

# Querying Record

# Query Record Fields

## Source JSON Record

```
{  
    "id" : 6523,  
    "ident" : "00A",  
    "type" : "heliport",  
    "name" : "Total Rf Heliport",  
    "latitude_deg" : 40.0708,  
    "longitude_deg" : -74.93360137939453,  
    "elevation_ft" : 11,  
    "continent" : "NA",  
    "iso_country" : "US",  
    "iso_region" : "US-PA",  
    "municipality" : "Bensalem",  
    "scheduled_service" : "no",  
    "gps_code" : "00A",  
    "iata_code" : null,  
    "local_code" : "00A",  
    "home_link" : null,  
    "wikipedia_link" : null,  
    "keywords" : null  
}
```

## SQL Like Select Statement

```
Select * from DATA  
WHERE  
scheduled_service = 'no'
```

# Querying Record Field

QueryRecord Processor  
to query like sql

 <span style="color: orange;">!</span> QueryRecord	QueryRecord 1.15.3 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Required field		<input checked="" type="checkbox"/>	<input type="button" value="+"/>
Property	Value		
Record Reader	<span style="color: orange;">?</span> No value set		
Record Writer	<span style="color: orange;">?</span> No value set		
Include Zero Record FlowFiles	<span style="color: orange;">?</span> true		
Cache Schema	<span style="color: orange;">?</span> true		
Default Decimal Precision	<span style="color: orange;">?</span> 10		
Default Decimal Scale	<span style="color: orange;">?</span> 0		

# Querying Record Field

QueryRecord Processor  
to query like sql

	<b>QueryRecord</b>	QueryRecord 1.15.3
		org.apache.nifi - nifi-standard-nar
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Required field

Property	Value
Record Reader	JsonTreeReader
Record Writer	JsonRecordSetWriter
Include Zero Record FlowFiles	true
Cache Schema	true
Default Decimal Precision	10
Default Decimal Scale	0
yes	SELECT * FROM FLOWFILE WHERE scheduled_se... 

EL ✓ PARAM ✓

```
1 SELECT * FROM FLOWFILE
2 WHERE scheduled_service = 'yes'
```

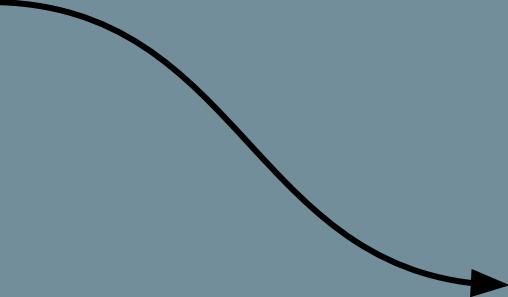
Set empty string

CANCEL OK

# Query Record Fields

```
Select * from FLOWFILE  
WHERE  
scheduled_service = 'yes'
```

FlowFile



```
{  
  "id" : 6523,  
  "ident" : "00A",  
  "type" : "heliport",  
  "name" : "Total Rf Heliport",  
  "latitude_deg" : 40.0708,  
  "longitude_deg" : -74.93360137939453,  
  "elevation_ft" : 11,  
  "continent" : "NA",  
  "iso_country" : "US",  
  "iso_region" : "US-PA",  
  "municipality" : "Bensalem",  
  "scheduled_service" : "no",  
  "gps_code" : "00A",  
  "iata_code" : null,  
  "local_code" : "00A",  
  "home_link" : null,  
  "wikipedia_link" : null,  
  "keywords" : null  
}
```

Putting to database

# Inserting JSON into DB

JSON

```
{  
  "id": 40814,  
  "ident": "SK-377",  
  "type": "small_airport",  
  "name": "Barbosa Airport",  
  "latitude_deg": 5.943333,  
  "longitude_deg": -73.611389,  
  "elevation_ft": 5176,  
  "continent": "SA",  
  "iso_country": "CO",  
  "iso_region": "Santander Department",  
  "municipality": "Barbosa",  
  "scheduled_service": "no",  
  "gps_code": null,  
  "iata_code": null,  
  "local_code": "BSA",  
  "home_link": null,  
  "wikipedia_link": null,  
  "keywords": null,  
  "frequency_mhz": "121.9"  
}
```

Airports Table

```
CREATE TABLE AIRPORTS (  
  ID INT PRIMARY KEY NOT NULL,  
  IDENT TEXT NOT NULL,  
  TYPE TEXT NOT NULL,  
  NAME TEXT NOT NULL,  
  LATITUDE_DEG DECIMAL,  
  LONGITUDE_DEG DECIMAL,  
  ELEVATION_FT INT,  
  CONTINENT TEXT,  
  ISO_COUNTRY TEXT,  
  ISO_REGION TEXT,  
  MUNICIPALITY TEXT,  
  SCHEDULED_SERVICE TEXT,  
  GPS_CODE TEXT,  
  IATA_CODE TEXT,  
  LOCAL_CODE TEXT,  
  HOME_LINK TEXT,  
  KEYWORDS TEXT,  
  FREQUENCY_MHZ TEXT  
);
```

# – PutDatabaseRecord Processor

CRUD operations  
with Database

PutDatabaseRecord		
PutDatabaseRecord 1.14.0 org.apache.nifi - nifi-standard-nar		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Configure Processor

⚠ Invalid

SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field +

Property	Value
Record Reader	No value set
Database Type	Generic
Statement Type	No value set
Data Record Path	No value set
Database Connection Pooling Service	No value set
Catalog Name	No value set
Schema Name	No value set
Table Name	No value set
Translate Field Names	true
Unmatched Field Behavior	Ignore Unmatched Fields
Unmatched Column Behavior	Fail on Unmatched Columns
Quote Column Identifiers	false

CANCEL APPLY

# – PutDatabaseRecord Processor

Insert data to DB

	<b>PutDatabaseRecord</b>	PutDatabaseRecord 1.14.0
		org.apache.nifi - nifi-standard-nar
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Required field

Property	Value	
Record Reader	JsonTreeReader	→
Database Type	PostgreSQL	
Statement Type	INSERT	
Data Record Path	No value set	
Database Connection Pooling Service	Postgres_DBCPConnectionPool	→
Catalog Name	No value set	
Schema Name	No value set	
Table Name	airports	
Translate Field Names	true	
Unmatched Field Behavior	Ignore Unmatched Fields	
Unmatched Column Behavior	Fail on Unmatched Columns	
Quote Column Identifiers	false	

# \_ PutDatabaseRecord Processor

Statement Type	 <b>INSERT</b>	
Data Record Path	 No value set	
<b>Database Connection Pooling Service</b>	 <b>Postgres_DBCPConnectionPool</b>	
Catalog Name	 No value set	



Required field

Property	Value
Database Connection URL	 <code>jdbc:postgresql://localhost:5432/postgres</code>
Database Driver Class Name	 <code>org.postgresql.Driver</code>
Database Driver Location(s)	 <code>./postgresql-42.2.2.jar</code>
Kerberos User Service	 No value set
Kerberos Credentials Service	 No value set
Kerberos Principal	 No value set
Kerberos Password	 No value set
Database User	 <code>postgres</code>
Password	 Sensitive value set
Max Wait Time	 <code>500 millis</code>
Max Total Connections	 <code>8</code>
Validation query	 No value set
Minimum Idle Connections	 <code>0</code>
Max Idle Connections	 <code>8</code>

# \_ PutDatabaseRecord Processor

## Results

```
[postgres=# select * from airports;
 id | ident |      type      |           name           | latitude_deg | longitude_deg | elevation_ft | continent | iso_country | iso_region |   municipal
ity | scheduled_service | gps_code | iata_code | local_code | home_link | keywords
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 40814 | SK-377 | small_airport | Barbosa Airport | 5.943333 | -73.611389 | 5176 | SA | CO |  | Barbosa
       | no     |             |                 | BSA        |            |            |            |            |            |
342103 | ZYXW  | closed    | Mt Logan Airstrip | 60.79299 | -138.694027 | 1997 | NA | CA |  | Unorganized
Yukon | no     |             | ZYXW          | ZYA        | YK90        |            |            |            |
317861 | ZYYK  | medium_airport | Yingkou Lanqi Airport | 40.542524 | 122.3586 | AS | CN |  | Laobian, Yi
ngkou | yes    |             | ZYYK          | YKH        |            |            |            |            |
       | SKAR   | medium_airport | El Eden Airport | 4.45278  | -75.7664 | 3990 | SA | CO |  | Armenia
       | yes    |             | SKAR          | AXM        | AXM         |            |            |            |
(4 rows)

postgres=# ]
```

# Custom Development with Apache NiFi

# Custom Development with Apache NiFi

## Capabilities

ExecuteScript/ExecuteGroovyScript

InvokeScriptedProcessor

Custom Processor/Controller Services

# Building Custom Processor

# Building Custom Processors

## Why Custom stuff?

NiFi comes with out of the box **298** processors and **103** controller services readily available for your needs (at the time of writing) .

What if we need something which is not readily available?

OR

What if the existing set of processors is somehow **underperforming** and you would turn them into a custom processor

Custom Processors to the rescue!!

# Building Custom Processors

## Pre-requisites

Java 8

Maven

NiFi 1.15.3

IntelliJ / Eclipse

# Building Custom Processors

## Development

Custom Processor

ExecuteScript

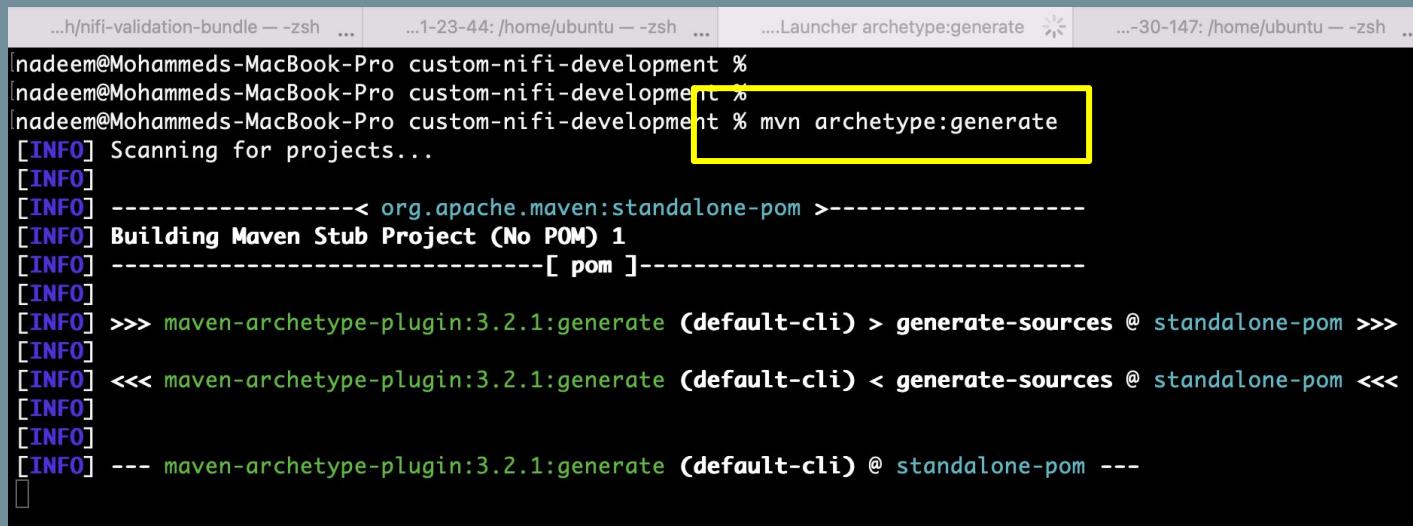
# Creating Custom Processor

# Steps to create custom processor

## Step 1: Maven archetype

Maven archetype provides easiest way to generate NiFi processor

```
$ > mvn archetype:generate
```



The screenshot shows a terminal window with four tabs at the top. The active tab contains the command `mvn archetype:generate`, which is highlighted with a yellow rectangle. The terminal output shows the process of generating a Maven project, including scanning for projects, building a Maven Stub Project, and executing the maven-archetype-plugin to generate sources.

```
[nadeem@Mohammeds-MacBook-Pro custom-nifi-development %] [nadeem@Mohammeds-MacBook-Pro custom-nifi-development %] [nadeem@Mohammeds-MacBook-Pro custom-nifi-development % mvn archetype:generate] [INFO] Scanning for projects... [INFO] [INFO] -----< org.apache.maven:standalone-pom >----- [INFO] Building Maven Stub Project (No POM) 1 [INFO] -----[ pom ]----- [INFO] [INFO] >>> maven-archetype-plugin:3.2.1:generate (default-cli) > generate-sources @ standalone-pom >>> [INFO] <<< maven-archetype-plugin:3.2.1:generate (default-cli) < generate-sources @ standalone-pom <<< [INFO] [INFO] --- maven-archetype-plugin:3.2.1:generate (default-cli) @ standalone-pom ---
```

# Step 1: mvn archetype:generate

```
3155: remote -> uk.co.markg.archetypes:java11-junit5 (An archetype for generate java 11 projects with junit 5.)  
3156: remote -> uk.co.nemstix:basic-javaee7-archetype (A basic Java EE7 Maven archetype)  
3157: remote -> uk.co.solong:angular-spring-archetype (So Long archetype for RESTful spring services with an AngularJS frontend. Includes debian deployment)  
3158: remote -> us.fatehi:schemacrawler-archetype-maven-project (-)  
3159: remote -> us.fatehi:schemacrawler-archetype-plugin-command (-)  
3160: remote -> us.fatehi:schemacrawler-archetype-plugin-dbconnector (-)  
3161: remote -> us.fatehi:schemacrawler-archetype-plugin-lint (-)  
3162: remote -> ws.osiris:osiris-archetype (Maven Archetype for Osiris)  
3163: remote -> xyz.luan.generator:xyz-gae-generator (-)  
3164: remote -> xyz.luan.generator:xyz-generator (-)  
3165: remote -> za.co.absa.hyperdrive:component-archetype (-)  
3166: remote -> za.co.absa.hyperdrive:component-archetype_2.11 (-)  
3167: remote -> za.co.absa.hyperdrive:component-archetype_2.12 (-)  
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1969: nifi  
Choose archetype:  
1: remote -> org.apache.nifi:nifi-processor-bundle-archetype (-)  
2: remote -> org.apache.nifi:nifi-service-bundle-archetype (-)  
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): :
```

type `nifi`

```
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1969: nifi  
Choose archetype:  
1: remote -> org.apache.nifi:nifi-processor-bundle-archetype (-)  
2: remote -> org.apache.nifi:nifi-service-bundle-archetype (-)  
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): :
```

# Step 1: mvn archetype:generate

```
SiOR: Remote > 2a:cc:ab5a:nycarlive:component-archetype_2.1.2 < 
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1969: nifi
Choose archetype:
1: remote -> org.apache.nifi:nifi-processor-bundle-archetype (-)
2: remote -> org.apache.nifi:nifi-service-bundle-archetype (-)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): : 1
Choose org.apache.nifi:nifi-processor-bundle-archetype version:
1: 0.0.2-incubating
2: 0.1.0-incubating
3: 0.2.0-incubating
4: 0.2.1
5: 0.3.0
6: 0.4.0
7: 0.4.1
8: 0.5.0
9: 0.5.1
10: 0.6.0
11: 0.6.1
12: 0.7.0
13: 0.7.1
14: 0.7.2
15: 0.7.3
16: 0.7.4
17: 1.0.0-BETA
18: 1.0.0
19: 1.0.1
20: 1.1.0
```

```
12: 0.7.0
13: 0.7.1
14: 0.7.2
15: 0.7.3
16: 0.7.4
17: 1.0.0-BETA
18: 1.0.0
19: 1.0.1
20: 1.1.0
21: 1.1.1
22: 1.1.2
23: 1.2.0
24: 1.3.0
25: 1.4.0
26: 1.5.0
27: 1.6.0
28: 1.7.0
29: 1.7.1
30: 1.8.0
31: 1.9.0
32: 1.9.1
33: 1.9.2
34: 1.10.0
35: 1.11.0
36: 1.11.1
37: 1.11.2
38: 1.11.3
39: 1.11.4
40: 1.12.0
41: 1.12.1
42: 1.13.0
43: 1.13.1
44: 1.13.2
45: 1.14.0
46: 1.15.0
47: 1.15.1
48: 1.15.2
49: 1.15.3
50: 1.16.0
51: 1.16.1
52: 1.16.2
53: 1.16.3
54: 1.17.0
```

```
Choose a number: 54: 49
Define value for property 'artifactBaseName': ■
```

# Step 1: mvn archetype:generate

```
46: 1.15.0  
47: 1.15.1  
48: 1.15.2  
49: 1.15.3  
50: 1.16.0  
51: 1.16.1  
52: 1.16.2  
53: 1.16.3  
54: 1.17.0
```

Choose a number: 54: 49

Define value for property 'artifactBaseName':

# Step 1: mvn archetype:generate

```
Choose a number: 54: 49
Define value for property 'artifactBaseName': example
[INFO] Using property: nifiVersion = 1.15.3
Define value for property 'groupId': org.learning.nifi
Define value for property 'artifactId': example-processor
Define value for property 'version' 1.0-SNAPSHOT: : 1.0
Define value for property 'package' org.learning.nifi.processors.example: :
Confirm properties configuration:
artifactBaseName: example
nifiVersion: 1.15.3
groupId: org.learning.nifi
artifactId: example-processor
version: 1.0
package: org.learning.nifi.processors.example
Y: : Y
```

# Step 1: mvn archetype:generate

```
Define value for property 'artifactBaseName': example
[INFO] Using property: nifiVersion = 1.15.3
Define value for property 'groupId': org.learning.nifi
Define value for property 'artifactId': example-processor
Define value for property 'version' 1.0-SNAPSHOT: : 1.0
Define value for property 'package' org.learning.nifi.processors.example: :
Confirm properties configuration:
artifactBaseName: example
nifiVersion: 1.15.3
groupId: org.learning.nifi
artifactId: example-processor
version: 1.0
package: org.learning.nifi.processors.example
Y: : Y
[INFO] -----
[INFO] Using following parameters for creating project from Archetype: nifi-processor-bundle-archetype:1.15.3
[INFO] -----
[INFO] Parameter: groupId, Value: org.learning.nifi
[INFO] Parameter: artifactId, Value: example-processor
[INFO] Parameter: version, Value: 1.0
[INFO] Parameter: package, Value: org.learning.nifi.processors.example
[INFO] Parameter: packageInPathFormat, Value: org/learning/nifi/processors/example
[INFO] Parameter: package, Value: org.learning.nifi.processors.example
[INFO] Parameter: version, Value: 1.0
[INFO] Parameter: artifactBaseName, Value: example
[INFO] Parameter: groupId, Value: org.learning.nifi
[INFO] Parameter: artifactId, Value: example-processor
[INFO] Parameter: nifiVersion, Value: 1.15.3
[INFO] Project created from Archetype in dir: /Users/nadeem/Documents/custom-nifi-development/example-processor
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15:55 min
[INFO] Finished at: 2022-10-09T10:29:10+05:30
[INFO] -----
```

# Step 1: mvn archetype:generate

```
[INFO] Parameter: groupId, Value: org.learning.nifi
[INFO] Parameter: artifactId, Value: example-processor
[INFO] Parameter: nifiVersion, Value: 1.15.3
[INFO] Project created from Archetype in dir: /Users/nadeem/Documents/custom-nifi-development/example-processor
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  15:55 min
[INFO] Finished at: 2022-10-09T10:29:10+05:30
[INFO] -----
[nadeem@Mohammeds-MacBook-Pro custom-nifi-development % ls
example-processor
[nadeem@Mohammeds-MacBook-Pro custom-nifi-development % cd example-processor
[nadeem@Mohammeds-MacBook-Pro example-processor % ls
nifi-example-nar      nifi-example-processors pom.xml
```

# Step 1: mvn archetype:generate

```
.  
├── nifi-example-nar  
├── nifi-example-processors  
└── pom.xml
```

2 directories, 1 file

```
[nadeem@Mohammeds-MacBook-Pro custom-nifi-development % cd example-processor  
[nadeem@Mohammeds-MacBook-Pro example-processor % tree  
.  
├── nifi-example-nar  
│   └── pom.xml  
├── nifi-example-processors  
│   └── pom.xml  
└── src  
    ├── main  
    │   ├── java  
    │   │   └── org  
    │   │       └── learning  
    │   │           └── nifi  
    │   │               └── processors  
    │   │                   └── example  
    │   │                       └── MyProcessor.java  
    │   └── resources  
    │       └── META-INF  
    │           └── services  
    │               └── org.apache.nifi.processor.Processor  
    └── test  
        ├── java  
        │   └── org  
        │       └── learning  
        │           └── nifi  
        │               └── processors  
        │                   └── example  
        │                       └── MyProcessorTest.java  
    └── pom.xml  
20 directories, 6 files  
nadeem@Mohammeds-MacBook-Pro example-processor %
```

# Steps to create custom processor

## Step 2: Deploy the processor

Before we get into further details, let's deploy and visualize the processor

```
$ > mvn clean install
```

```
[nadeem@Mohammeds-MacBook-Pro custom-nifi-development % cd example-processor
[nadeem@Mohammeds-MacBook-Pro example-processor % mvn clean install
[INFO] Scanning for projects...
[INFO] Inspecting build with total of 3 modules...
[INFO] Installing Nexus Staging features:
[INFO]   ... total of 3 executions of maven-deploy-plugin replaced with nexus-staging-maven-plugin
[INFO] -----
[INFO] Reactor Build Order:
[INFO]   -
```

# Step 2: mvn clean install

```
[nadeem@Mohammeds-MacBook-Pro custom-nifi-development % cd example-processor
[nadeem@Mohammeds-MacBook-Pro example-processor % mvn clean install
[INFO] Scanning for projects...
[INFO] Inspecting build with total of 3 modules...
[INFO] Installing Nexus Staging features:
[INFO]   ... total of 3 executions of maven-deploy-plugin replaced with nexus-staging-maven-plugin
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] example-processor                                [pom]
[INFO] nifi-example-processors                         [jar]
[INFO] nifi-example-nar                               [nar]
[INFO]
[INFO] -----< org.learning.nifi:example-processor >-----
[INFO] Building example-processor 1.0                  [1/3]
[INFO] ----- [ pom ] -----
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ example-processor ---
[INFO]
[INFO] --- maven-enforcer-plugin:3.0.0:enforce (enforce-maven-version) @ example-processor ---
[INFO]
[INFO] --- maven-enforcer-plugin:3.0.0:enforce (enforce-maven) @ example-processor ---
[INFO]
[INFO] --- maven-enforcer-plugin:3.0.0:enforce (enforce-no-snapshots) @ example-processor ---
```

# Step 2: mvn clean install

```
[INFO] -----  
[INFO] Reactor Summary for example-processor 1.0:  
[INFO]  
[INFO] example-processor ..... SUCCESS [ 0.920 s]  
[INFO] nifi-example-processors ..... SUCCESS [ 3.264 s]  
[INFO] nifi-example-nar ..... SUCCESS [ 0.389 s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 5.346 s  
[INFO] Finished at: 2022-10-09T11:08:15+05:30  
[INFO] -----  
nadeem@Mohammeds-MacBook-Pro example-processor %
```

# – Steps to create custom processor

## Step 2: Deploy the processor

Before we get into further details, let's deploy and visualize the processor

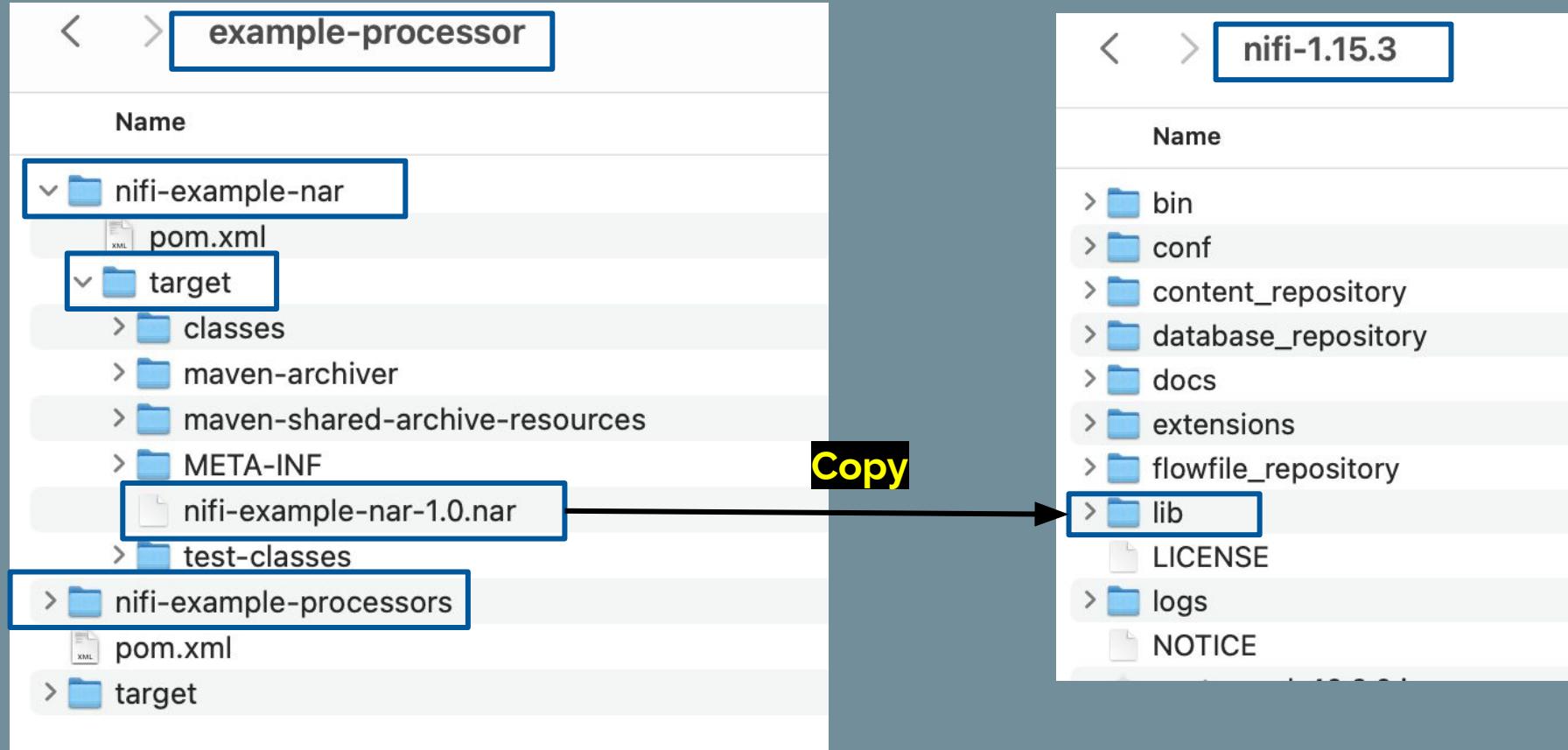
```
$ > mvn clean install
```

Copy processor archive to NiFi

Packing system

- Java Archives → **.jar**
- Java Web Archives → **.war**
- NiFi Archives → **.nar**

# Step 2: Deploy the processor



# – Steps to create custom processor

## Step 2: Deploy the processor

Before we get into further details, let's deploy and visualize the processor

```
$ > mvn clean install
```

Copy processor archive to NiFi

Restart NiFi

# Custom Processor

The screenshot shows the NiFi Flow interface with the 'Add Processor' dialog open. The dialog lists available processors, with 'MyProcessor' selected. The interface includes a toolbar at the top, a left sidebar with 'Operate' and 'NiFi Flow Process Group' sections, and a bottom status bar.

**Add Processor**

Type	Version	Tags
MyProcessor	1.0	example

Source: all groups

Displaying 1 of 299

Tags: example

Processor List:

- amazon attributes
- avro aws consume
- csv database
- delete fetch get
- hadoop ingest
- insert json listen
- logs message
- pubsub put query
- record restricted
- source text
- update

MyProcessor 1.0 org.learning.nifi - nifi-example-nar

Provide a description

CANCEL ADD

NiFi Flow

# Custom Processor

The screenshot shows the NiFi user interface with a central canvas and various toolbars and panels.

**Toolbar:** Includes icons for Create, Insert, Delete, Copy, Paste, and others, along with a search bar and a date/time stamp (12:01:14 IST).

**Navigator Panel:** Shows a tree view of the flow, with a purple node and a green node visible.

**Operate Panel:** Displays the current NiFi Flow Process Group (df0140a1-0181-1000-9eae-a1116a1a928d) and provides options to Operate, Add, and Delete components.

**Processor Overview:** A detailed view of the "MyProcessor" component, which is a MyProcessor 1.0 instance from org.learning.nifi - nifi-example-nar. The processor has the following metrics:

Metric	Value	Time Period
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

# Custom Processor

The screenshot shows the NiFi Flow interface with a central configuration dialog open over the main workspace.

**Toolbar:** Standard NiFi icons for flow control (e.g., start, stop, pause), monitoring (e.g., metrics, logs), and system navigation.

**Metrics Bar:** Displays various system metrics including flow count (0), byte count (0 / 0 bytes), and error count (2).

**Configure Processor Dialog:** Titled "Configure Processor | MyProcessor 1.0".

**Status:** Shows an "Invalid" status with a yellow warning icon.

**Tab Navigation:** SETTINGS, SCHEDULING, PROPERTIES, COMMENTS. The PROPERTIES tab is selected.

**Properties Table:** A table titled "Required field" with one row.

Property	Value
My property	<small>?</small> No value set

**Buttons:** CANCEL and APPLY at the bottom right of the dialog.

**Left Sidebar:** "Operate" section for "MyProcessor Processor" (ID: bb70c215-0183-1000-a39b-fb2d97799d91). It includes icons for settings, metrics, logs, and delete, along with a copy icon.

**Bottom Bar:** "NiFi Flow" text.

# Project Setup in IntelliJ

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "example-processor > nifi-example-processors > src > main > java > org > learning > nifi > processors > example > MyProcessor". A yellow box highlights the project structure.
- Code Editor:** The main editor shows the `MyProcessor.java` file. It contains annotations like `@Tags`, `@CapabilityDescription`, and `@SeeAlso`. The class definition is:

```
public class MyProcessor extends AbstractProcessor {
```
- Properties Dialog:** A modal dialog titled "Configure Processor | MyProcessor 1.0" is open. It has tabs for "SETTINGS", "SCHEDULING", "PROPERTIES", and "COMMENTS". The "PROPERTIES" tab is selected, showing a table with one row: "My property" (Property) and "No value set" (Value). A yellow box highlights this dialog.
- Tool Window:** The bottom right shows the "Maven" tool window with a table of build metrics:

	In	Out	Tasks/Time
0 (0 bytes)	0 (0 bytes)	5 min	
Read/Write 0 bytes / 0 bytes	0 (0 bytes)	5 min	
0 (0 bytes)	0 (0 bytes)	5 min	
0:00:00.000	0:00:00.000	5 min	

A yellow box highlights this window, and an arrow points from the "Properties" dialog to it.
- Log:** The bottom pane shows the log output with repeated entries:

```
[INFO] Inspecting build with total of 1 modules...
[INFO] Installing Nexus Staging features:
[INFO] ... total of 1 executions of maven-deploy-plugin replaced with nexus-staging-maven-plugin
[INFO] Inspecting build with total of 1 modules...
[INFO] Installing Nexus Staging features:
[INFO] ... total of 1 executions of maven-deploy-plugin replaced with nexus-staging-maven-plugin
[INFO] Inspecting build with total of 1 modules...
[INFO] Installing Nexus Staging features:
[INFO] ... total of 1 executions of maven-deploy-plugin replaced with nexus-staging-maven-plugin
```
- Bottom Bar:** The bottom bar includes icons for Version Control, TODO, Problems, Terminal, Services, Build, and Dependencies. A status message at the bottom left says "Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Maven library shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago)". The bottom right shows the time as 48:14 and the terminal encoding as UTF-8.

```
public class MyProcessor extends AbstractProcessor {  
  
    1 usage  
    public static final PropertyDescriptor MY_PROPERTY = new PropertyDescriptor  
        .Builder().name("MY_PROPERTY")  
        .displayName("My property")  
        .description("Example Property")  
        .required(true)  
        .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)  
        .build();  
  
    1 usage  
    public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()  
        .name("MY_RELATIONSHIP")  
        .description("Example relationship")  
        .build();  
  
    5 usages  
    private List<PropertyDescriptor> descriptors;  
  
    5 usages  
    private Set<Relationship> relationships;  
  
    @Override  
    protected void init(final ProcessorInitializationContext context) {...}  
  
    @Override  
    public Set<Relationship> getRelationships() { return this.relationships; }  
  
    2 usages  
    @Override  
    public final List<PropertyDescriptor> getSupportedPropertyDescriptors() { return descriptors; }
```

All properties

All relationships

```
public class MyProcessor extends AbstractProcessor {
```

1 usage

```
public static final PropertyDescriptor MY_PROPERTY = new PropertyDescriptor
    .Builder().name("MY_PROPERTY")
    .displayName("My property")
    .description("Example Property")
    .required(true)
    .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)
    .build();
```

My property

1 usage

```
public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()
    .name("MY_RELATIONSHIP")
    .description("Example relationship")
    .build();
```

5 usages

```
private List<PropertyDescriptor> descriptors;
```

5 usages

```
private Set<Relationship> relationships;
```

```
@Override
```

```
protected void init(final ProcessorInitializationContext context) {...}
```

```
@Override
```

```
public Set<Relationship> getRelationships() { return this.relationships; }
```

2 usages

```
@Override
```

```
public final List<PropertyDescriptor> getSupportedPropertyDescriptors() { return descriptors; }
```

MyProcessor		
MyProcessor 1.0		
org.learning.nifi - nifi-example-nar		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

No value set



Example Property

Expression language scope: Not Supported

Sensitive property: false Value

```
public class MyProcessor extends AbstractProcessor {  
  
    1 usage  
    public static final PropertyDescriptor MY_PROPERTY = new PropertyDescriptor  
        .Builder().name("MY_PROPERTY")  
        .displayName("My property")  
        .description("Example Property")  
        .required(true)  
        .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)  
        .build();
```

```
1 usage  
public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()  
    .name("MY_RELATIONSHIP")  
    .description("Example relationship")  
    .build();
```

```
5 usages  
private List<PropertyDescriptor> descriptors;  
  
5 usages  
private Set<Relationship> relationships;  
  
@Override  
protected void init(final ProcessorInitializationContext context) {...}  
  
@Override  
public Set<Relationship> getRelationships() { return this.relationships; }
```

```
2 usages  
@Override  
public final List<PropertyDescriptor> getSupportedPropertyDescriptors() { return descriptors; }
```

MENTS

Automatically Terminate Relationships ?  
 **MY\_RELATIONSHIP**  
Example relationship

```
public class MyProcessor extends AbstractProcessor {  
  
    1 usage  
    public static final PropertyDescriptor MY_PROPERTY = new PropertyDescriptor  
        .Builder().name("MY_PROPERTY")  
        .displayName("My property")  
        .description("Example Property")  
        .required(true)  
        .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)  
        .build();  
  
    1 usage  
    public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()  
        .name("MY_RELATIONSHIP")  
        .description("Example relationship")  
        .build();  
  
    5 usages  
    private List<PropertyDescriptor> descriptors;  
  
    5 usages  
    private Set<Relationship> relationships;  
  
    @Override  
    protected void init(final ProcessorInitializationContext context) {...}  
  
    @Override  
    public Set<Relationship> getRelationships() { return this.relationships; }  
  
    2 usages  
    @Override  
    public final List<PropertyDescriptor> getSupportedPropertyDescriptors() { return descriptors; }
```

Storing all Properties and relationships

# Initialize method

```
public class MyProcessor extends AbstractProcessor {  
  
    1 usage  
    public static final PropertyDescriptor MY_PROPERTY = new PropertyDescriptor  
        .Builder().name("MY_PROPERTY")  
        .displayName("My property")  
        .description("Example Property")  
        .required(true)  
        .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)  
        .build();  
  
    1 usage  
    public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()  
        .name("MY_RELATIONSHIP")  
        .description("Example relationship")  
        .build();  
  
    5 usages  
    private List<PropertyDescriptor> descriptors;  
  
    5 usages  
    private Set<Relationship> relationships;  
  
    @Override  
    protected void init(final ProcessorInitializationContext context) {...}  
  
    @Override  
    public Set<Relationship> getRelationships() { return this.relationships; }  
  
    2 usages  
    @Override  
    public final List<PropertyDescriptor> getSupportedPropertyDescriptors() { return descriptors; }
```

This method is called when processor is first dropped on the canvas

Initializing with properties and relationships

```
@Override  
protected void init(final ProcessorInitializationContext context) {...}
```



```
@Override  
protected void init(final ProcessorInitializationContext context) {  
    descriptors = new ArrayList<>();  
    descriptors.add(MY_PROPERTY);  
    descriptors = Collections.unmodifiableList(descriptors);  
  
    relationships = new HashSet<>();  
    relationships.add(MY_RELATIONSHIP);  
    relationships = Collections.unmodifiableSet(relationships);  
}
```

**This method is called when processor is first dropped on the canvas**

**Initializing with properties and relationships**

```
@Override
protected void init(final ProcessorInitializationContext context) {
    descriptors = new ArrayList<>();
    descriptors.add(MY_PROPERTY);
    descriptors = Collections.unmodifiableList(descriptors);

    relationships = new HashSet<>();
    relationships.add(MY_RELATIONSHIP);
    relationships = Collections.unmodifiableSet(relationships);
}

@Override
public Set<Relationship> getRelationships() { return this.relationships; }

2 usages
@Override
public final List<PropertyDescriptor> getSupportedPropertyDescriptors() { return descriptors; }

@OnScheduled
public void onScheduled(final ProcessContext context) {}

@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {...}
}
```

# Processor Lifecycle

# Processor Lifecycle

@OnAdded

@OnRemoved

@OnScheduled

@Unscheduled

@Stopped

NiFi Framework  
calls your methods  
based on annotation  
defined

```
@OnScheduled  
public void onScheduled(final ProcessContext context) {}
```

# Processor Lifecycle

@OnAdded

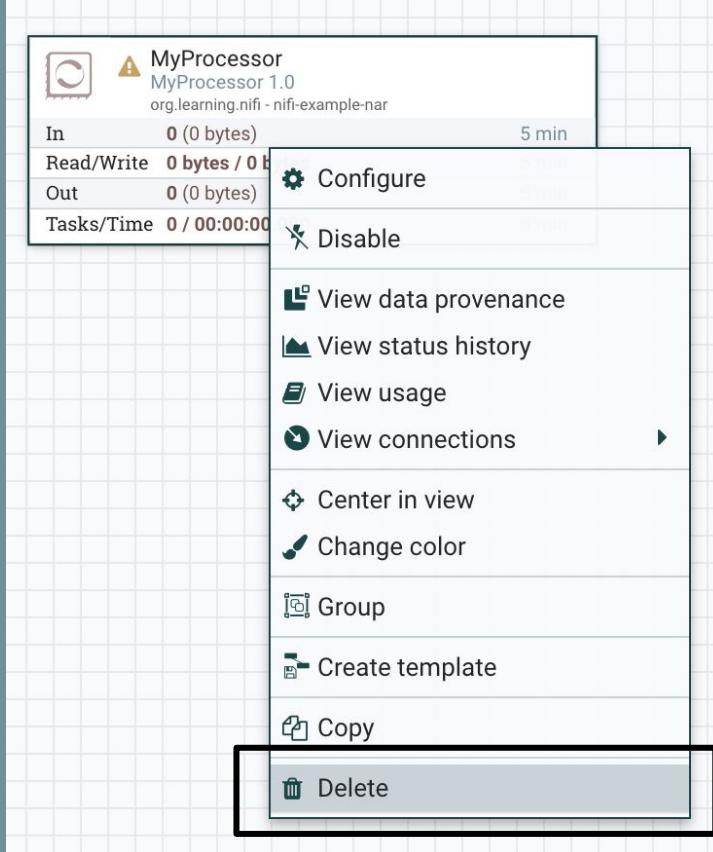
The screenshot shows the Apache NiFi user interface. At the top, there is a toolbar with various icons for actions like start, stop, and refresh. Below the toolbar, a header bar displays metrics: 1 process, 88 / 32.09 KB, 0 errors, 0 warnings, 16 pending tasks, 28 scheduled tasks, 8 active tasks, 0 failed tasks, and 0 completed tasks. On the left, there is a sidebar with a 'Navigate' button and search/filter tools. The main area is a canvas where processors are placed. A large black arrow points from the top center of the interface down to a specific processor card on the canvas. The processor card for 'MyProcessor' is shown with the following details:

	<b>MyProcessor</b> MyProcessor 1.0 org.learning.nifi - nifi-example-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

At the bottom of the interface, there is an 'Operate' button.

# Processor Lifecycle

@OnRemoved



# Processor Lifecycle

@OnScheduled

 MyProcessor	MyProcessor 1.0	org.learning.nifi - nifi-example-nar
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

SCHEDULING

Scheduling Strategy ?

Timer driven

Concurrent Tasks ?

1

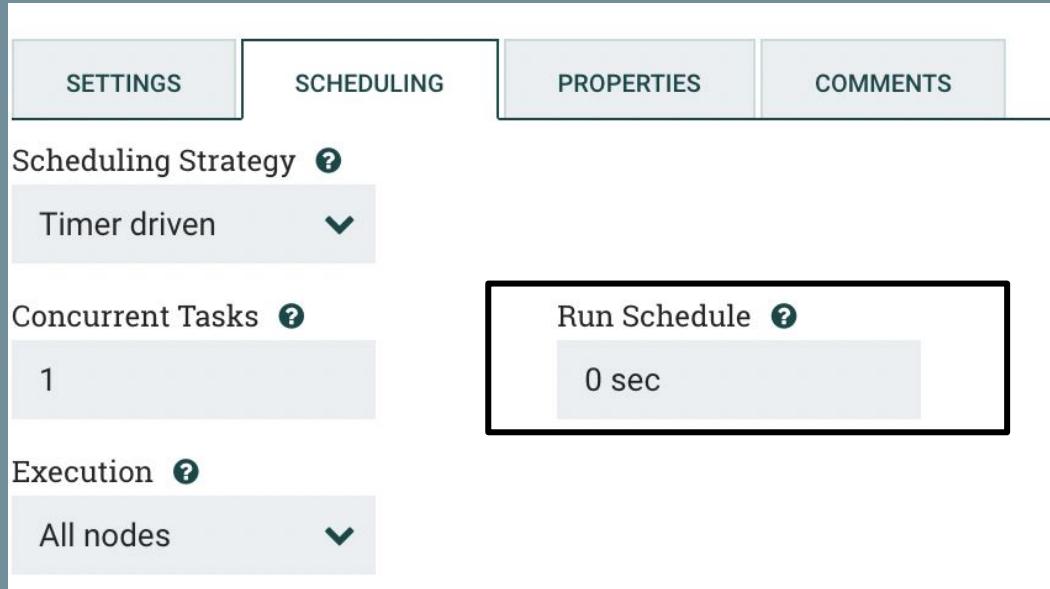
Run Schedule ?

0 sec

Execution ?

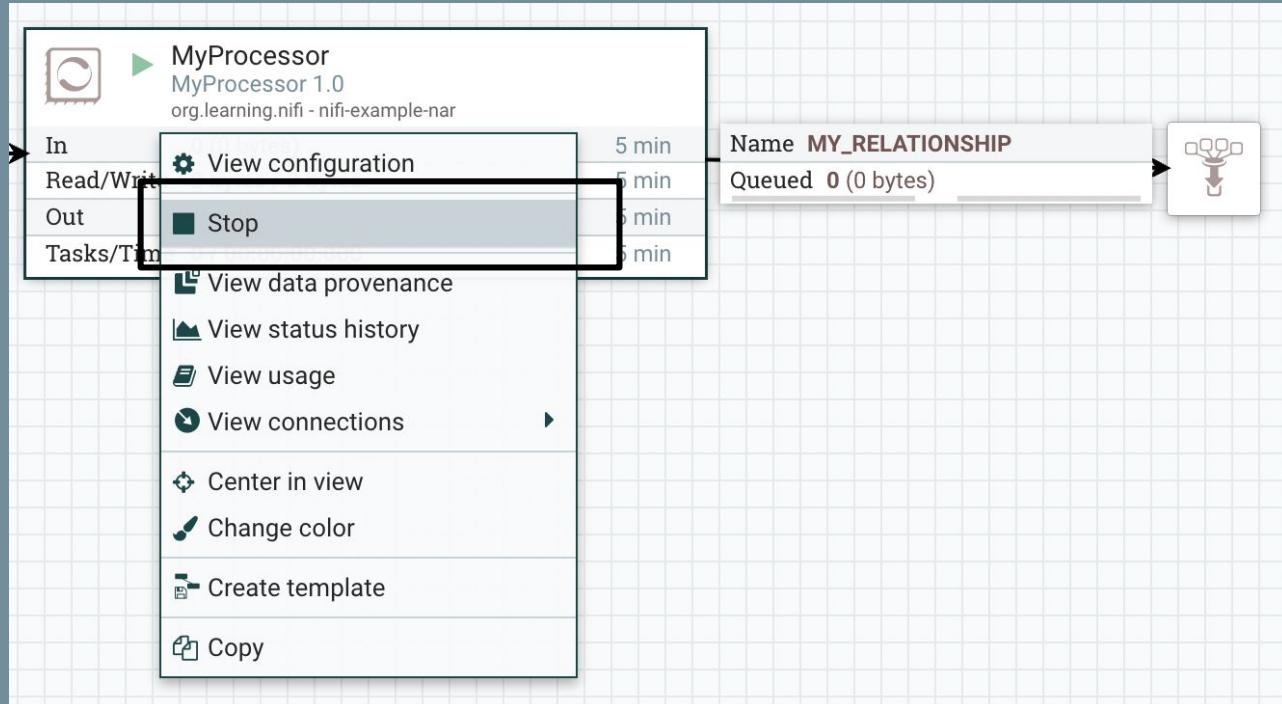
All nodes

SETTINGS PROPERTIES COMMENTS



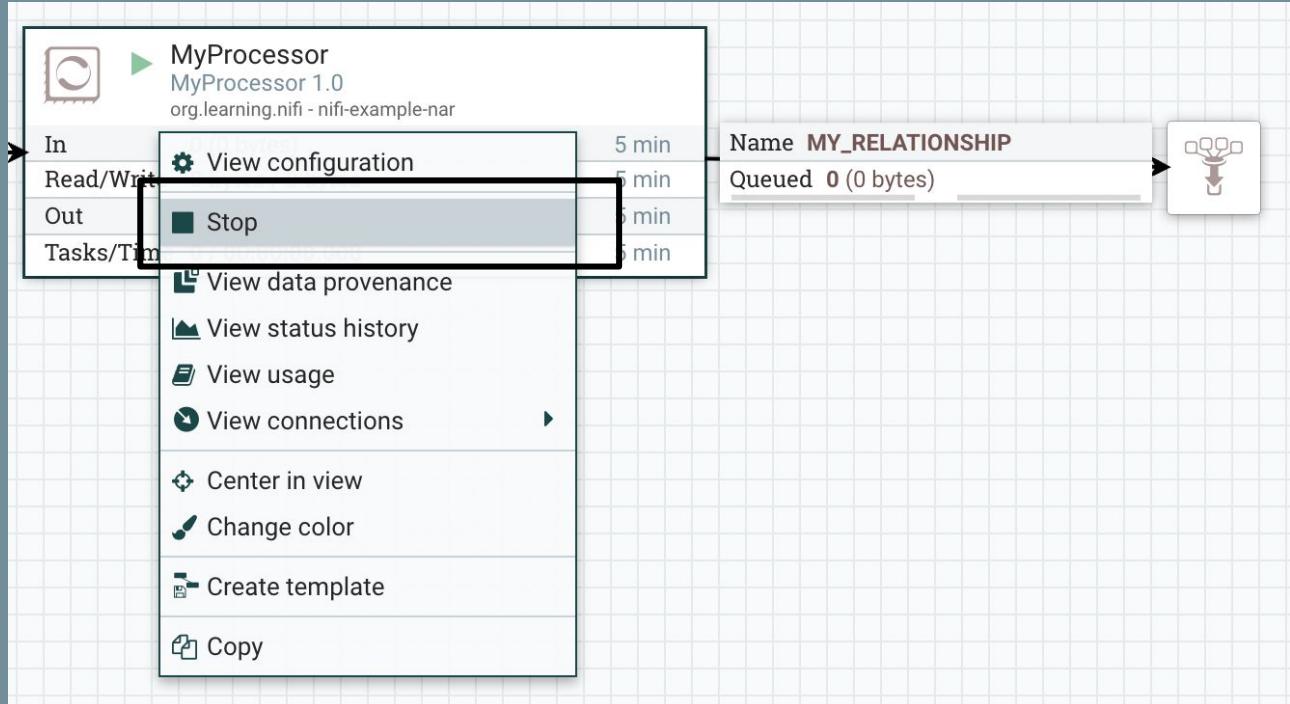
# Processor Lifecycle

@OnStopped



# Processor Lifecycle

@OnUnscheduled



# Processor Lifecycle

```
@OnAdded
public void onAdded() {
    System.out.println("On Added is called");
}

@OnScheduled
public void onScheduled(final ProcessContext context) {
    System.out.println("On Scheduled is called");
}

@OnUnscheduled
public void onUnscheduled() {
    System.out.println("On Unscheduled is called");
}

@OnStopped
public void onStopped() {
    System.out.println("On Stopped is called");
}

@OnRemoved
public void onRemoved() {
    System.out.println("On Removed is called");
}
```

—

# onTrigger method

# OnTrigger method in Processor

```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
    // TODO implement  
}
```

# OnTrigger method in Processor

```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
    // TODO implement  
}
```

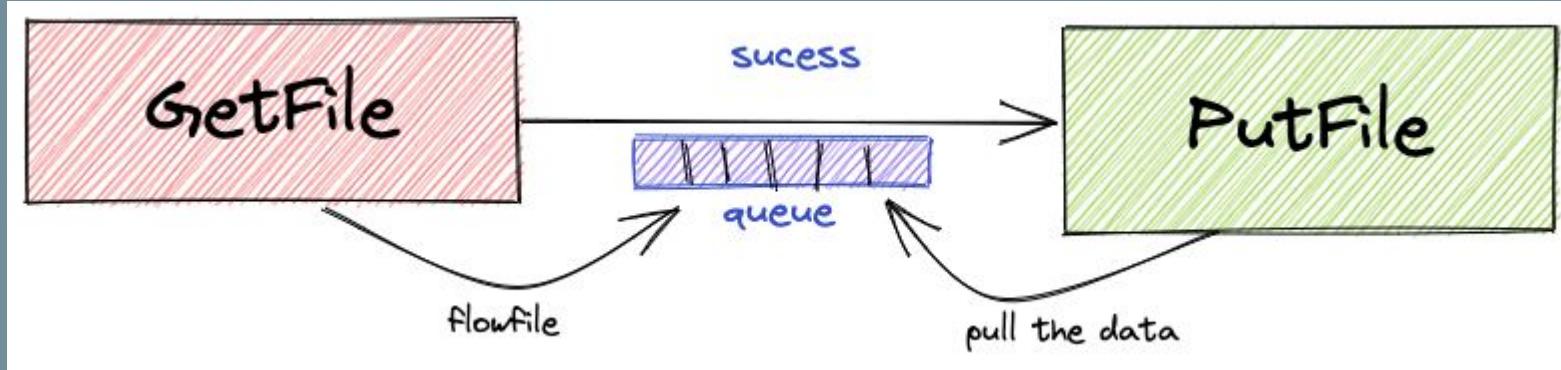
OnTrigger method is called only when the processor is scheduled (i.e started) and there is exist work for a processor to do.

A work exists for a processor when there is an incoming a flowfile

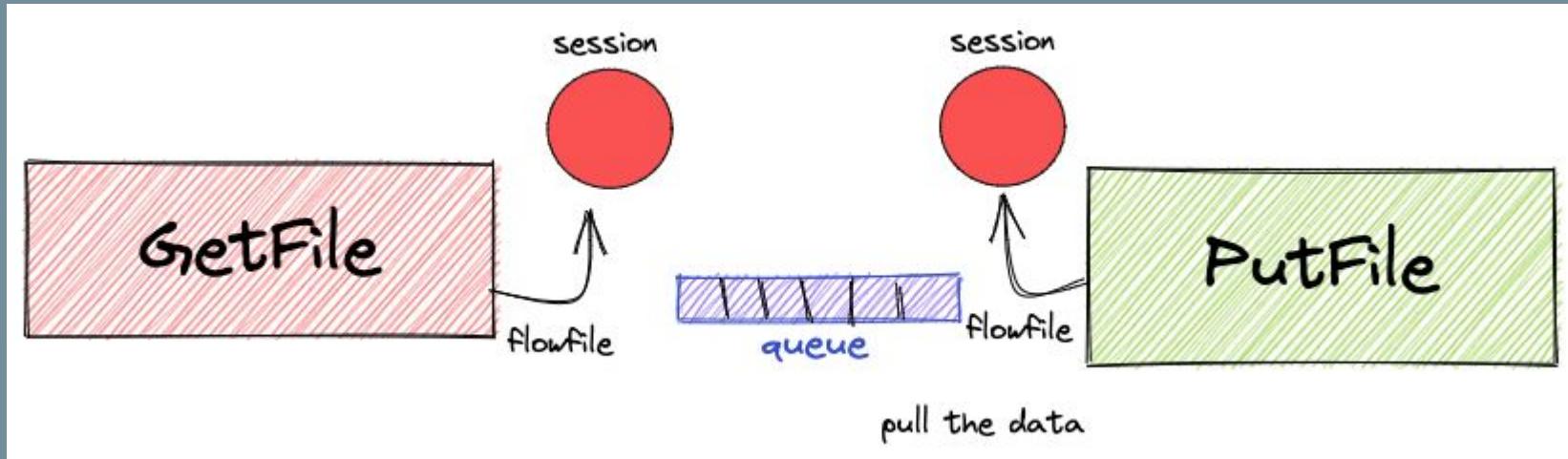
Or when a there is no incoming connection, then also it is called only when @TriggerEmpty annotation is present

# Understanding Session

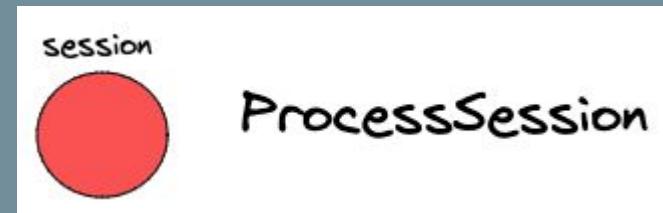
# OnTrigger method in Processor



# OnTrigger method in Processor

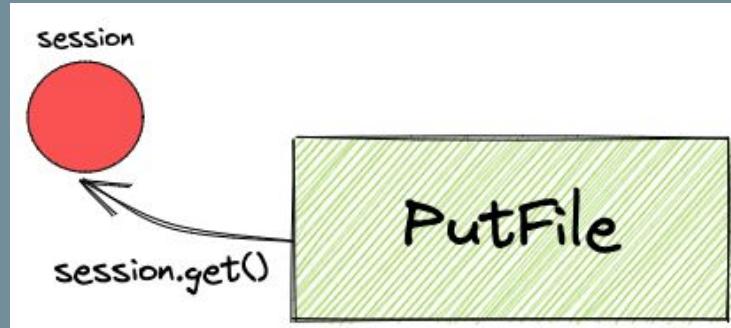


```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
    // TODO implement  
}
```



# OnTrigger method in Processor

```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
    // TODO implement  
}
```



# OnTrigger method in Processor

```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
  
    // transferring flowfile to next processor  
    session.transfer(flowFile, MY_RELATIONSHIP);  
}
```



# OnTrigger method in Processor

```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
  
    // transferring flowfile to next processor  
    session.transfer(flowFile, MY_RELATIONSHIP);  
}
```



# Reading incoming flowfile content

# Reading flowfile content

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

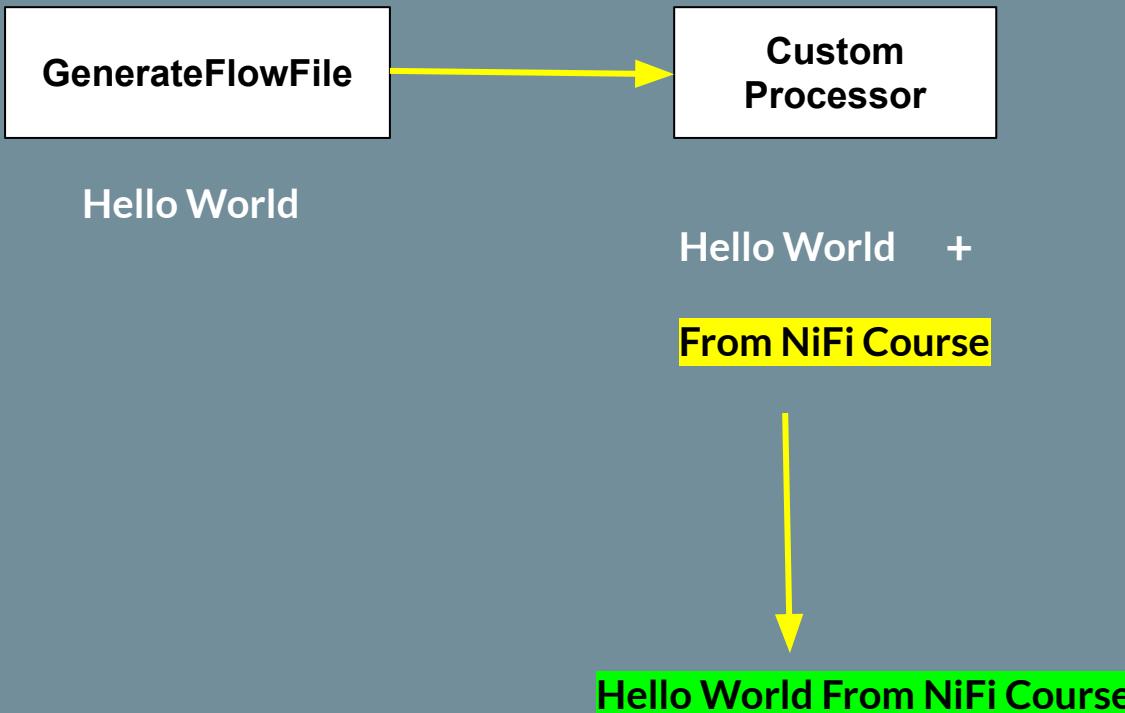
    final StringBuilder sb = new StringBuilder();

    // reading flowfile and storing as content
    session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

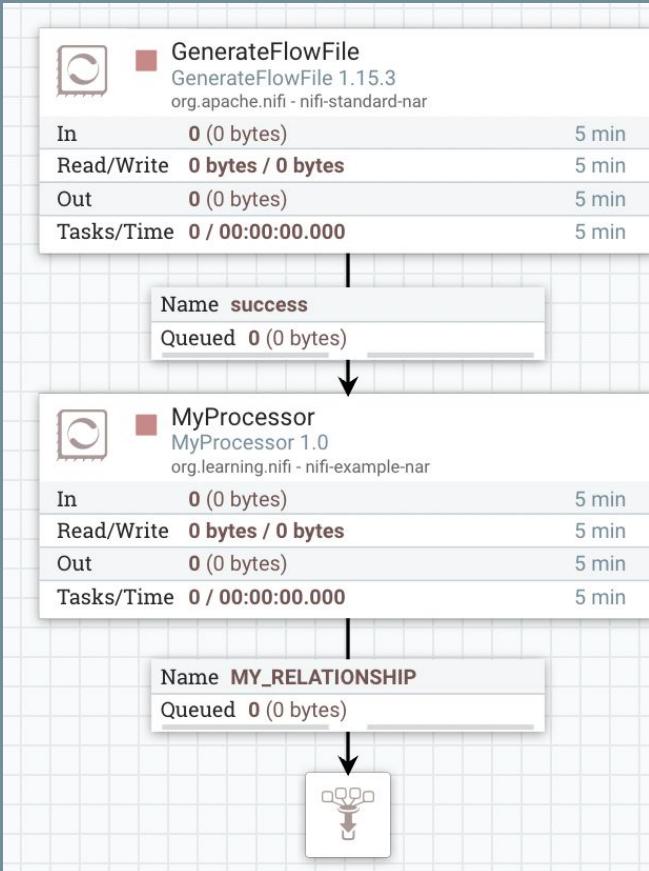
    final String content = sb.toString();

    // transferring flowfile to next processor
    session.transfer(flowFile, MY_RELATIONSHIP);
}
```

# What are we going to build



# Hello World Processor



**Hello World**

**Hello World From Nifi Course**

—

# Concatenating incoming flowfile

# Concating incoming FlowFile

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

    final StringBuilder sb = new StringBuilder();

    // reading flowfile and storing as content
    session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

    final String content = sb.toString();

    // performing concat on incoming content
    final String resultantString = content.concat( str: " From NiFi Custom Processor");
}
```

Writing result back to  
flowfile

# Writing the result into flowfile back

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

    final StringBuilder sb = new StringBuilder();

    // reading flowfile and storing as content
    session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

    final String content = sb.toString();

    // performing concat on incoming content
    final String resultantString = content.concat(" From NiFi Custom Processor");

    // writing the resultant string back to the flowfile
    flowFile = session.write(flowFile, out -> out.write(resultantString.getBytes(Charset.defaultCharset())));

    // transferring flowfile to next processor
    session.transfer(flowFile, MY_RELATIONSHIP);
}
```

# Reading from Properties

# Reading from Processor Properties

```
// performing concat on incoming content  
final String resultantString = content.concat(" From NiFi Custom Processor");
```

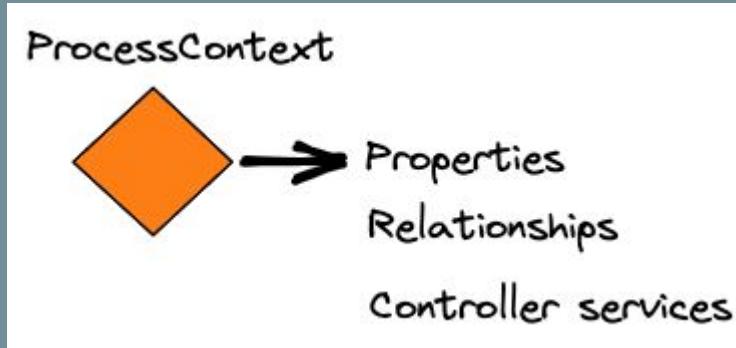
Required field



Property	Value
My property	No value set

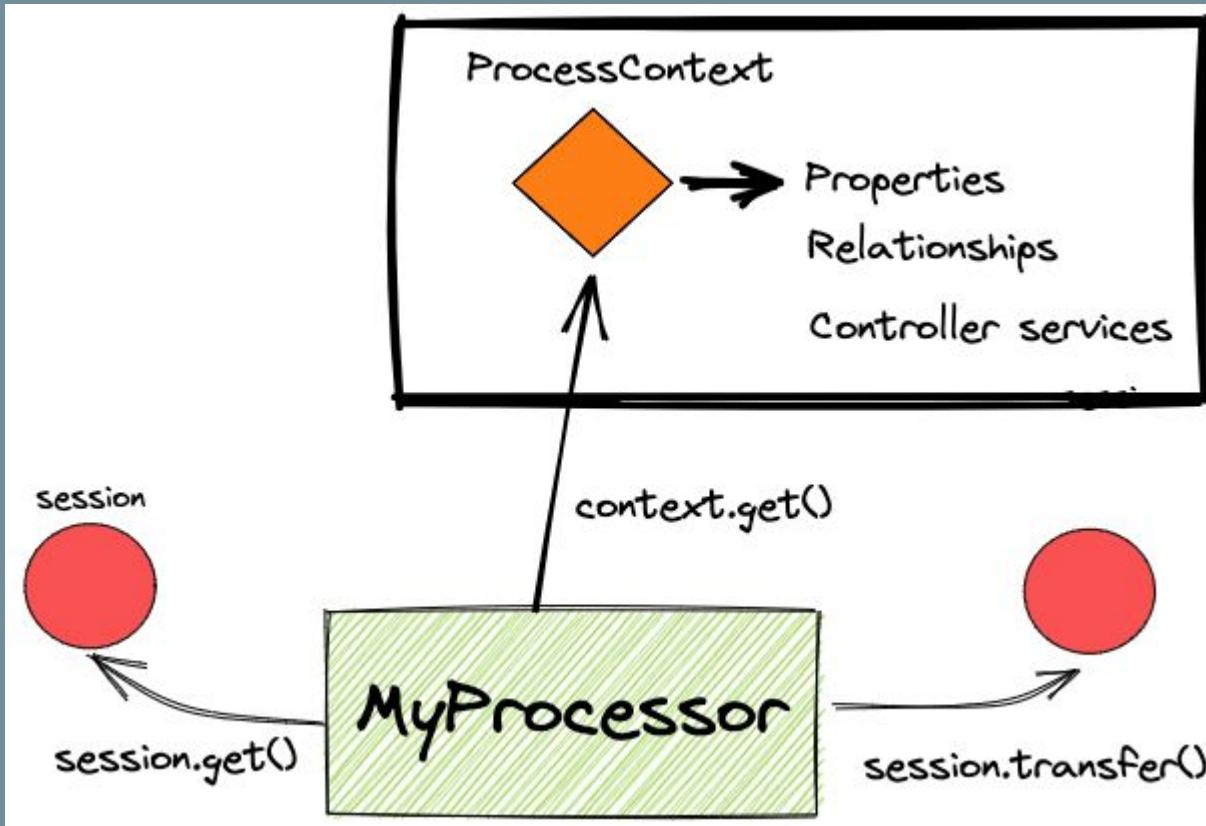
# Understanding ProcessContext

# ProcessContext



```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) -  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }
```

# ProcessContext



# ProcessContext

```
@Override  
public void onTrigger(final ProcessContext context, final ProcessSession session) {  
  
    FlowFile flowFile = session.get();  
    if ( flowFile == null ) {  
        return;  
    }  
  
    // getting MY_PROPERTY property value  
    final String property = context.getProperty(MY_PROPERTY).getValue();
```

Property

Value

My property



No value set



# ProcessContext

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

    final StringBuilder sb = new StringBuilder();

    // reading flowfile and storing as content
    session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

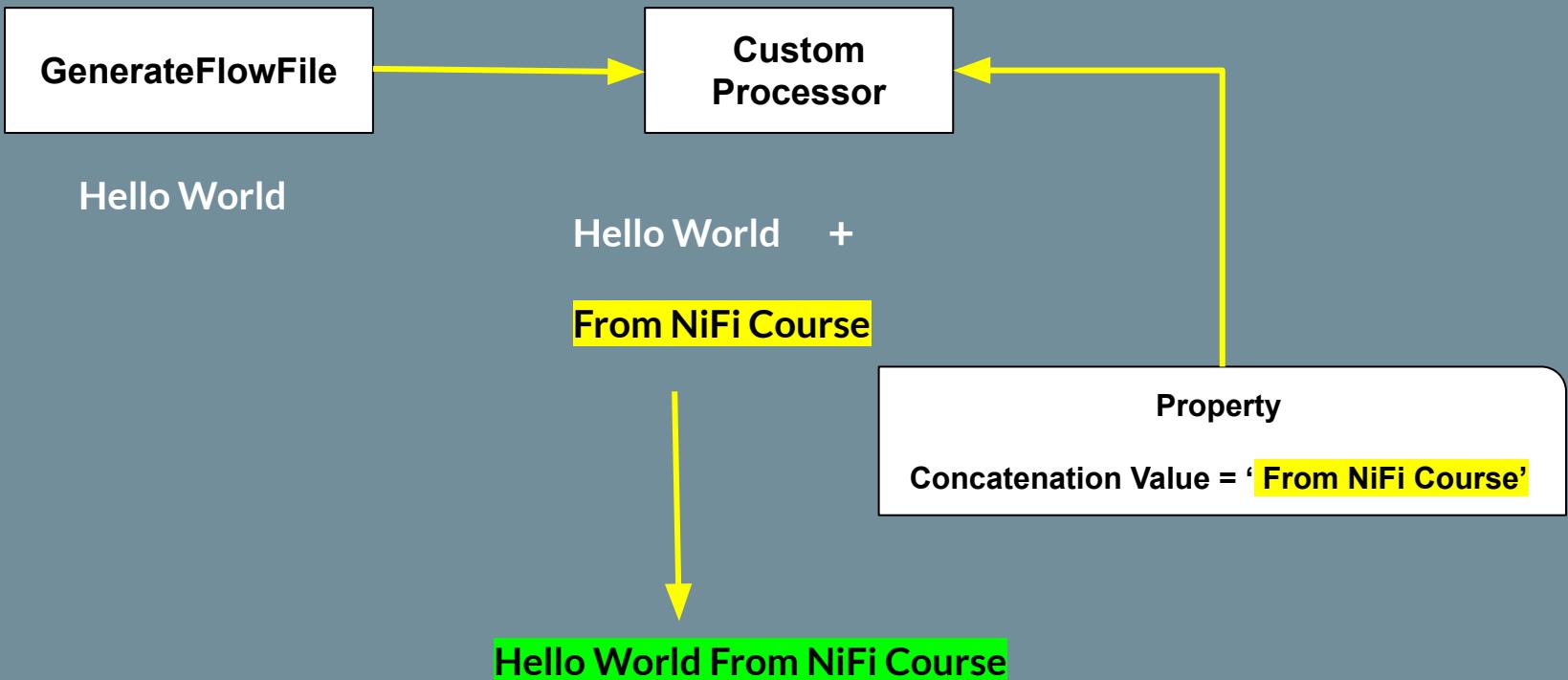
    final String content = sb.toString();

    // performing concat on incoming content
    final String resultantString = content.concat(" From NiFi Custom Processor");

    // writing the resultant string back to the flowfile
    flowFile = session.write(flowFile, out -> out.write(resultantString.getBytes(Charset.defaultCharset())));

    // transferring flowfile to next processor
    session.transfer(flowFile, MY_RELATIONSHIP);
}
```

# Reading from property



# ProcessContext

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

    // getting MY_PROPERTY property value
    final String property = context.getProperty(MY_PROPERTY).getValue();

    final StringBuilder sb = new StringBuilder();

    // reading flowfile and storing as content
    session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

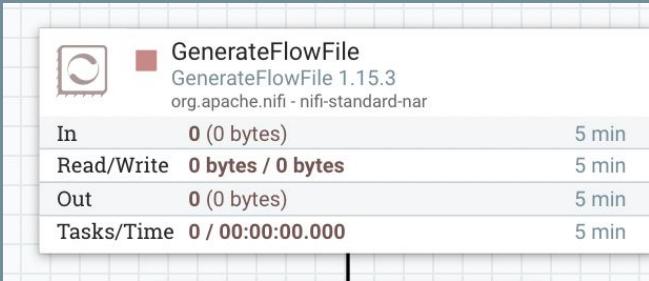
    final String content = sb.toString();

    // performing concat on incoming content
    final String resultantString = content.concat(property);

    // writing the resultant string back to the flowfile
    flowFile = session.write(flowFile, out -> out.write(resultantString.getBytes(Charset.defaultCharset())));

    // transferring flowfile to next processor
    session.transfer(flowFile, MY_RELATIONSHIP);
}
```

# Hello World Processor



Hello World



Hello World **From Nifi Course**

A screenshot of the NiFi interface showing the properties of the MyProcessor processor. The properties table has two rows: "Property" and "Value". The first row contains "My property" and "From NiFi Course". There is also a "Required field" label above the table. The top right corner of the interface has a checkbox and a plus sign button.

Property	Value
My property	From NiFi Course

# Refactoring My\_Property

# Renaming My Property

Required field		<input checked="" type="checkbox"/>	<input type="button" value="+"/>
Property	Value		
My property	From NiFi Course		

2 usages

```
public static final PropertyDescriptor MY_PROPERTY = new PropertyDescriptor
    .Builder().name("MY_PROPERTY")
    .displayName("My property")
    .description("Example Property")
    .required(true)
    .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)
    .build();
```

# Renaming My Property

Required field



Property	Value	Concatenates provided value to the incoming flowfile EDULING PROPERTIES COMMENTS Expression language scope: Not Supported Sensitive property: false History: • From NiFi Course - 01/23/2023 01:46:28 IST (anonymous)
Concatenation Value	From NiFi Course	

2 usages

```
public static final PropertyDescriptor CONCAT_PROPERTY = new PropertyDescriptor
    .Builder().name("CONCAT_PROPERTY")
    .displayName("Concatenation Value")
    .description("Concatenates provided value to the incoming flowfile")
    .required(true)
    .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)
    .build();
```

# Refactoring My\_Relationship

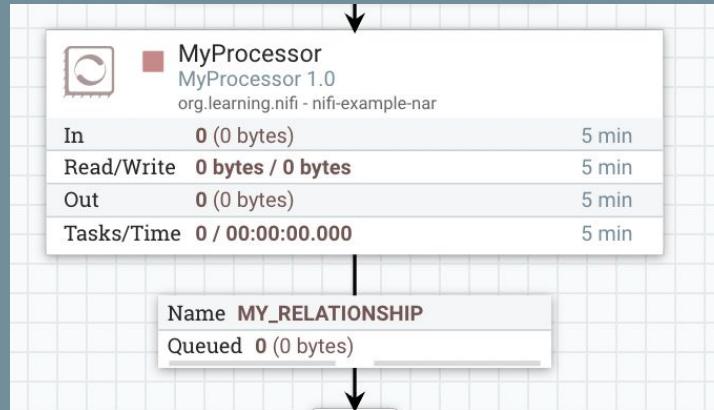
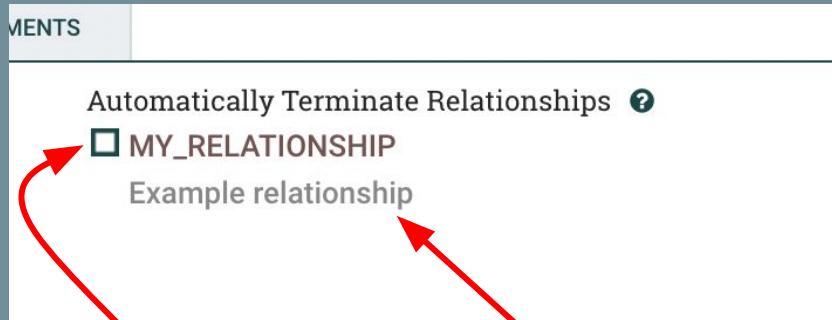
# Renaming My Relationship

MENTS

Automatically Terminate Relationships ⓘ

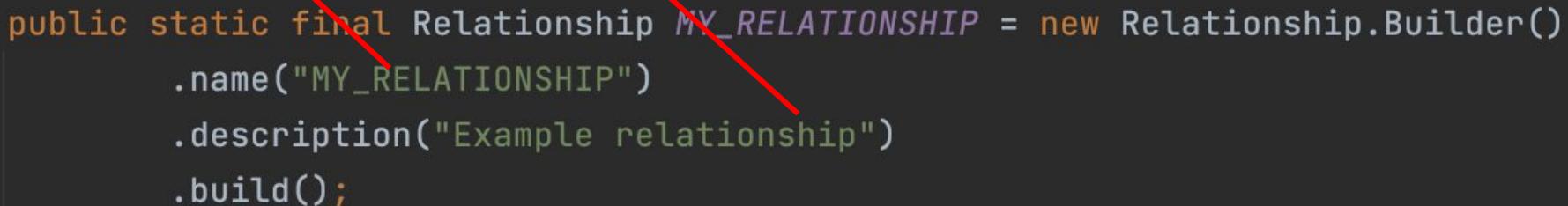
**MY\_RELATIONSHIP**

Example relationship



2 usages

```
public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()  
    .name("MY_RELATIONSHIP")  
    .description("Example relationship")  
    .build();
```

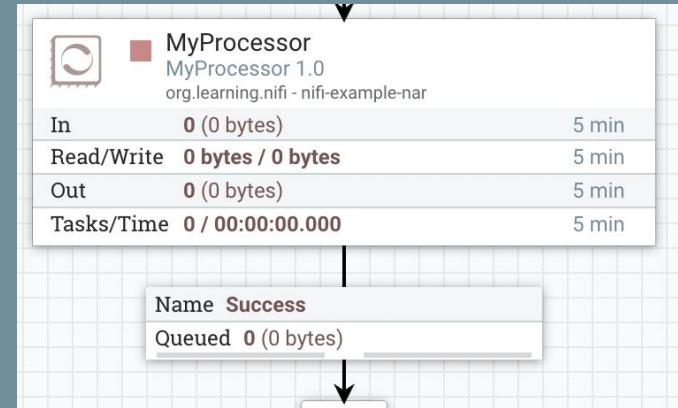


# Renaming My Relationship

Automatically Terminate Relationships [?](#)

Success

Successfully processed flowfiles are transferred to this relationship



# Renaming MY\_Processor

# Renaming My Relationship

	<b>MyProcessor</b> MyProcessor 1.0 org.learning.nifi - nifi-example-nar
In	<b>0 (0 bytes)</b>
Read/Write	<b>0 bytes / 0 bytes</b>
Out	<b>0 (0 bytes)</b>
Tasks/Time	<b>0 / 00:00:00.000</b>

2 usages

```
@Tags({"example"})  
@CapabilityDescription("Provide a description")  
@SeeAlso({})  
@ReadsAttributes({@ReadsAttribute(attribute="", description="")})  
@WritesAttributes({@WritesAttribute(attribute="", description="")})  
public class MyProcessor extends AbstractProcessor {
```

2 usages

# Renaming My Relationship

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure for 'example-processor-bkp' located at '~/Documents/custom-nifi-development'. The structure includes 'nifi-example-nar', 'nifi-example-processors' (which contains 'src' with 'main/java/org.learning.nifi.processors.example/ConcatText'), 'resources/META-INF.services/org.apache.nifi.processor.Processor', 'test' (with 'java/org.learning.nifi.processors.example/ConcatTextTest'), and 'target' and 'pom.xml' files. The 'External Libraries' section is also visible.

The code editor shows the Java class 'ConcatText' from the 'src/main/java/org.learning.nifi.processors.example' package. The class extends 'AbstractProcessor'. It imports various Java utilities and annotations like @Tags, @CapabilityDescription, @SeeAlso, @ReadsAttributes, and @WritesAttributes. The class has two usages of a static finalPropertyDescriptor named 'CONCAT\_PROPERTY'.

```
35 import org.apache.nifi.processor.util.StandardValidators;
36
37 import java.nio.charset.Charset;
38 import java.util.ArrayList;
39 import java.util.Collections;
40 import java.util.HashSet;
41 import java.util.List;
42 import java.util.Set;
43
44 /**
45  * @Tags({"example"})
46  * @CapabilityDescription("Provide a description")
47  * @SeeAlso({})
48  * @ReadsAttributes({@ReadsAttribute(attribute="", description(""))})
49  * @WritesAttributes({@WritesAttribute(attribute="", description(""))})
50
51     public static final PropertyDescriptor CONCAT_PROPERTY = new PropertyDescriptorBuilder()
52         .name("CONCAT_PROPERTY")
53         .description("The name of the relationship to change")
54         .required(true)
55         .validator(StandardValidators.NON_EMPTY_VALIDATOR)
56         .build();
```

# Another Important Change

The screenshot shows a Java project structure in a dark-themed IDE. The project is named 'example-processor-bkp' and contains several sub-folders and files:

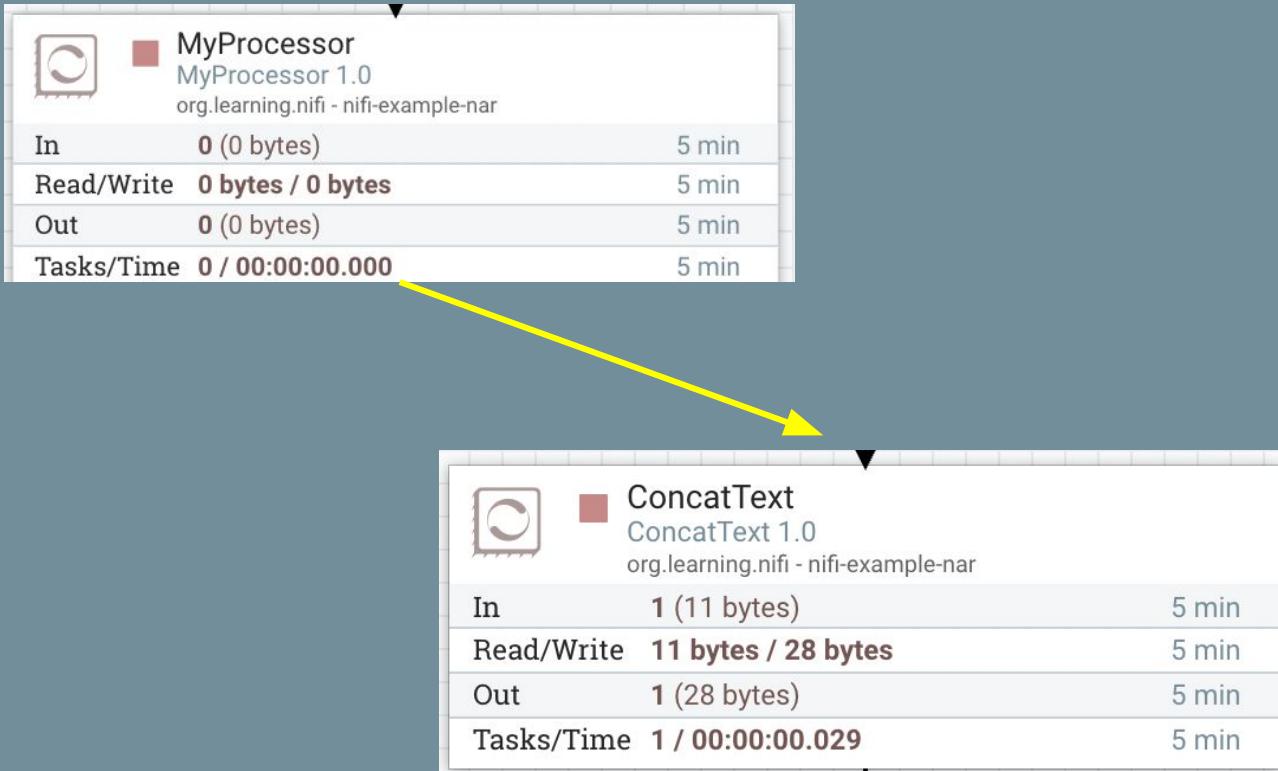
- Project**: Shows the overall project structure.
- src**: Contains:
  - main**: Contains a **java** folder with a file named **ConcatText**.
  - resources**: Contains a **META-INF.services** folder with a file named **org.apache.nifi.processor.Processor**.
- test**: Contains:
  - java**: Contains a **org.learning.nifi.processors.example** folder with a file named **ConcatTextTest**.
- target**: Contains **pom.xml**.

The code editor on the right displays the content of the **org.apache.nifi.processor.Processor** file. The code is a standard Apache license notice followed by the package declaration:

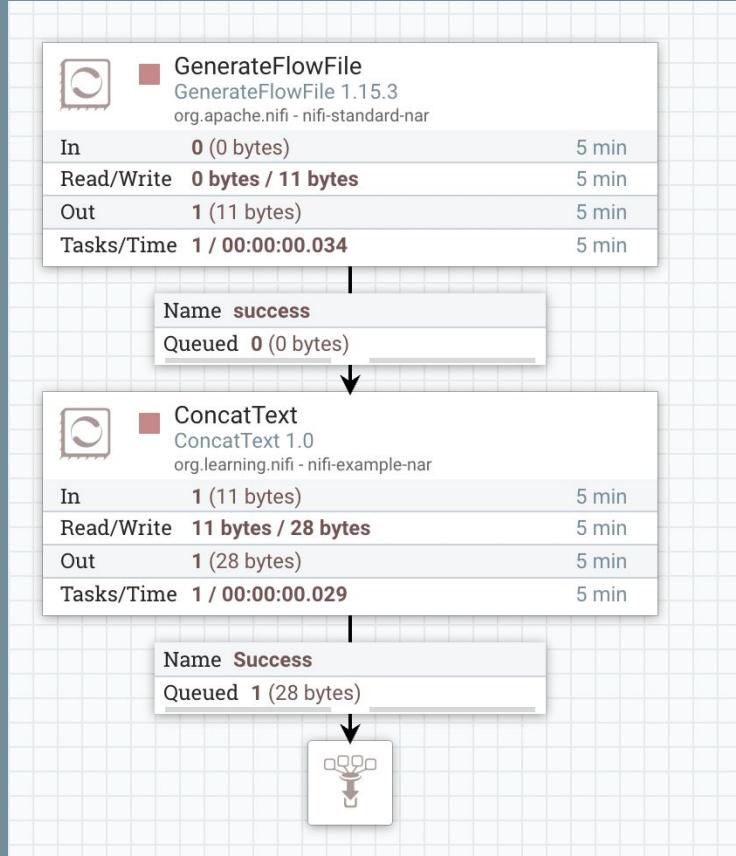
```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

org.learning.nifi.processors.example.ConcatText
```

# Renaming My Relationship



# ConcatText Processor



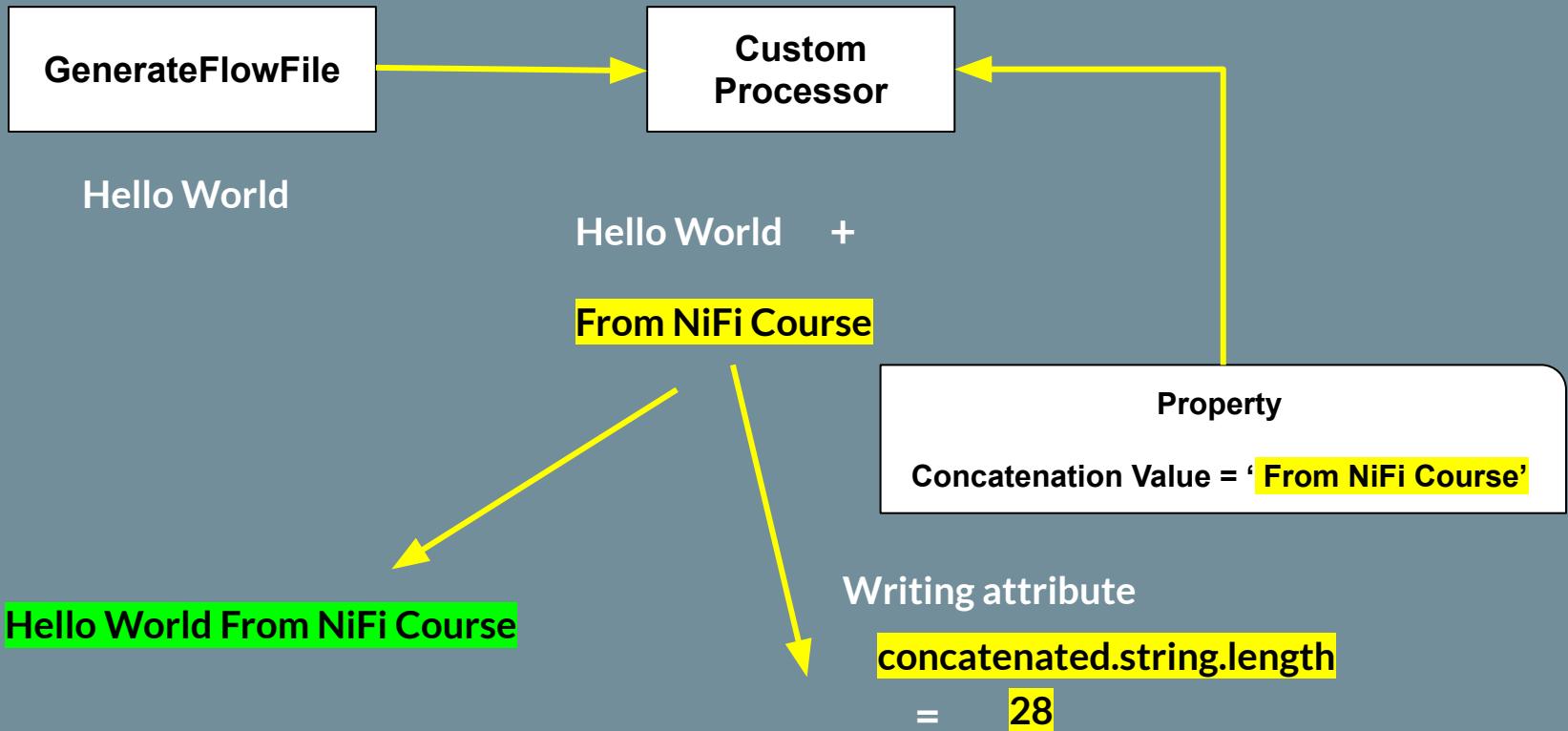
# Reading and Writing FlowFile Attributes

# Reading/Writing Attributes

```
2 usages
@Tags({"example"})
@CapabilityDescription("Provide a description")
@SeeAlso({})
@ReadsAttributes({
    @ReadsAttribute(attribute="", description="")
})
@WritesAttributes({
    @WritesAttribute(attribute="", description="")
})
public class ConcatText extends AbstractProcessor {
```

# Writing FlowFile Attributes

# Writing Attributes



# Writing Attributes

## Writes Attributes:

Name	Description	FlowFile
concatenated.string.length	Length of concatenated string written as attribute to the	

```
2 usages
@Tags({"example"})
@CapabilityDescription("Provide a description")
@SeeAlso({})
@ReadsAttributes({
    @ReadsAttribute(attribute="", description="")
})
@WritesAttributes({
    @WritesAttribute(attribute="concatenated.string.length", description="Length of concatenated string " +
        "written as attribute to the outgoing flowfile")
}
)

public class ConcatText extends AbstractProcessor {
```

# Writing Attributes

```
// performing concat on incoming content
final String resultantString = content.concat(property);

// writing the resultant string back to the flowfile
flowFile = session.write(flowFile, out -> out.write(resultantString.getBytes(Charset.defaultCharset())));

// add attribute to the flowfile
flowFile = session.putAttribute(flowFile, key: "concatenated.string.length", String.valueOf(resultantString.length()));

// transferring flowfile to next processor
session.transfer(flowFile, SUCCESS);
}
```

FlowFile

DETAILS ATTRIBUTES

Attribute Values

concatenated.string.length	28
filename	dd811581-e4dc-40c9-832c-1bc06d47d070
path	./
uuid	dd811581-e4dc-40c9-832c-1bc06d47d070

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    // getting flowfile from the input queue
    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

    // getting MY_PROPERTY property value
    final String property = context.getProperty(CONCAT_PROPERTY).getValue();

    final StringBuilder sb = new StringBuilder();

    // reading flowfile and storing as content
    session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

    final String content = sb.toString();

    // performing concat on incoming content
    final String resultantString = content.concat(property);

    // writing the resultant string back to the flowfile
    flowFile = session.write(flowFile, out -> out.write(resultantString.getBytes(Charset.defaultCharset())));

    // add attribute to the flowfile
    flowFile = session.putAttribute(flowFile, key: "concatenated.string.length", String.valueOf(resultantString.length()));

    // transferring flowfile to next processor
    session.transfer(flowFile, SUCCESS);
}
```

# Writing Processor Description

# Writing Processor Description

## Add Processor

Source Displaying 2 of 299 concat

Type	Version	Tags
ConcatText	1.0	example
MergeContent	1.15.3	zip, flowfile-stream-v3, flowfile-...

all groups ▾

amazon attributes  
avro aws consume  
csv database  
delete fetch get  
hadoop ingest  
insert json listen  
logs message  
pubsub put query  
record restricted  
source text  
update

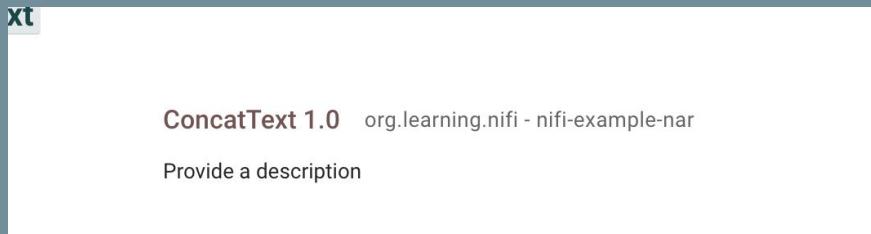
2 usages

```
@Tags({"example"})
@CapabilityDescription("Provide a description")
@SeeAlso({})
@ReadsAttributes({@ReadsAttribute(attribute="", description(""))})
@WritesAttributes({@WritesAttribute(attribute="", description(""))})
public class MyProcessor extends AbstractProcessor {
```

ConcatText 1.0 org.learning.nifi - nifi-example-nar

Provide a description

# Writing Processor Description



2 usages

```
@Tags({"example"})
@CapabilityDescription("Provide a description")
@SeeAlso({})
@ReadsAttributes({@ReadsAttribute(attribute="", description(""))})
@WritesAttributes({@WritesAttribute(attribute="", description(""))})
public class MyProcessor extends AbstractProcessor {
```

2 usages

# Processor Tags

# Processor Tags

## Add Processor

Source Displaying 2 of 299 concat

Type ▲	Version	Tags
ConcatText	1.0	example
MergeContent	1.15.3	zip, flowfile-stream-v3, flowfile-...

amazon attributes  
avro aws consume  
csv database  
delete fetch get  
hadoop ingest  
insert json listen  
logs message  
pubsub put query  
record restricted  
source text  
update

2 usages

```
@Tags({"example"})
@CapabilityDescription("Provide a description")
@SeeAlso({})
@ReadsAttributes({@ReadsAttribute(attribute="", description(""))})
@WritesAttributes({@WritesAttribute(attribute="", description(""))})
public class MyProcessor extends AbstractProcessor {
```

# Processor Tags

## Add Processor

Source

Displaying 1 of 299

ConcatText

all groups

Type ▲

Version

Tags

amazon attributes  
avro aws consume  
csv database  
delete fetch get  
hadoop ingest  
insert json listen  
logs message  
pubsub put query  
record restricted  
source text  
update

ConcatText 1.0 org.learning.nifi - nifi-example-nar

Concatenates incoming flowfile content with the provided `concatenation value` property

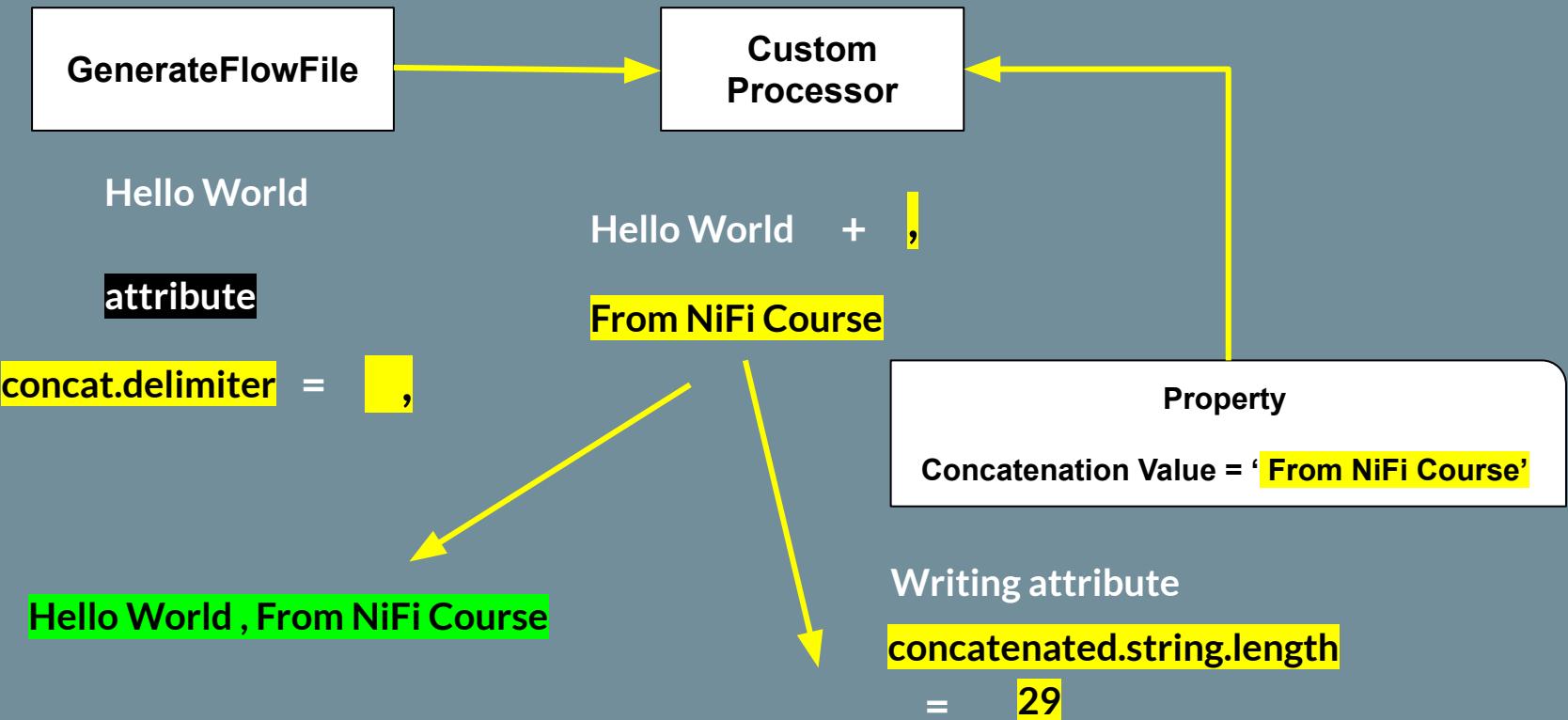
```
@Tags({"concat", "text", "append"})
```

```
@CapabilityDescription("Concatenates incoming flowfile content with the provided `concatenation value` property")
```

```
@SeeAlso({})
```

# Reading Attribute

# Reading Attributes



# Reading Attributes

## Reads Attributes:

Name	Description
concat.delimiter	Delimiter to be used while concatenating incoming flowfile with concatenation value property.

## Writes Attributes:

Name	Description
concatenated.string.length	Length of concatenated string written as attribute to the outgoing flowfile

# Reading Attributes

```
@Tags({"concat", "text", "append"})
@CapabilityDescription("Concatenates incoming flowfile content with the provided `concatenation value` property")
@SeeAlso({})
@ReadsAttributes({
    @ReadsAttribute(attribute="concat.delimiter", description="Delimiter to be used while concatenating incoming " +
        "flowfile with concatenation value property.")
})
@WritesAttributes({
    @WritesAttribute(attribute="concatenated.string.length", description="Length of concatenated string " +
        "written as attribute to the outgoing flowfile")
})
public class ConcatText extends AbstractProcessor {
```

## Reads Attributes:

Name	Description
concat.delimiter	Delimiter to be used while concatenating incoming flowfile with concatenation value property.

# Reading Attributes

```
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    // getting flowfile from the input queue
    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }

    // getting MY_PROPERTY property value
    final String property = context.getProperty(CONCAT_PROPERTY).getValue();

    // reading flowfile attribute
    final String delimiter = flowFile.getAttribute( key: "concat.delimiter" );
```

# Reading Attributes

Making conditional check.

If the attribute doesn't exist, it will return null, in that providing empty string ""

```
// getting MY_PROPERTY property value
final String property = context.getProperty(CONCAT_PROPERTY).getValue();

// reading flowfile attribute
String delimiter = flowFile.getAttribute(key: "concat.delimiter");
delimiter = delimiter == null ? "" : delimiter;
```

# Reading Attributes

```
// getting MY_PROPERTY property value
final String property = context.getProperty(CONCAT_PROPERTY).getValue();

// reading flowfile attribute
String delimiter = flowFile.getAttribute( key: "concat.delimiter");
delimiter = delimiter == null ? "" : delimiter;

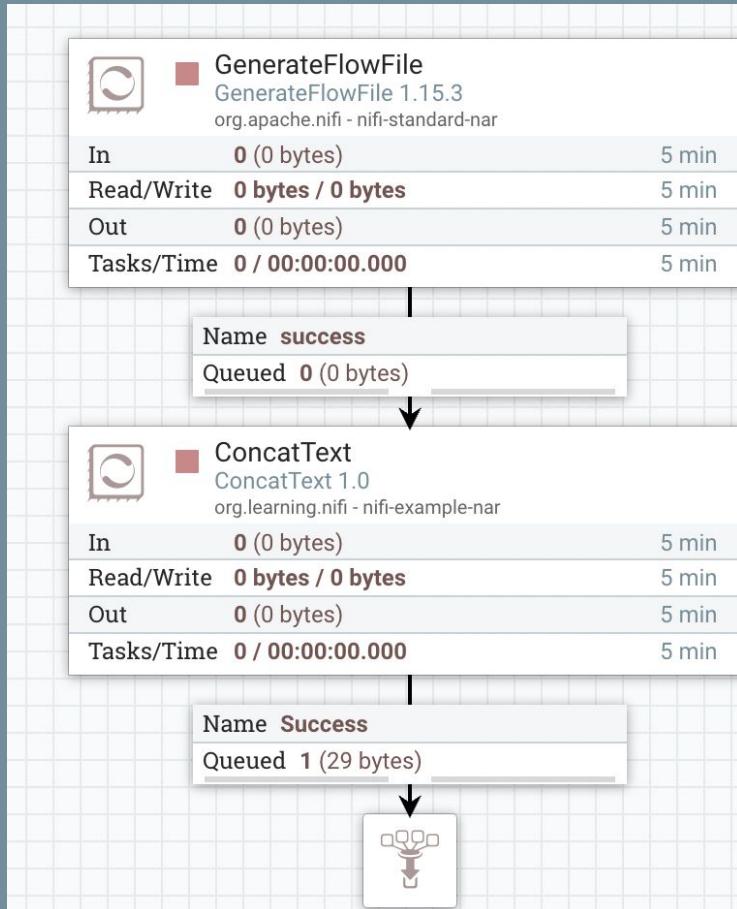
final StringBuilder sb = new StringBuilder();

// reading flowfile and storing as content
session.read(flowFile, in -> sb.append(IOUtils.toString(in, Charset.defaultCharset())));

final String content = sb.toString();

// performing concat on incoming content
final String resultantString = content.concat(delimiter).concat(property);
```

# Reading Attributes



Hello World

attribute

concat.delimiter = ,

Hello World , From Nifi Course

Write attribute

concatenated.string.length =29

# ConcatText Processor Result

## FlowFile

DETAILS

ATTRIBUTES

### Attribute Values

concat.delimiter

,

concatenated.string.length

29

filename

9f72f20c-108d-4b0e-8e93-8d04337a3e5f

path

./

uuid

9f72f20c-108d-4b0e-8e93-8d04337a3e5f