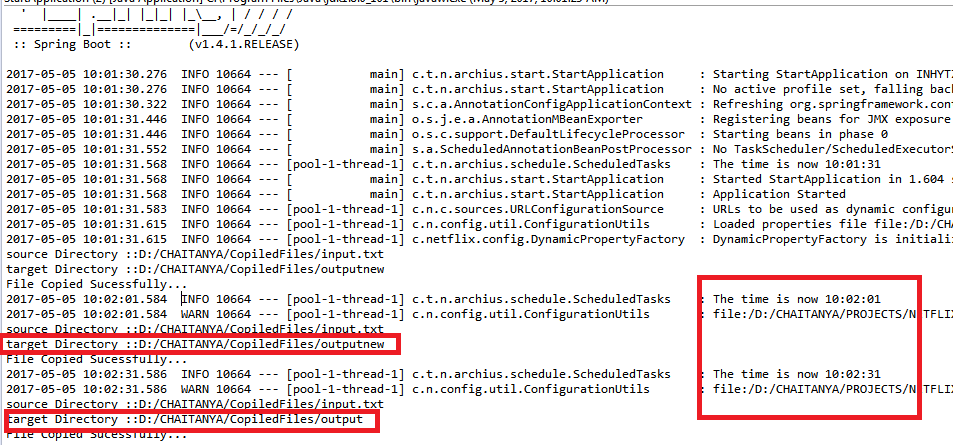
CopyFileContent fileOpe = **new** CopyFileContent();

fileOpe.CopyFileOperation();

}

**Use Case**: I have changed the target directory after the first scheduler finishes it task, So at the time of second scheduler executes its task it loads the new **targetdir** property from the config.properties file at run time .So that file will moved the new runtime location path.

On successfully completion of operation this log will be printed in console.



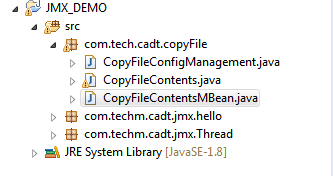
**2.Feature : JMX MBean that can be accessed via JConsole to inspect and invoke operations on properties**

**Prerequisites**

We must have installed the following

* JDK 1.5 or above

**Project Strecture**

****

**Project dependencies**

Not Required

**Interface File**

**(**Postfrix should be MBean**)**

Need to define all setter getter methods for List of properties & all the methods as a part of application.

|  |
| --- |
| **CopyFileContentsMBean.java** |
| **Path 🡪 src/main/java/com/**tech **/cadt/copyFile/**CopyFileContentsMBean.java |
| **package** com.tech.cadt.copyFile;  **public** **interface** CopyFileContentsMBean {  **public** **void** CopyFileOperation();  **public** **int** getThreadCount();  **public** **void** setThreadCount(**int** threadCount);  **public** String getSrcdir();  **public** **void** setSrcdir(String srcdir);  **public** String getTargetdir();  **public** **void** setTargetdir(String targetdir);  **public** String getFileName();  **public** **void** setFileName(String fileName);  **public** String doConfig();    } |

**Implementation File :**

|  |
| --- |
| **CopyFileContents.java** |
| **Path 🡪 src/main/java/com/tech /cadt/copyFile/CopyFileContent**.java |
| **package** com.tech.cadt.copyFile;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.FileNotFoundException;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **public** **class** CopyFileContents **implements** CopyFileContentsMBean {    **private** String srcdir;  **private** String targetdir;  **private** String fileName;  **private** **int** threadCount;    // Setter & getter Methods.  **public** CopyFileContents(String srcdir, String targetdir, String fileName,  **int** threadCount) {  **super**();  **this**.srcdir = srcdir;  **this**.targetdir = targetdir;  **this**.fileName = fileName;  **this**.threadCount = threadCount;  }    @Override  **public** String doConfig() {  **return** "CopyFileContents [srcdir=" + srcdir + ", targetdir="  + targetdir + ", fileName=" + fileName + ",  threadCount="+ threadCount + "]";  }  **public** **void** CopyFileOperation(){  System.***out***.println(doConfig());  String completepath = targetdir+"/"+fileName;  InputStream input = **null**;  OutputStream output = **null**;  File sourceFile = **null**;  File destFile,outputFile = **null**;  **try** {  /\* Source file, from which content will be copied \*/  sourceFile = **new** File(srcdir); }  **catch** (Exception e) {  System.***err***.println("Excception occuredd in with source file :" + e);  }  **try** {  /\* destination file, where the content to be pasted \*/  destFile = **new** File(targetdir);  **if** (!destFile.exists()) {  destFile.mkdirs();  outputFile = **new** File(completepath);  System.***out***.println("DIR created");  }  } **catch** (Exception e) {  System.***out***.println("Exception occured with target file::" + e);  }  **try** {  /\* FileInputStream to read streams \*/  input = **new** FileInputStream(sourceFile);  /\* FileOutputStream to write streams \*/  output = **new** FileOutputStream(completepath);  **byte**[] buf = **new** **byte**[1024];  **int** bytesRead;  **while** ((bytesRead = input.read(buf)) > 0) {  output.write(buf, 0, bytesRead);  }  System.***out***.println("File Copied Sucessfully...");  } **catch** (FileNotFoundException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }  **finally** {  **try** {  **if** (**null** != input) {  input.close();  }  **if** (**null** != output) {  output.close();  }  } **catch** (IOException e) {  e.printStackTrace();  }  }    }    } |

**JMX Server Configuration File**

Steps Need to Follow

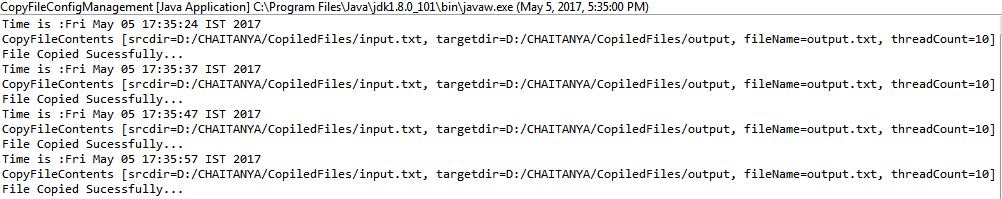
1. Create an MBeanServer that is the interface for MBean manipulation on the agent side. Use ***getPlatformMBeanServer()*** API method of ManagementFactory.

2. Define the object name for the MBean. The object name is an instance of the JMX class ***ObjectName*** and must contain a domain and a list of key-properties.

3. Create a new instance of the MBean and register it to MBeanServer, using ***registerMBean(Object object, ObjectName name)*** API method of MBeanServer.

|  |
| --- |
| **CopyFileConfigManagement.java** |
| **Path 🡪 src/main/java/com/tech / cadt /copyFile / CopyFileConfigManagement.java** |
| **package** com.tech.cadt.copyFile;  **import** java.lang.management.ManagementFactory;  **import** java.util.Date;  **import** javax.management.InstanceAlreadyExistsException;  **import** javax.management.MBeanRegistrationException;  **import** javax.management.MBeanServer;  **import** javax.management.MalformedObjectNameException;  **import** javax.management.NotCompliantMBeanException;  **import** javax.management.ObjectName;  **public** **class** CopyFileConfigManagement {  **private** **static** **final** **int** ***DEFAULT\_NO\_THREADS*** = 10;  **public** **static** **void** main(String[] args) **throws** MalformedObjectNameException,  InstanceAlreadyExistsException, MBeanRegistrationException,  NotCompliantMBeanException, InterruptedException {    MBeanServer mbs = ManagementFactory.*getPlatformMBeanServer*();  // register the MBean  String srcdir = "D:/CHAITANYA/CopiledFiles/input.txt";  String targetdir = "D:/CHAITANYA/CopiledFiles/output";  String fileName = "output.txt";  CopyFileContents mBean = **new** CopyFileContents(srcdir, targetdir,  fileName, ***DEFAULT\_NO\_THREADS***);  ObjectName name = **new** ObjectName(  "com.techm.cadt.copyFile:type=CopyFileContents");  mbs.registerMBean(mBean, name);  **do** {  Thread.*sleep*(10000);  Date now = **new** Date(); // initialize date  System.***out***.println("Time is :" + now);  mBean.CopyFileOperation();  } **while** (mBean.getThreadCount() != 0);  }  } |

**Console Log:**

****

## JConsole

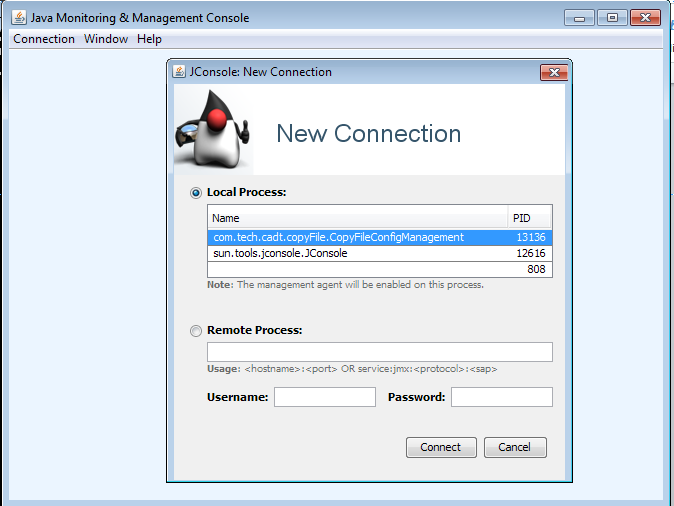
The Java SE platform includes the JConsole monitoring and management tool, which complies with the JMX specification. JConsole uses the extensive instrumentation of the Java VM (the platform MXBeans) to provide information about the performance and resource consumption of applications that are running on the Java platform.

## Monitoring an Application by Using JConsole

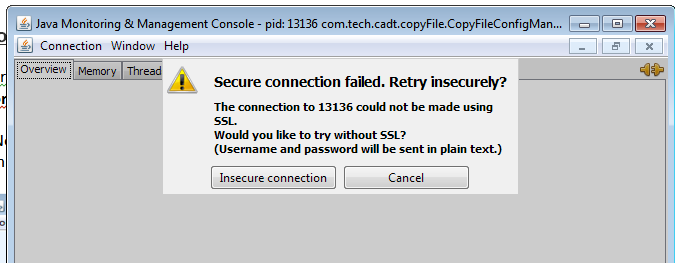
start JConsole by using the following command:

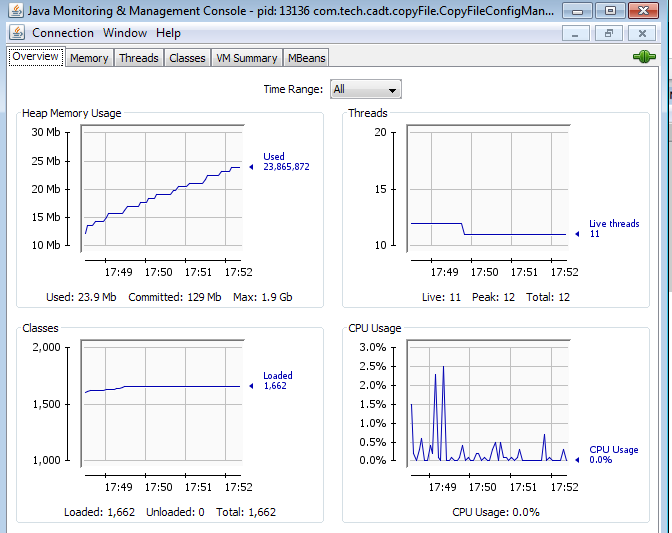
**jconsole**

A New Connection dialog box is displayed and select the corresponding instance and click connect.

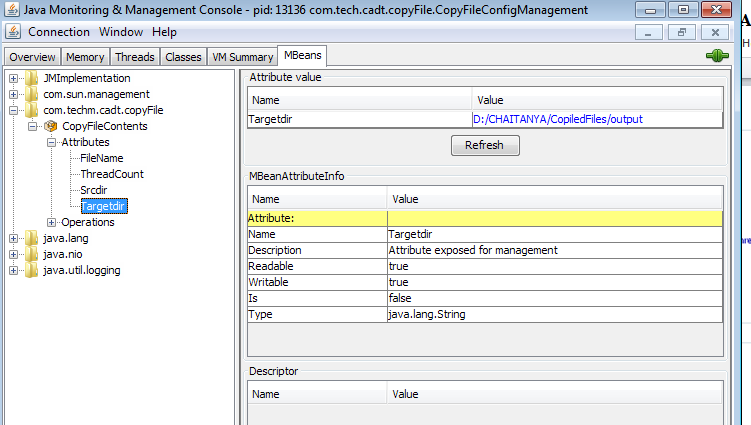


Click On Insecure Connection



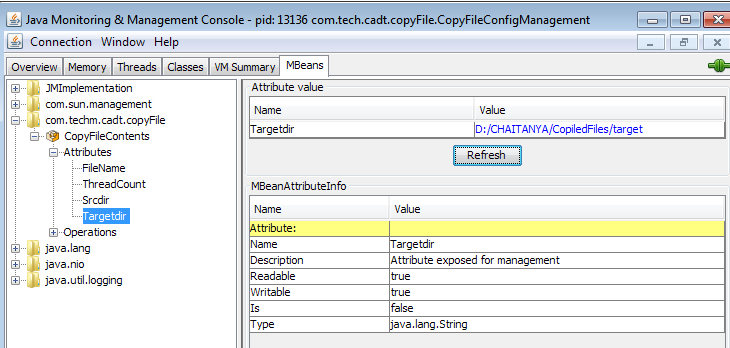


Select Means Tab and navigate to the application select the attributes .

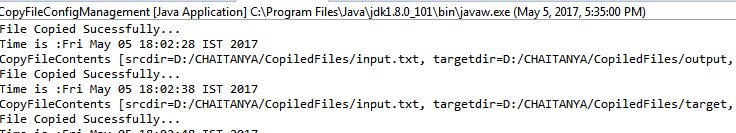


Changing the Attribute Values in the above console will reflects in the application at run time.

Let see I am changing the Target Dircetory attribute value.



**console log**



Target Directory has been changed without restarting the server.