**INTRODUCTION**

**SPRING INTEGRATION IN SPRING BOOT:**

**Spring Integration** is the best suite for an **Enterprise Application** which is working on **a** **Message Centric Architecture** or **an** **Event Driven Architecture** which needs to be integrated inside **a Spring Framework.**

The different **data sources** that we deal with are:

**1.File Systems**

**2.Messaging Queues or Messaging Channels.**

**3.Database.**

**4.FTP Server.**

**5.Secure FTP Server.**

**6.Object Stores, etc….**

So these are **different data stores** where we **collect data** and **persist data** as well as create different **Adapters** to connect to the **Application**.

**ADAPTERS** **DATASTORES**

FILE ADAPTER

FILES

**APPLICATION**

CHANNEL ADAPTER

MESSAGING QUEUES/CHANNELS

DB ADAPTER

DATABASE(DB)

FTP SERVER,SFTP SERVER

FTP ADAPTER

**Example(s):**

1. If we want to connect to a File Data store, we have to create a File Adapter.
2. If we want to connect to a Messaging Data store, we have to create a Messaging Adapter.

**For every type of Integration pattern we create an Adapter.**

**Spring Integration** gives us **different adapters** which can be integrated into **A Spring Based Application** using the **Spring Framework** and it suits well for the **Spring Boot Application** because it suits well for the **Spring Boot Application** because **Spring Boot** comes in its own **Tomcat**.

With **Spring Integration**, we can easily create:

1.**Event Driven Architecture’s**,

2.**Message Driven Architecture’s**.

**Objectives:**

**Design A Spring Integration inside a Spring Boot Application**

**using a File Adapter:**

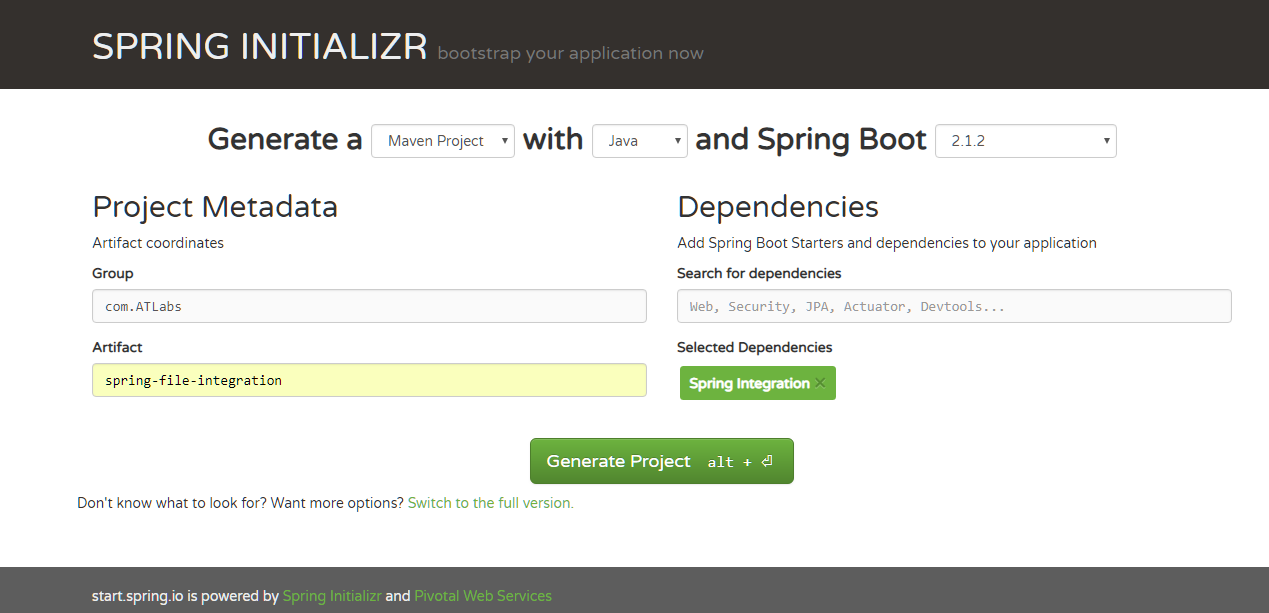
**(a).** Read files from a particular directory.

**(b).** Transform the content.

**(c).** Pushing the content to a different folder.

**Steps:**

1. Go to [**https://start.spring.io/**](https://start.spring.io/)
2. Configure the project using **SPRING INITIALZR**:



3.Unzip the extracted folder [**spring-file-integration.zip**] using **winrar**.

4.Open the [**spring-file-integration]** file in **Spring Tool Suite 4:**

can downloaded from <https://spring.io/tools>.

Spring Integration Flow:

return a type of Integration Flow

2.Integration Flows can be created using Integration Flows.

3.So there is a Utility class through which you can create the

Integration flows,

like a builder pattern[build upon each of these patterns,so we can do that with

spring Integration as well.

4.Create a fileReader

5.Use FileReadingMessageSource is a adapter which can go and read a file.

6.Read from directory "source1",return source.

7.Another bean,allow spring boot to use it.

8.As a part of the from in the Integration Flow,I need to read the files from the particular directory

called "source".

9.Now once we read the source we need to process it.

10.we need to transform and finally we need to write it back.

11.In that case we can use the .handle option

12.So before the handle lets write the transformation.

13.So I need to read the file and its content and then I need to

transform it into something else.

14.For now what I will be getting in the transformation part is the

file name because we are using the file reading message source

15.We are not using the file we are just using the directory and getting all the files

to be read from that source.

16.We will create a transformer and in the transformer we are going to create a method

which is going to return a string say(transform),the input for this is nothing but a file

name or a Filepath,so once we get a path we need to return something[making it empty]

17.Using the Paths.get(filepath),we can directly give the filePath.

18.Now we need to read this as well.

19.Using Java8 we are going to use Files.ReadAllBytes,so I am going to say read all my files.

Files.readAllBytes(Paths.get(filePath)),so this is going to throw a IO Exception.

20.And we need to convert this into a String Files.readAllBytes(Paths.get(filePath)).toString

21.So this is reading all the data from this particular file,the File.ReadAllBytes will read all the data

from the file and since these are bytes I can do a new String on this.

new String(Files.readAllBytes(Paths.get(filePath))) assign this to content of the file.

String content = new String(Files.readAllBytes(Paths.get(filePath))) ;

22.For now we are not doing any transformation later on we can come back and do some transformation.

so that we can identify whether the transformation is working or not.

23.So now lets go back go to the Configuration File,so in the transformer,we are saying that use my Transfomer Class and we need to Autowire this.

and I need to provide the method which needs to be called.

24.The method name is called Transform transform(transformer,"transform") and the object is now injected here.

25.Now once the message are transformed lets move into the handlerpart,we need to handle it.

26.So we need a file writer (),so we did a file reader so after transforming it we need to write it.

27.So we need a file handler(),FileWritingMessageHandler and we need provide the directory in which we need to write this,

going to say destination.So we create a folder called destination.So this is where we are going to write the file back.

28.So we have the reader we have the writer and in the handler we need to setExpectedReply(false).

29.I don't need any reply from this so I am just going use this.Make this as bean and make this as public.

30.So I have the file reader,I have the file writer,however there is an error[file-incompatible-type] which shows that

@Bean

public IntergrationFlows integrationFlow(){

return IntegrationFlows.from(fileReader())

.transform(transfomer,"transform")

.handle(fileWriter()));

}

So in order to provide a file reader we need to read the file so how often do we need to read the file.

31.So we need to provide a polar here.

@Bean

public IntegrationFlow integrationFlow() {

return IntegrationFlows.from(fileReader(), new Consumer<SourcePollingChannelAdapterSpec> () {

@Override

public void accept(SourcePollingChannelAdapterSpec sourcePollingChannelAdapterSpec) {

SourcePollingChannelAdapterSpec.poller((Pollers.fixedDelay(500));

}

})

And there is something called Pollers,Pollers is a utility inside the Spring Integration Itself,which is used to get the fixed delay,

or any configuration can be delayed here.

So,I am going to say for every 500 [milli-seconds],you just querry this guy.

32.So I am going to use Lamda's here:

@Bean

public IntegrationFlow integrationFlow() {

return IntegrationFlows.from(fileReader(),

SourcePollingChannelAdapterSpec -> SourcePollingChannelAdapterSpec.poller((Pollers.fixedDelay(500));

}

})

--renamed--

@Bean

public IntegrationFlow integrationFlow() {

return IntegrationFlows.from(fileReader(),

spec -> spec.poller((Pollers.fixedDelay(500));

}

})

33.So what we are doing is using the spring Integration Flows,we are saying that we will read the file,

using the file reader and the file reader is here where it reads the file from the directory source,

and I am going to read every 500 Milli-Second(s),so the polar is going to have a delay of every 500 Milli-Second(s).

so this nothing but the polar.

34. And after the files are read I need to do some transformation,the file path is now given to the Transformer.

and inside the transformer we have a transformation method,lets do some transformation here.[Transformer.java]

I will just add the content as Transformed.

@Component

public class Transformer {

public String transform(String filePath) throws IOException {

String content = new String(Files.readAllBytes(Paths.get(filePath)));

return "Transformed"+content;

}

}

35.So what I am doing is I am transforming in this particular layer.

.transform(transformer, "transform")

and finally we need to write it back to another file.

so we have a file writer for that and the file writer used is

FileWritingMessageHandler.

So,these are different Spring Utilities created by Spring Integration.

These are different classes created by Spring Integration,which can be directly integrated.

and see that these are utilities provided by Spring Integration,in order too create a need to

build a kind of pattern here.

---------------------Summary-------------------------

Using spring Integration you can read/write into different data sources.

Here we have used file data source for both the purpose for reading as well as for writing.

Using Spring Integration Flows we can make a builder pattern of out of it.

Say you want to read using A File reader and I have the destination mention as source mentioned here.

Using the delay of 500 millsec we are going to poll the folder.

And we need to transform this data into a differnet form.

For now since I have a String,I am not doing anything fancy I am just reading the path,

Reading the file content and I am adding a text called transform.

And finally we are going to use the file writer with the destination folder mentioned here,where we are going to

write the files.

These are the only configuration we need to add.

So,this is how spring integration can be used inside a spring boot project.

We need to use differnet File Adapter,for now we are using Spring Integration File Adapter.

There are differnet File Adapters AMQP,JMS,etc.

We can integrate those into the Integration Flows here and get it done out of the box,and we can have

different Pollers,so here its a File Reader,so I have a poller,if you have queues,it will all be based on events.