Querying Record

Query Record Fields

Source JSON Record

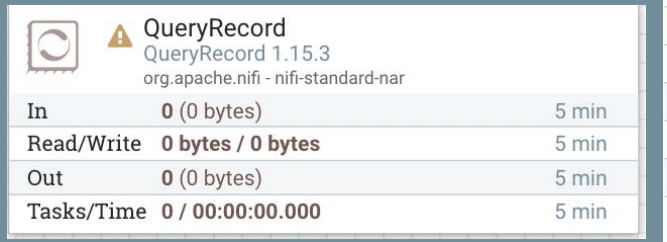

```
{
  "id" : 6523,
  "ident" : "00A",
  "type" : "heliport",
  "name" : "Total Rf Heliport",
  "latitude_deg" : 40.0708,
  "longitude_deg" : -74.93360137939453,
  "elevation_ft" : 11,
  "continent" : "NA",
  "iso_country" : "US",
  "iso_region" : "US-PA",
  "municipality" : "Bensalem",
  "scheduled_service"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

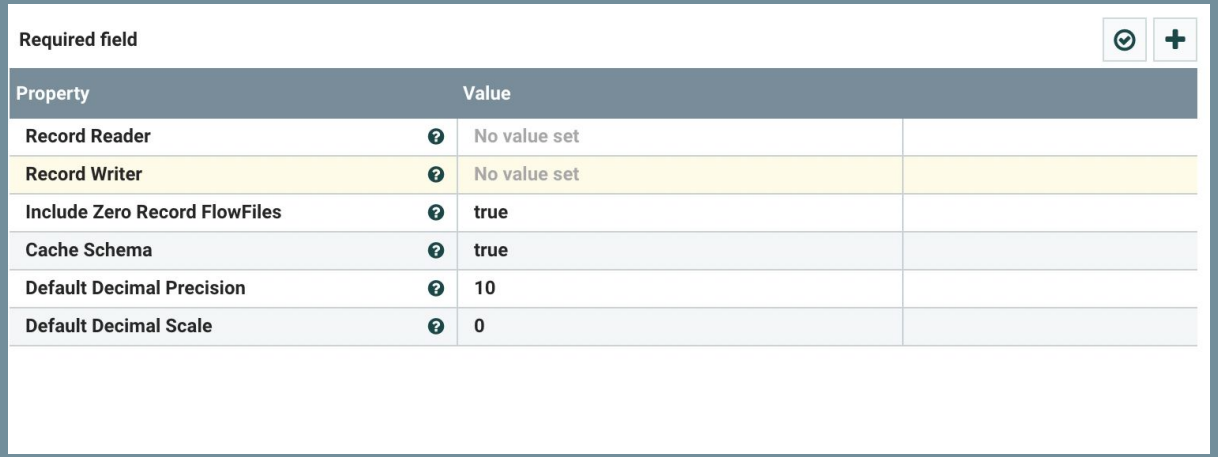

SQL Like Select Statement

Select * from DATA
WHERE
scheduled_service = 'no'

Querying Record Field

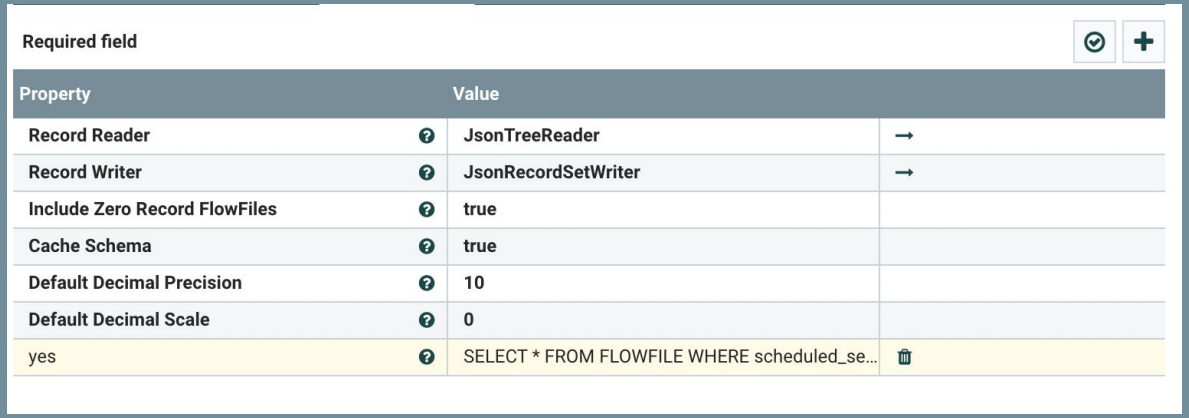
QueryRecord Processor to query like sql

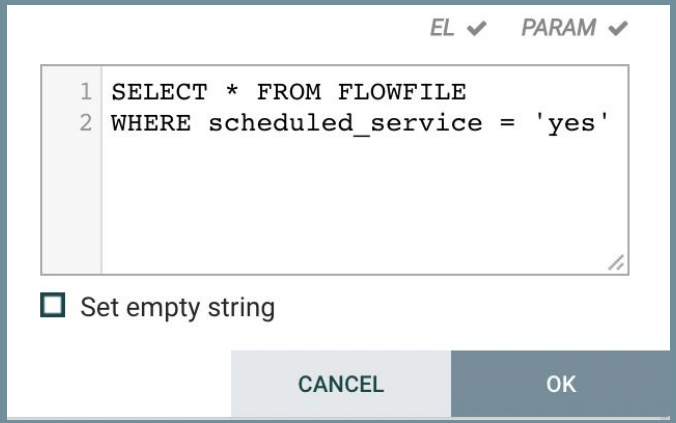




Querying Record Field

QueryRecord Processor to query like sql





Query Record Fields

Select * from FLOWFILE
WHERE
scheduled_service = ‘yes’

FlowFile

Putting to database

Inserting JSON into DB

JSON

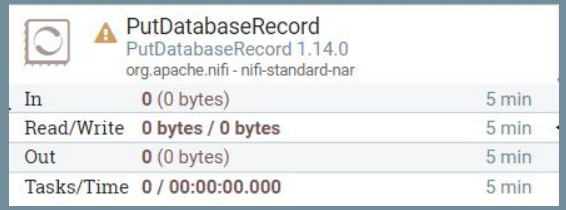
```
{
    "id": 40814,
    "ident": "SK-377",
    "type": "small_airport",
    "name": "Barbosa Airport",
    "latitude_deg": 5.943333,
    "longitude_deg": -73.611389,
    "elevation_ft": 5176,
    "continent": "SA",
    "iso_country": "CO",
    "iso_region": "Santander Department",
    "municipality": "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"
}
```

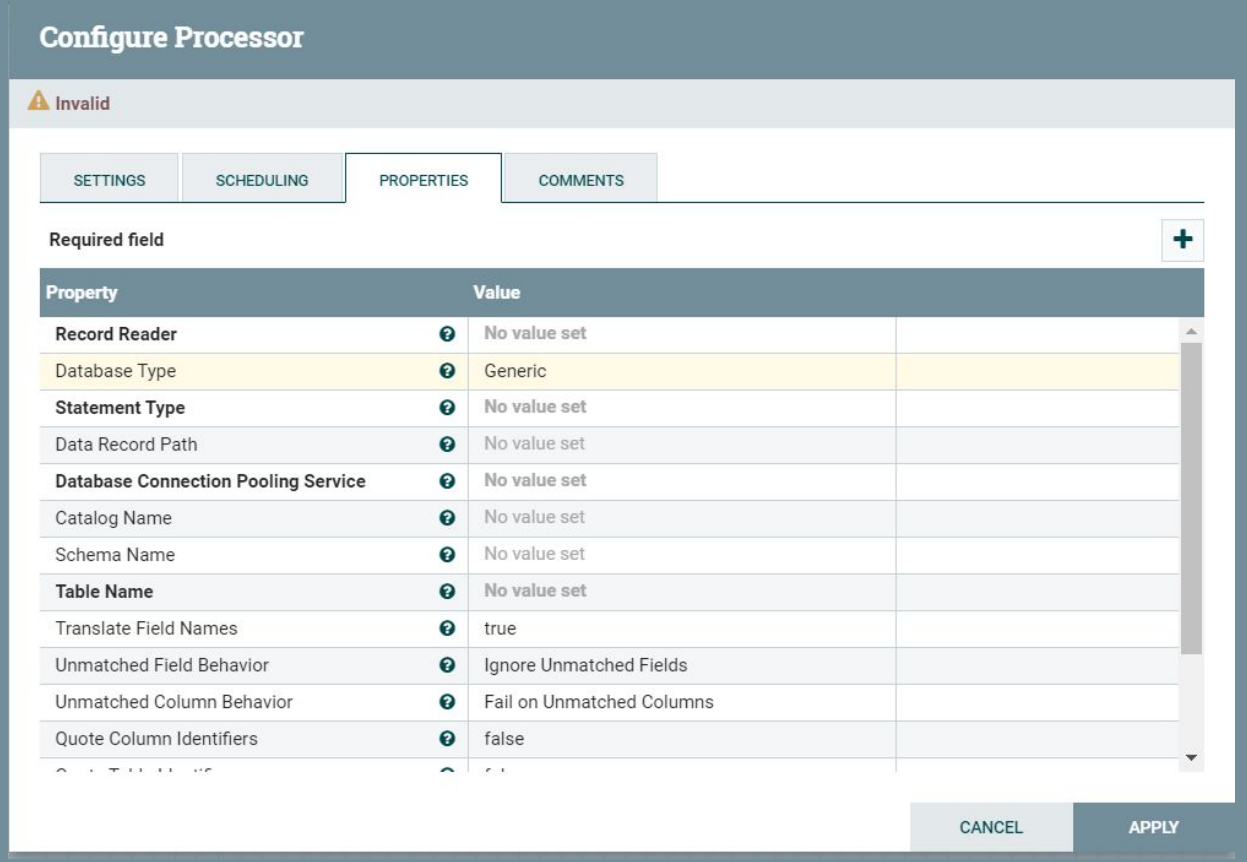
Airports Table

```
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_REGISON 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

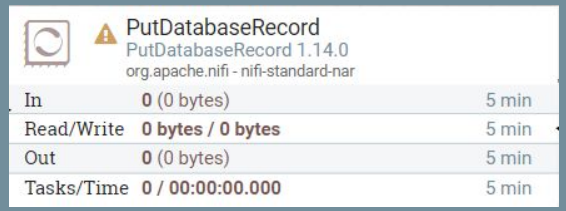
CRUD operations with Database

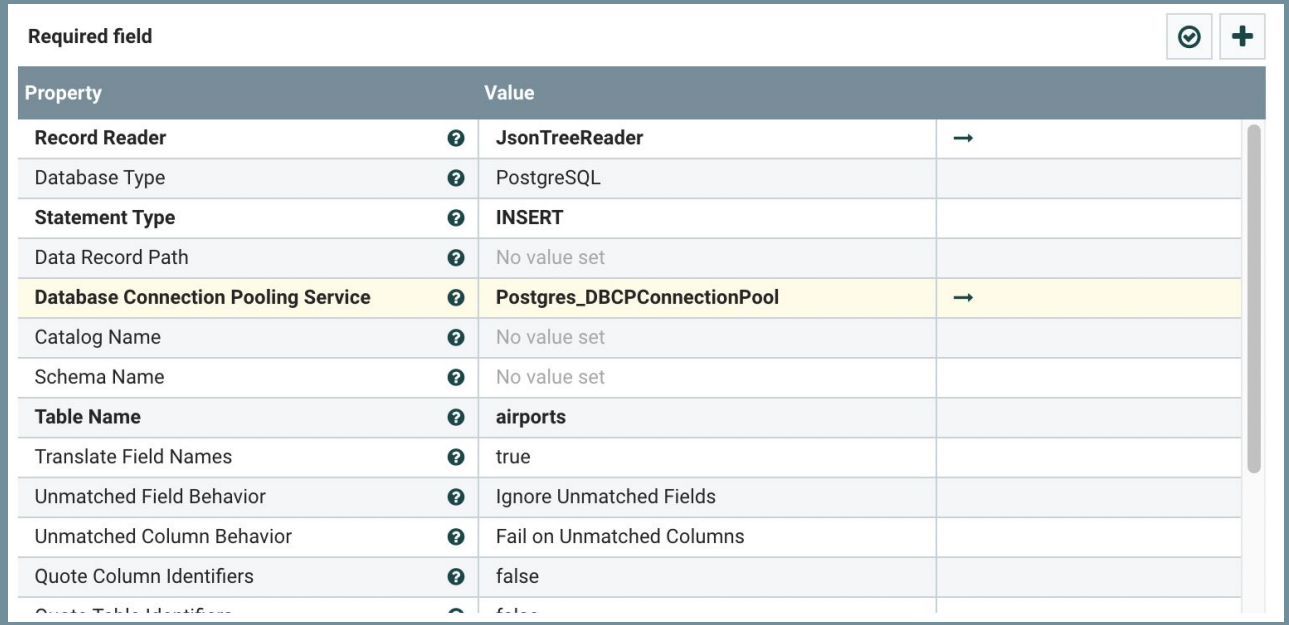




– PutDatabaseRecord Processor

Insert data to DB





– PutDatabaseRecord Processor

Statement Type	**INSERT**	
Data Record Path	No value set	
Database Connection Pooling Service	**Postgres_DBCPConnectionPool**	→
Catalog Name	No value set	

Required field

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

– PutDatabaseRecord Processor

Results

```
postgres=# select * from airports;
   id   | ident  |      type      |         name          | latitude_deg | longitude_deg | elevation_ft | continent | iso_country | iso_regison |   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       |            |           |
   6099 | 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

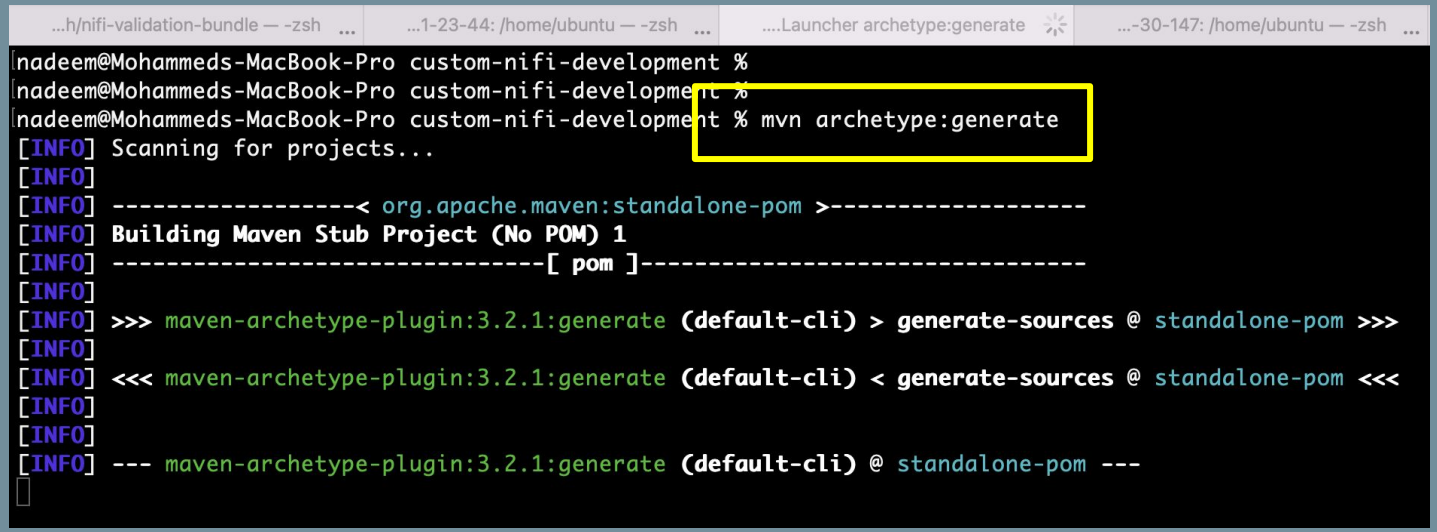
Creating Custom Processor

Steps to create custom processor

Step 1: Maven archetype

Maven archetype provides easiest way to generate NiFi processor

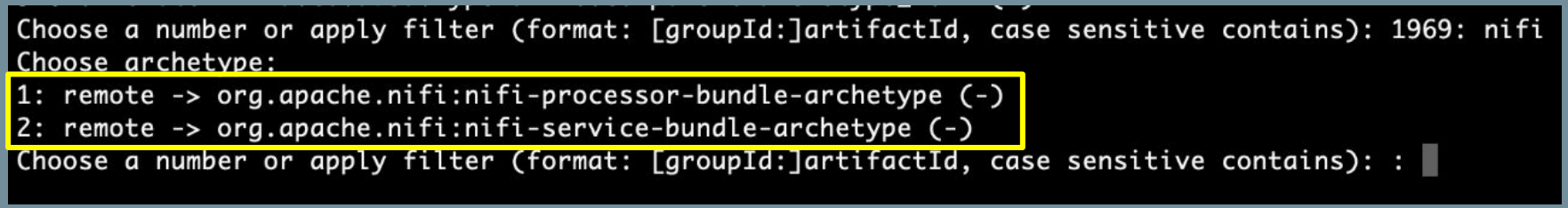
$ > mvn archetype:generate



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`

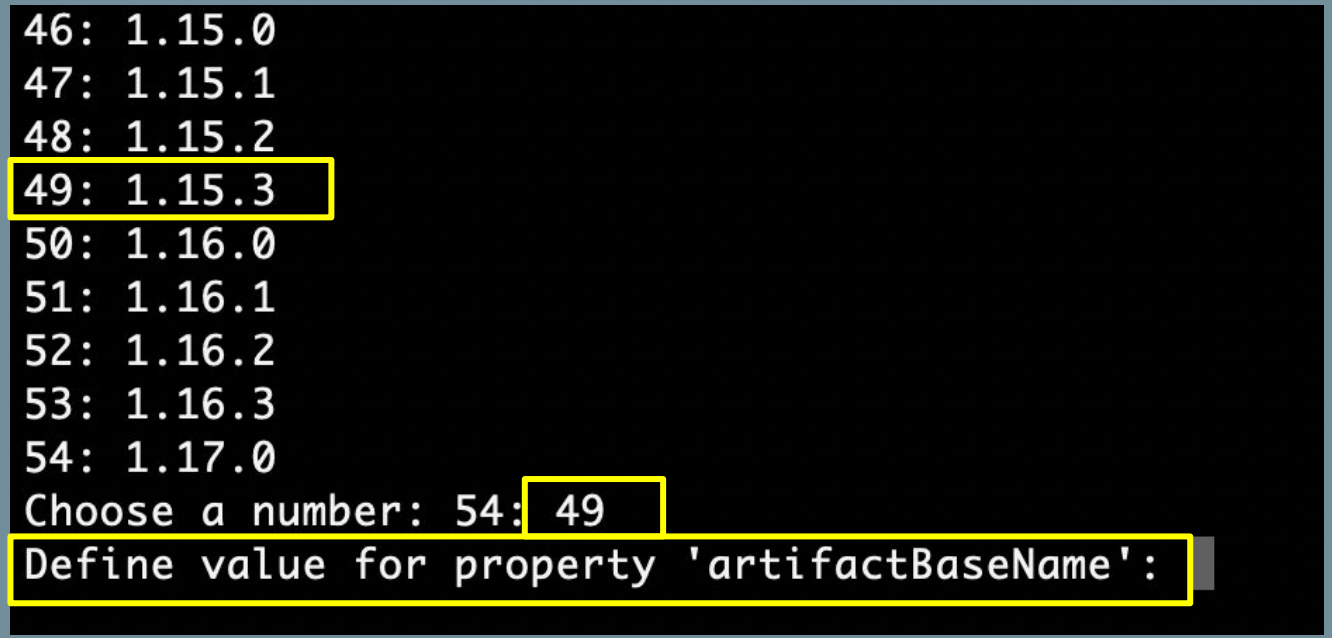


Step 1: mvn archetype:generate

```
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



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] ------------------------------------------------------------------------
nadeem@Mohammeds-MacBook-Pro custom-nifi-development %
```

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:
```

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

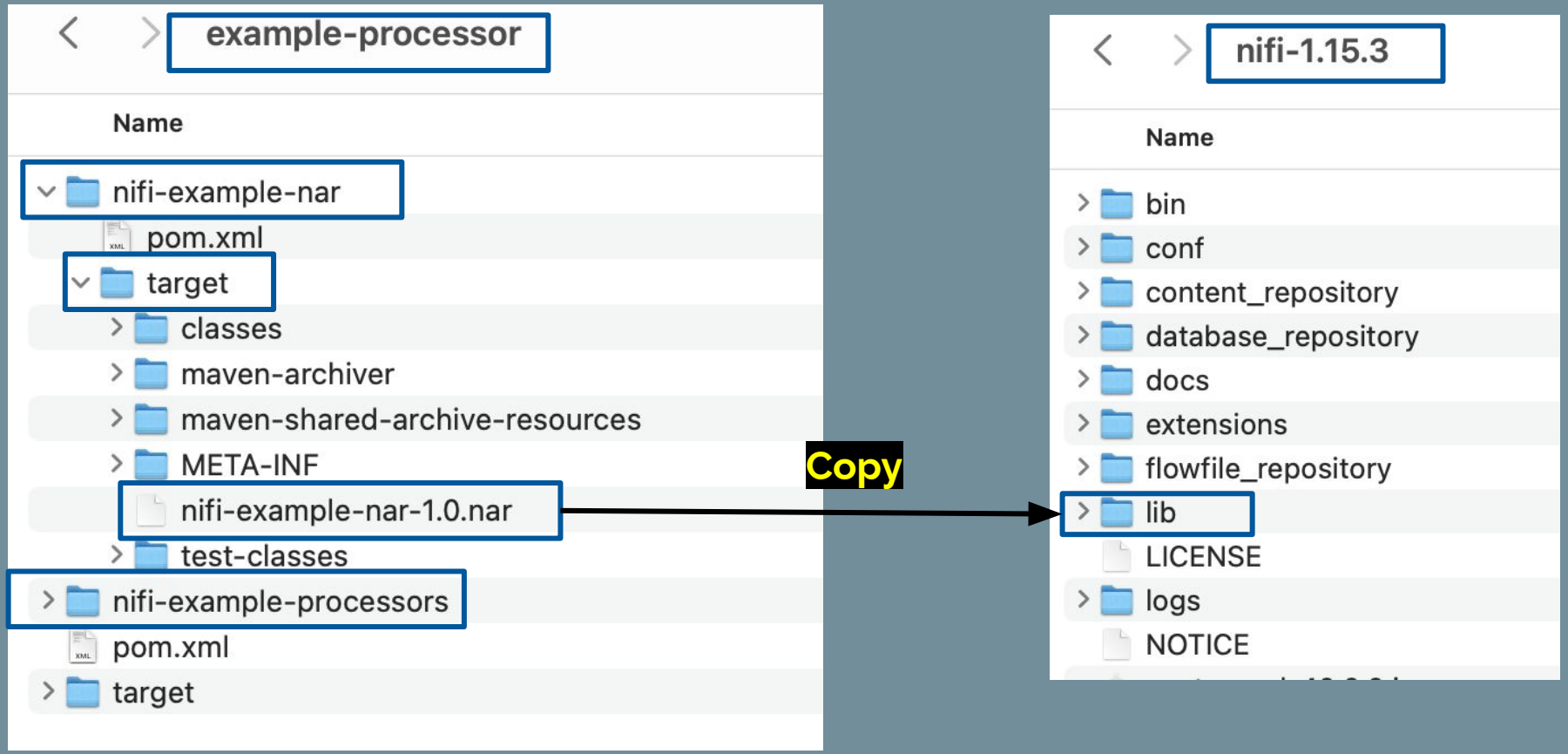
$ > `mvn clean install`

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

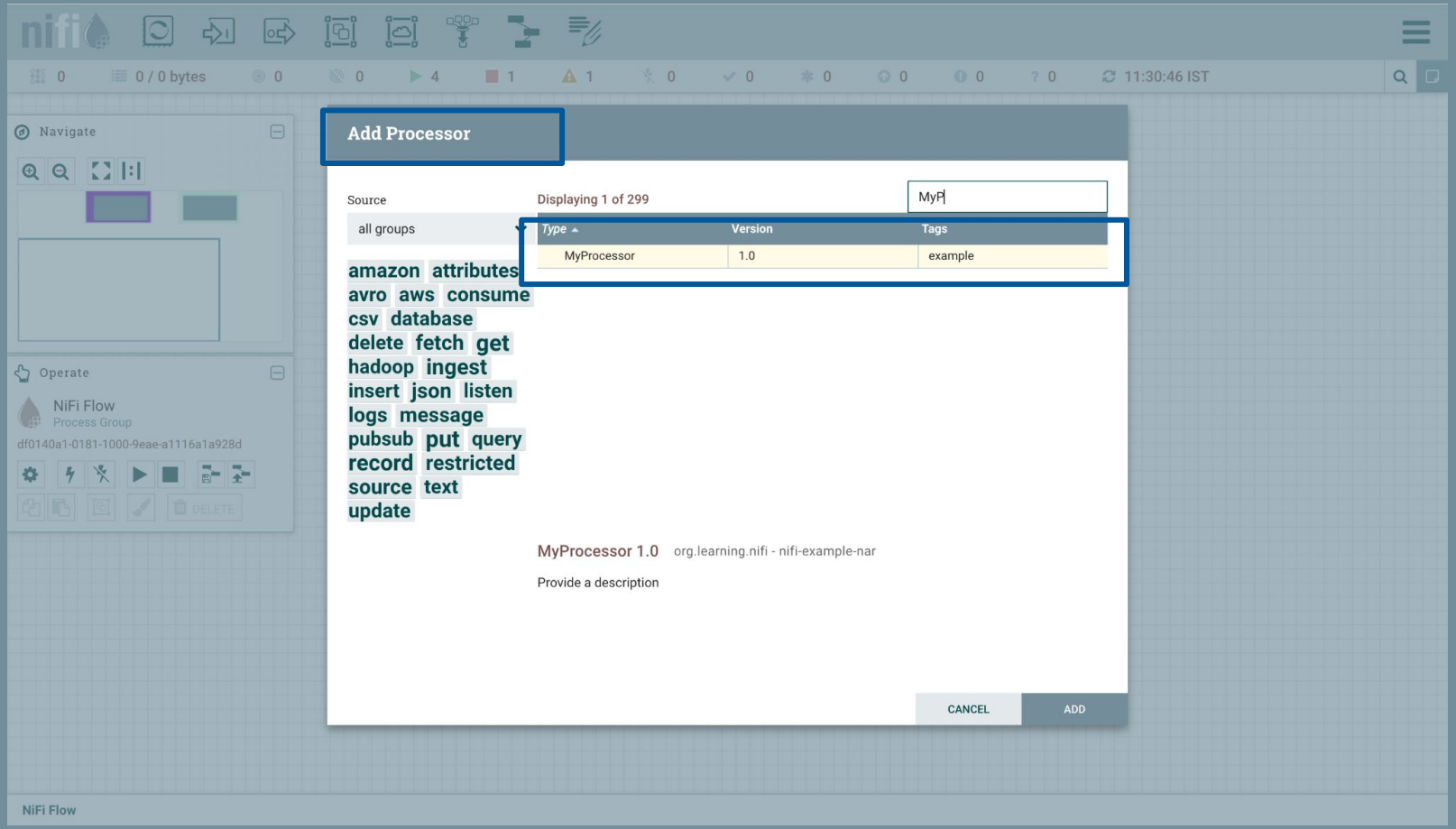
Before we get into further details, let's deploy and visualize the processor

$ > mvn clean install

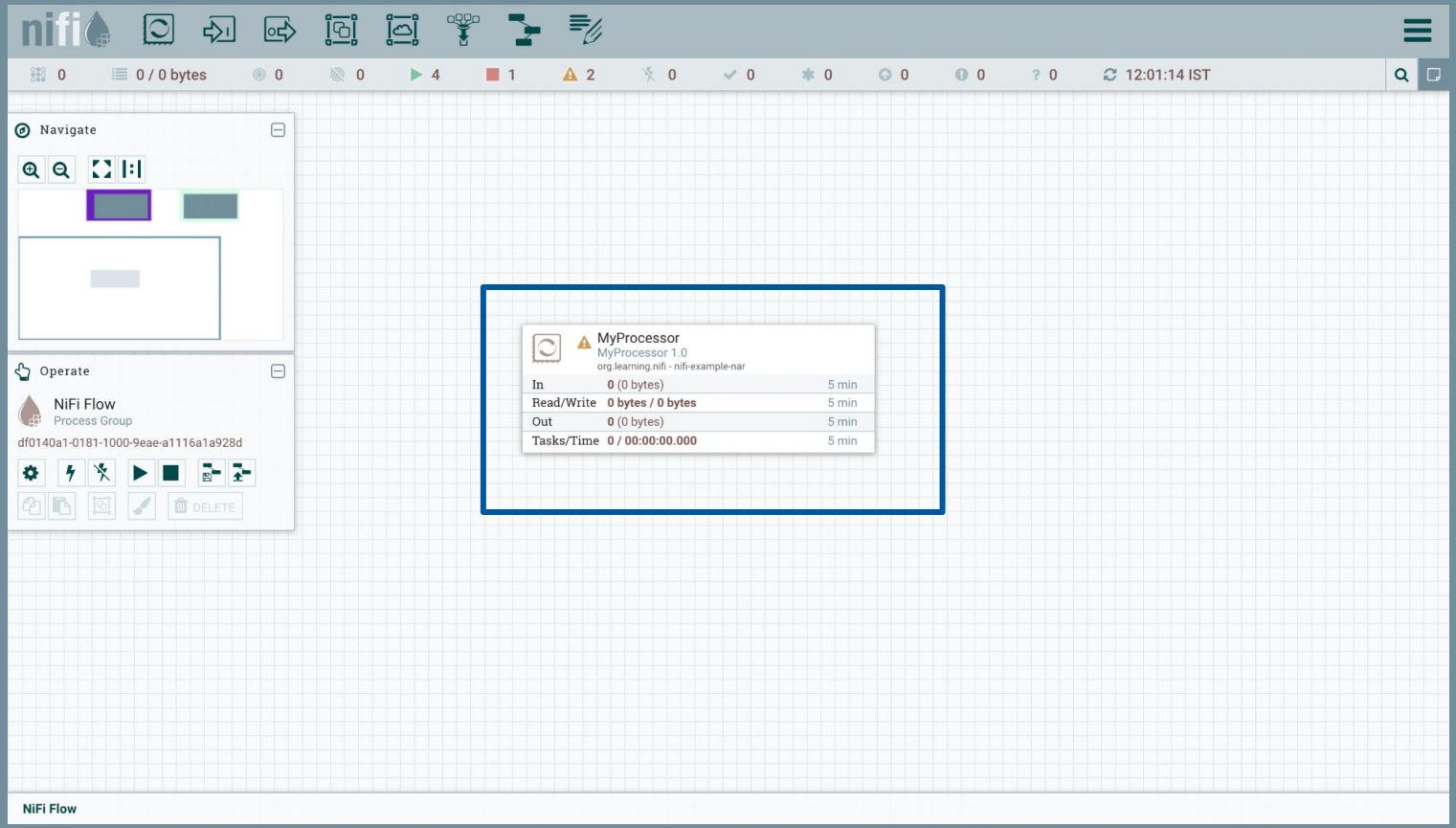
Copy processor archive to NiFi

Restart NiFi

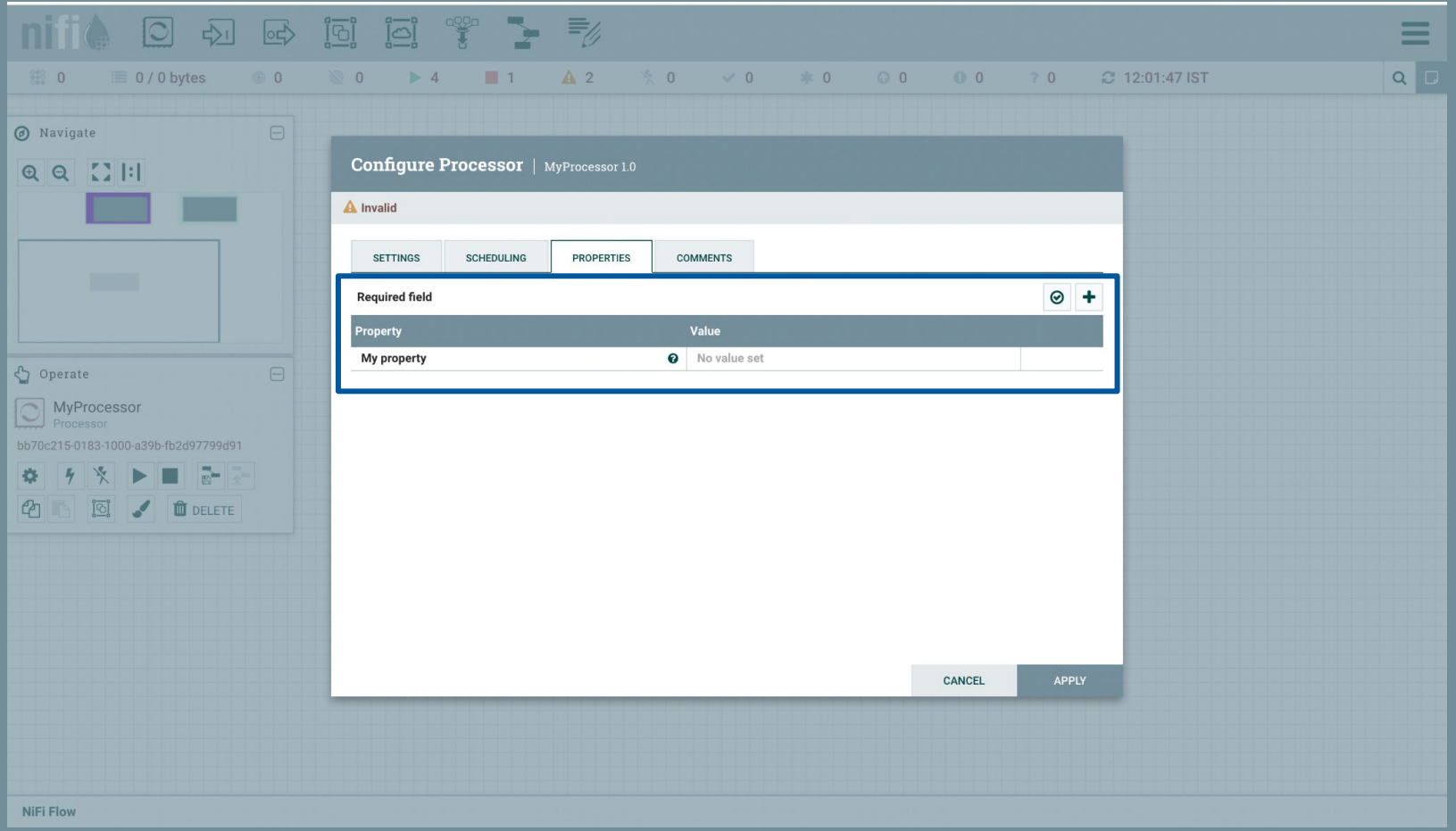
Custom Processor



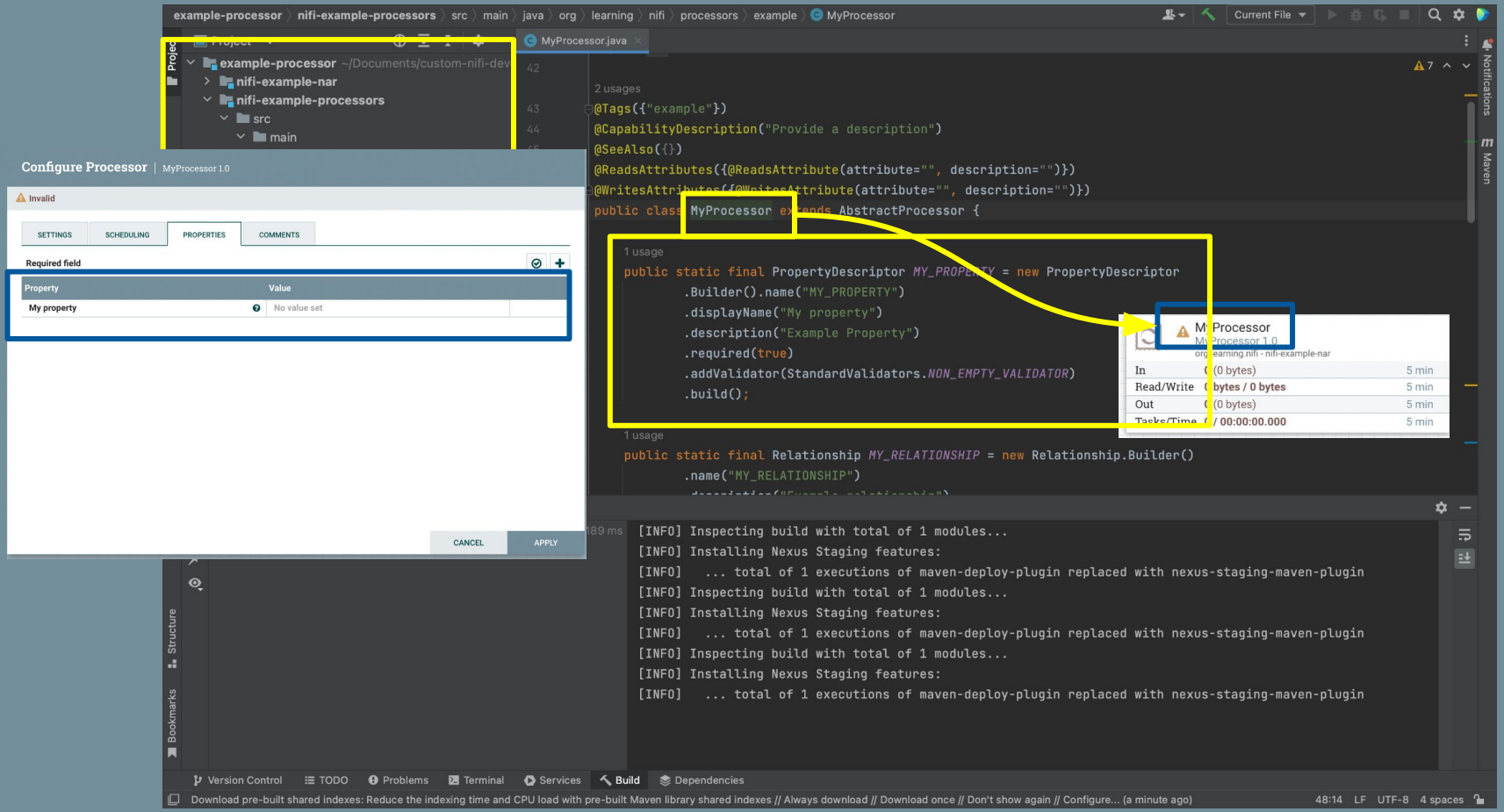
Custom Processor



Custom Processor



Project Setup in IntelliJ

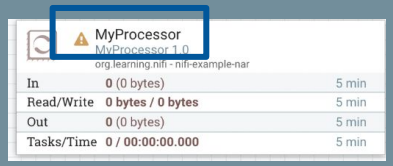


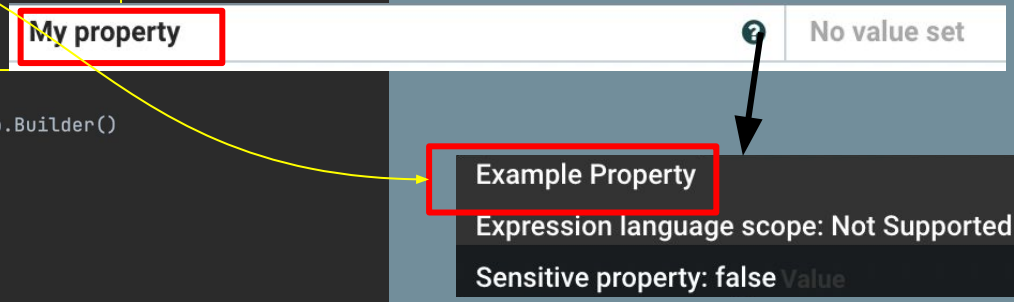
```java
public class MyProcessor extends AbstractProcessor {

    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

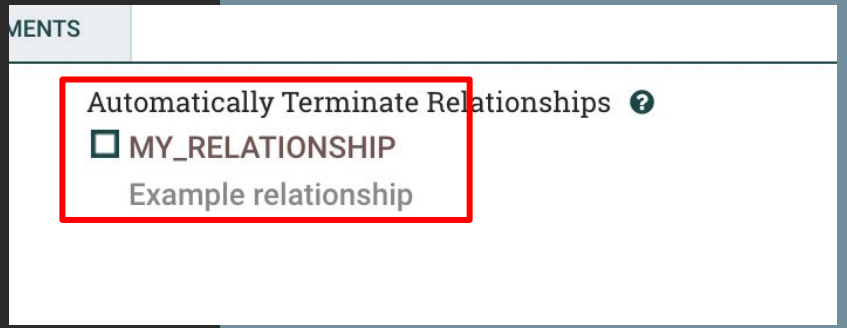
All relationships

```java
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; }
```





```java
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; }
```



```java
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

```java
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

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

```java
@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

```java
    @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

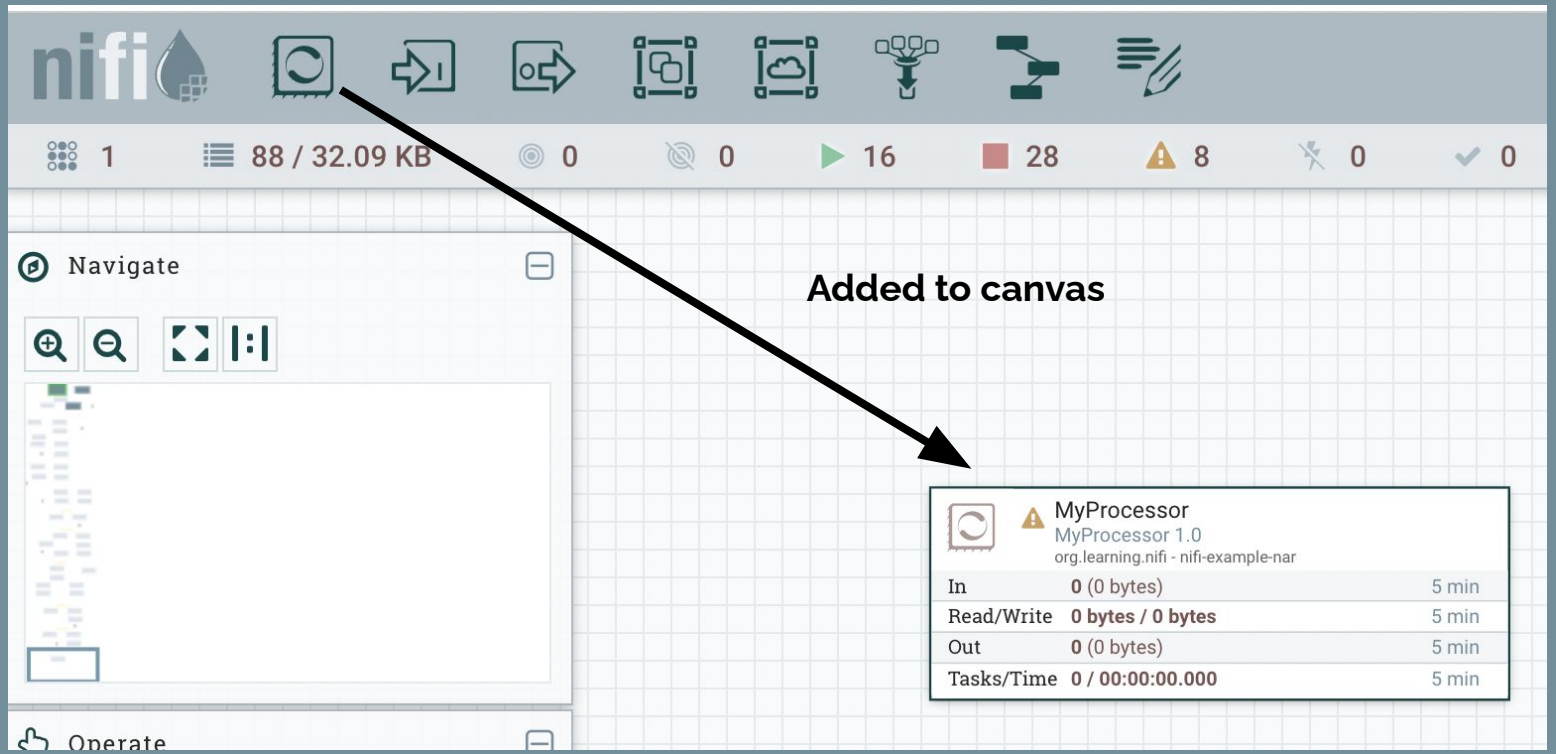
@Stopped

NiFi Framework calls your methods based on annotation defined

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

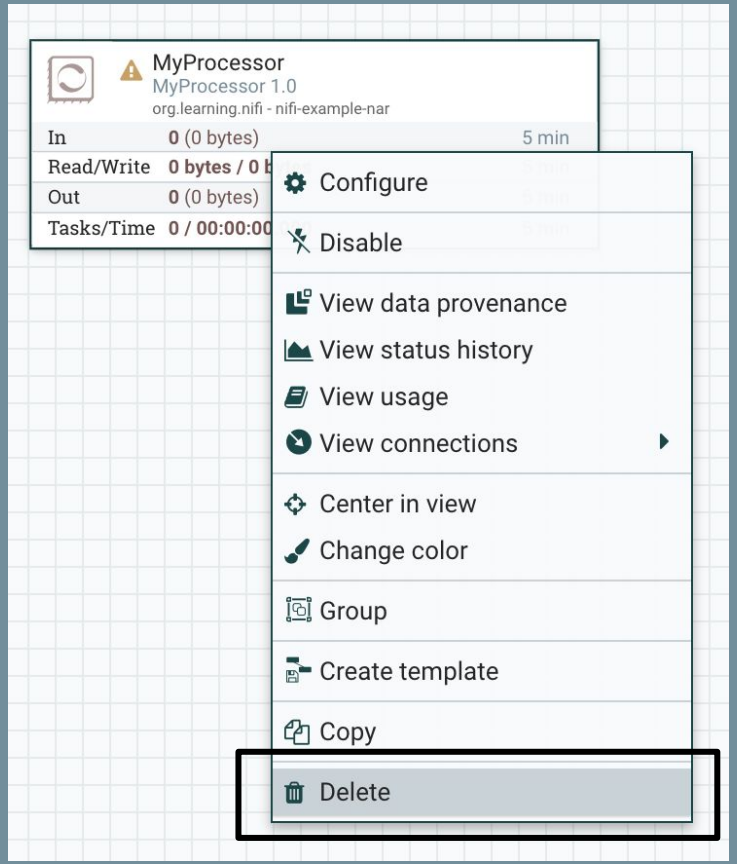
Processor Lifecycle

@OnAdded



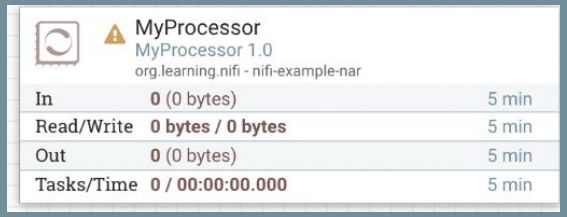
Processor Lifecycle

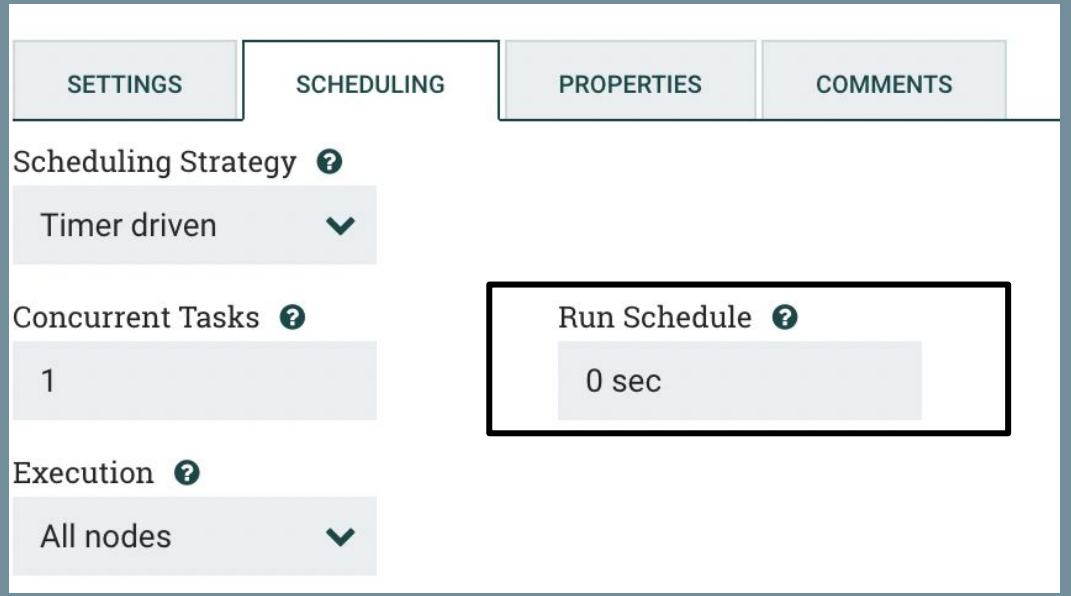
@OnRemoved



Processor Lifecycle

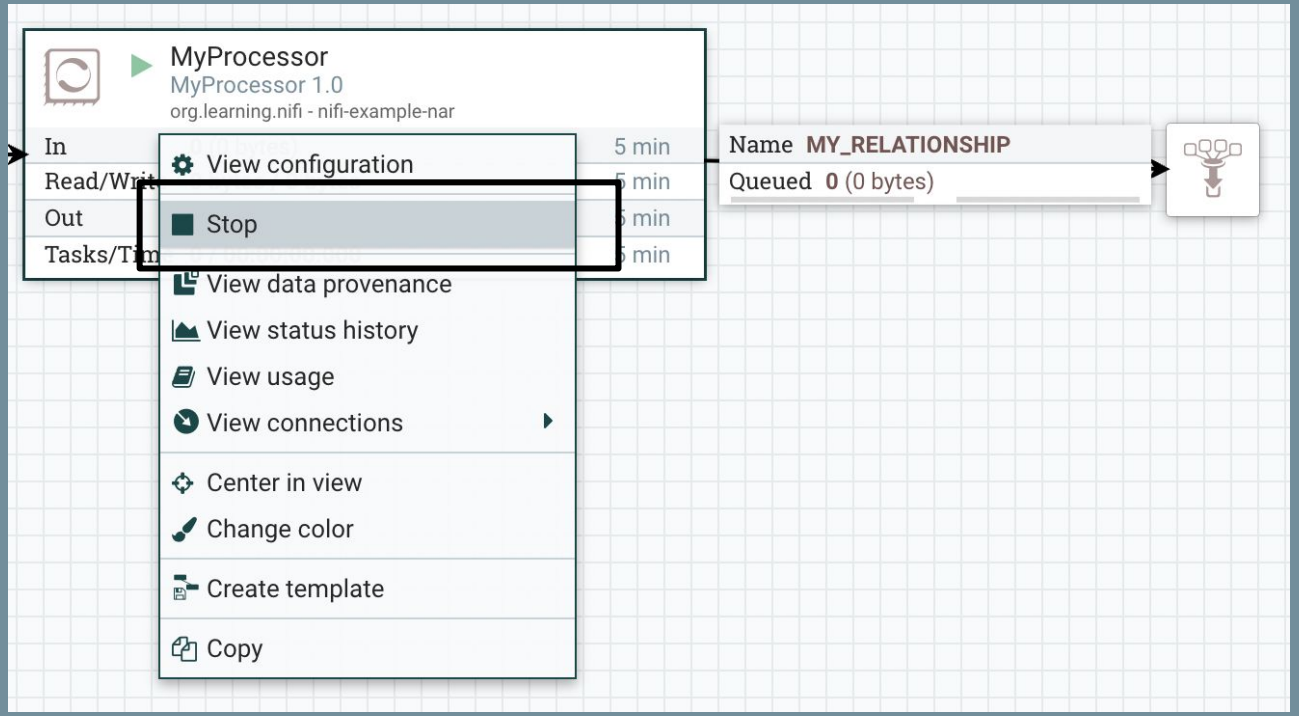
@OnScheduled





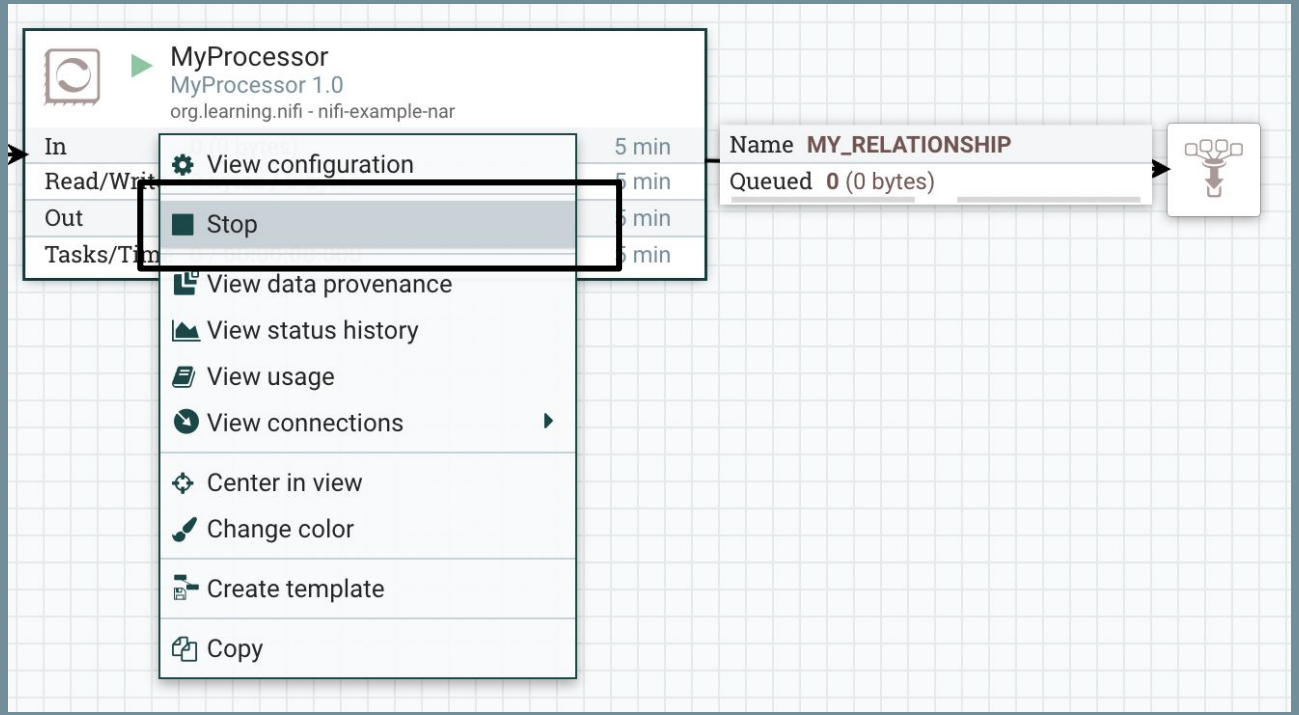
Processor Lifecycle

@OnStopped



Processor Lifecycle

@OnUnscheduled



Processor Lifecycle

```java
@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

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

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }
    // TODO implement
}
```

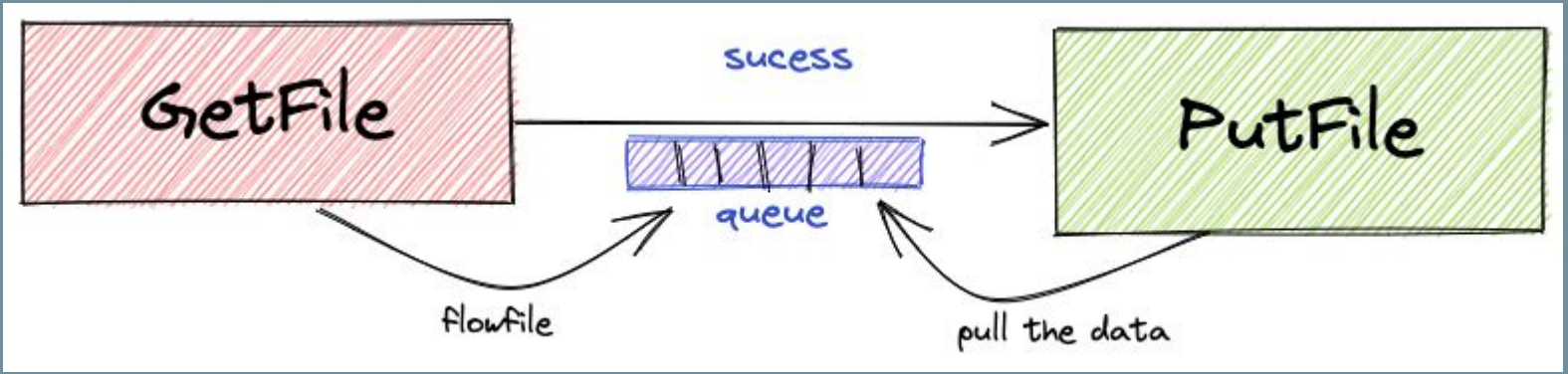
OnTrigger method is called only when the processor is scheduled (i.e started) and there is exist work for a processor to do.

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

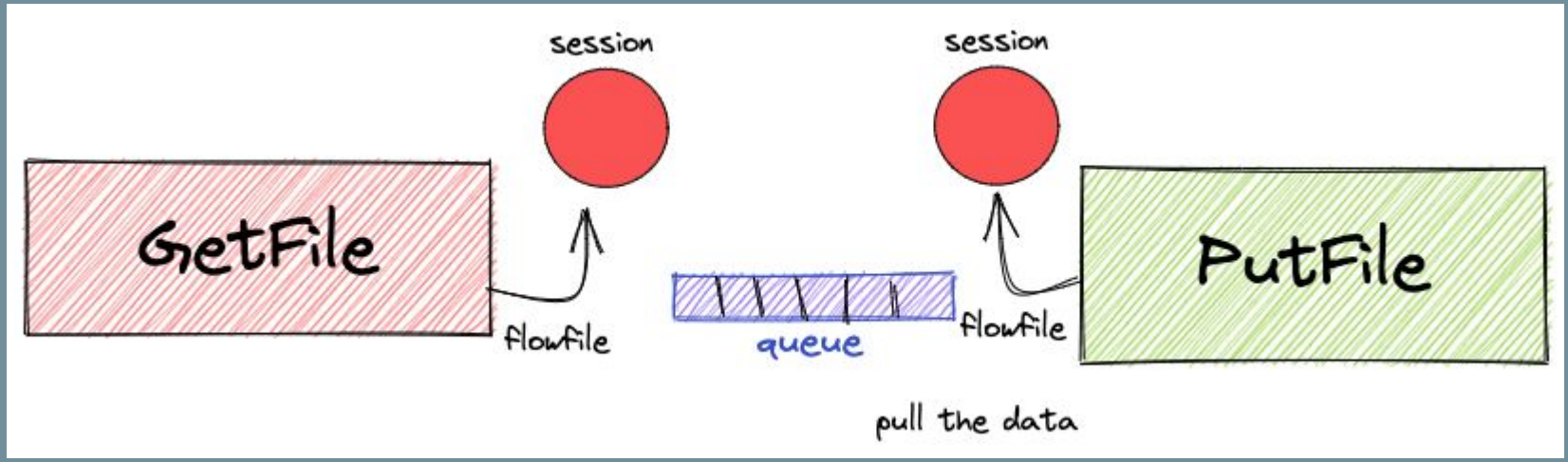
Or when a there is no incoming connection, then also it is called only when @TriggerEmpty annotation is present

Understanding Session

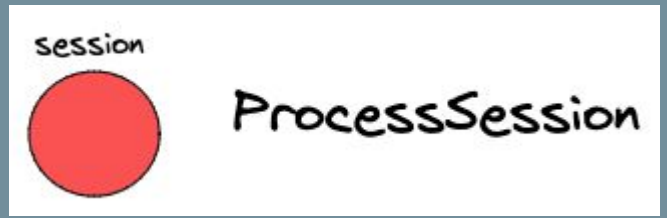
OnTrigger method in Processor



OnTrigger method in Processor

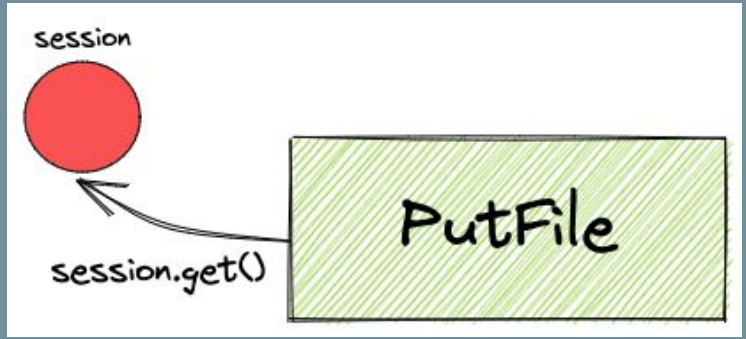



```java
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }
    // TODO implement
}
```



OnTrigger method in Processor

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

    FlowFile flowFile = session.get();
    if ( flowFile == null ) {
        return;
    }
    // TODO implement
}
```



OnTrigger method in Processor

```java
@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

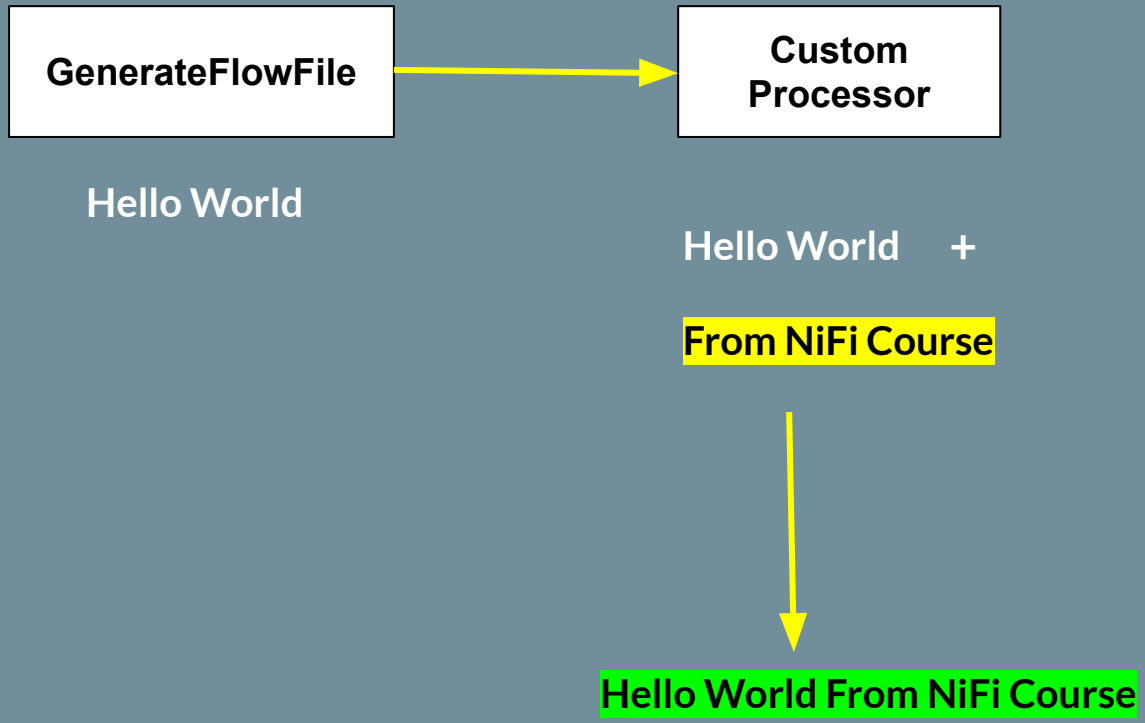
```java
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

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

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

Reading incoming flowfile content

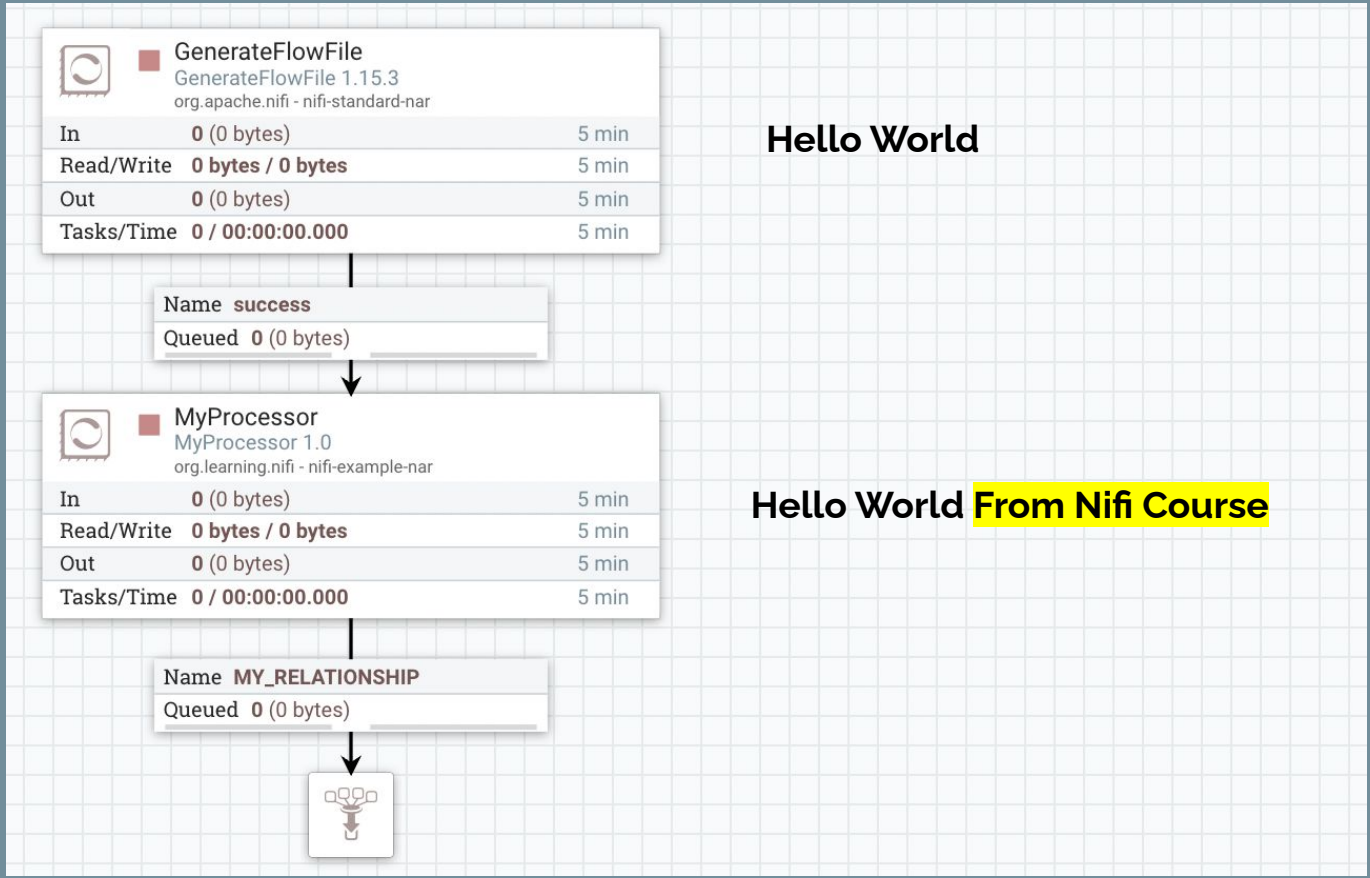
Reading flowfile content

```java
@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

Concating incoming flowfile

Concating incoming FlowFile

```java
@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

```java
@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 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);
}
```

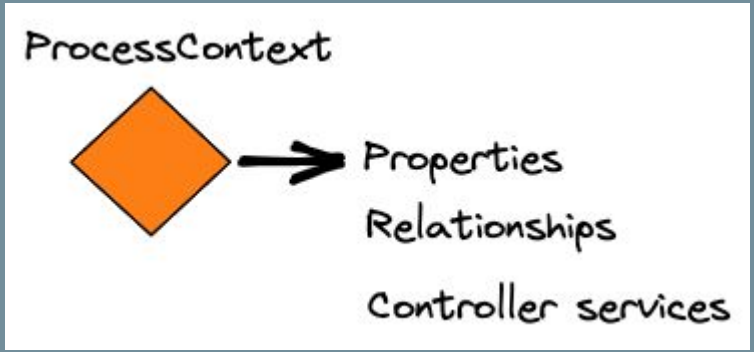
Reading from Properties

Reading from Processor Properties

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

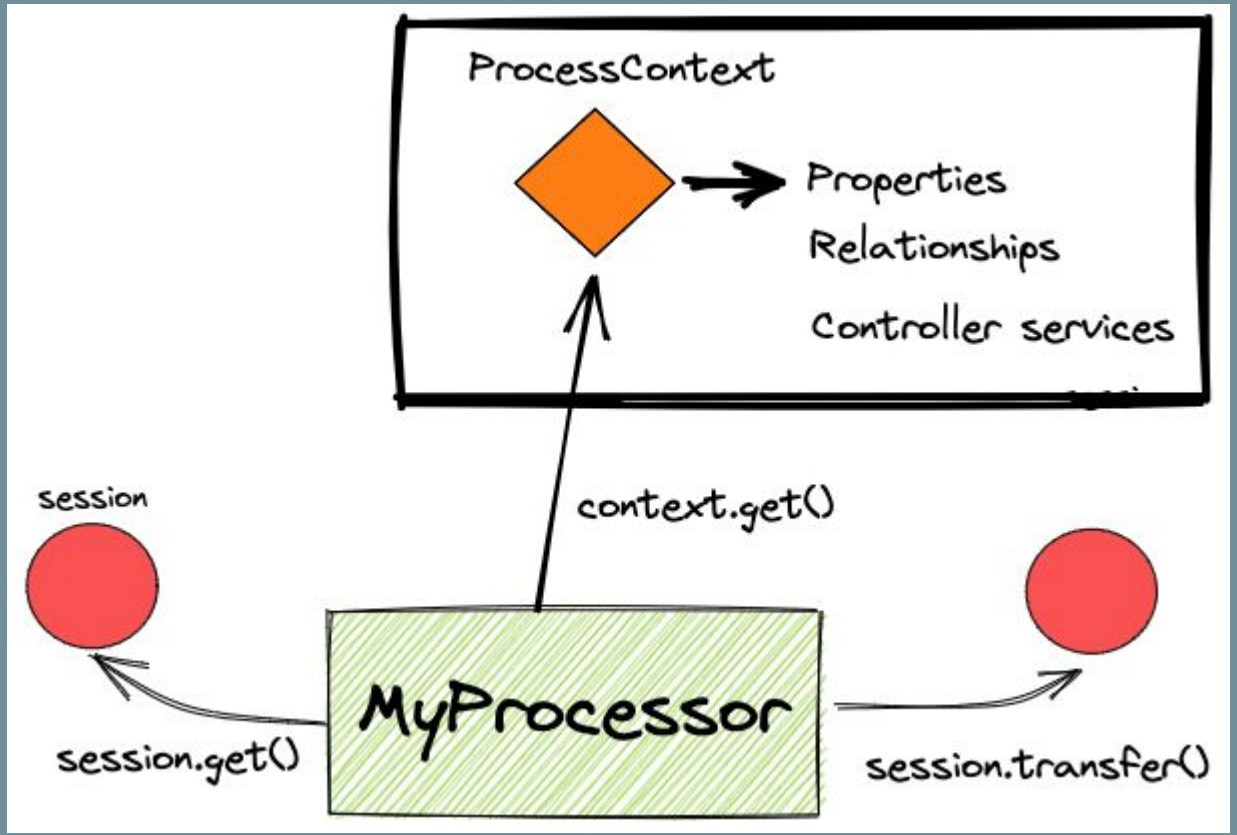
Understanding ProcessContext

ProcessContext




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

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

ProcessContext



ProcessContext

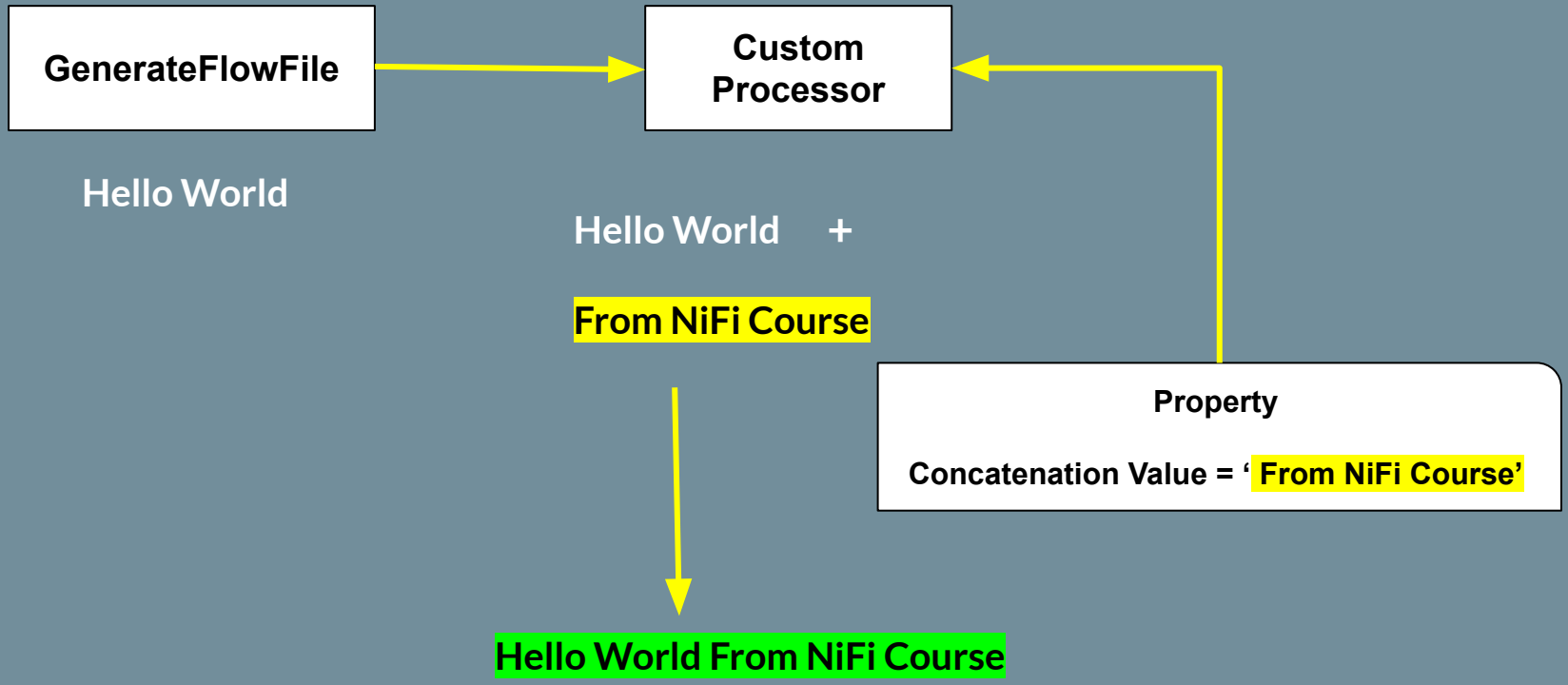
```java
@Override
public void onTrigger(final ProcessContext context, final ProcessSession session) {

    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

```java
@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 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);
}
```

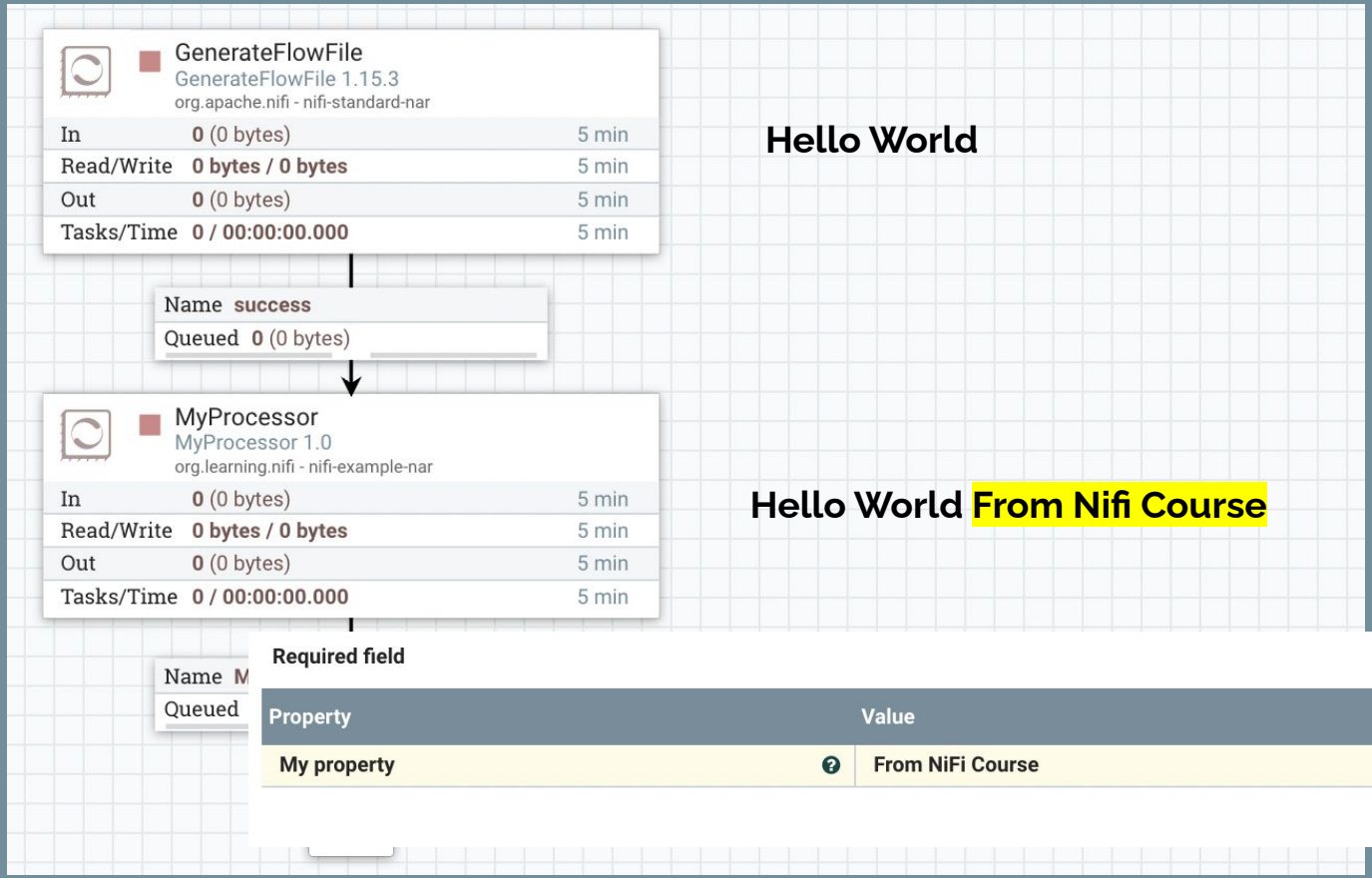
Reading from property



ProcessContext

```java
@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

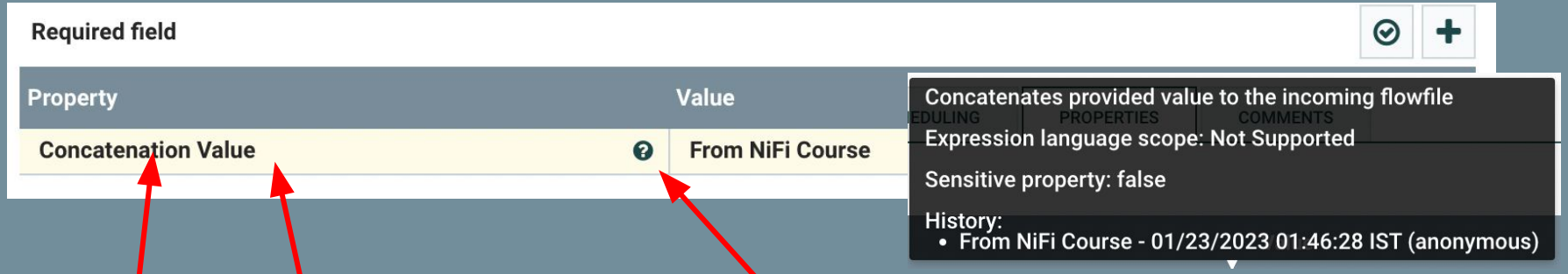


Refactoring My_Property

Renaming My Property


```
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();
```

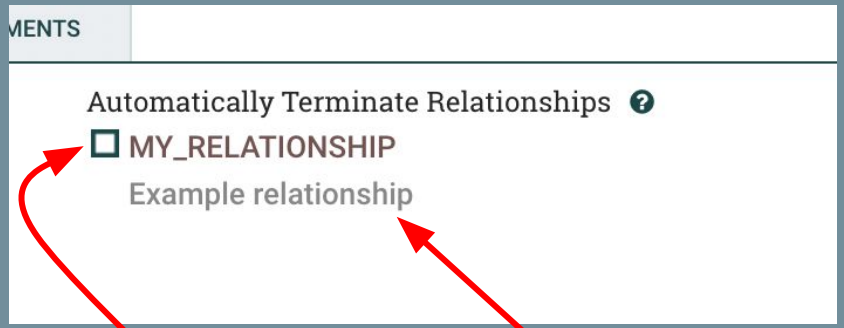
Renaming My Property

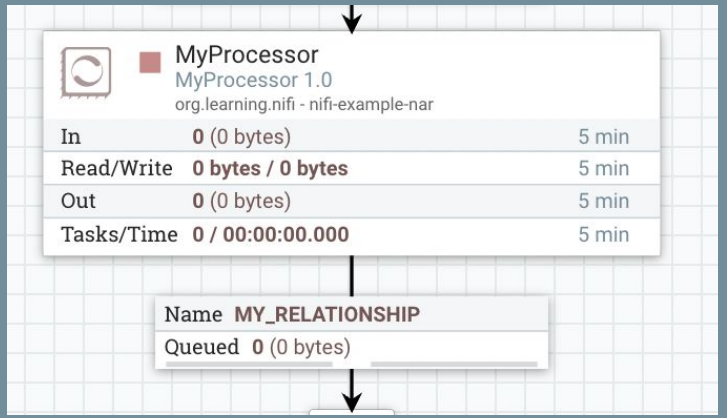



```
2 usages
public static final PropertyDescriptor CONCAT_PROPERTY = new PropertyDescriptor
        .Builder().name("CONCAT_PROPERTY")
        .displayName("Concatenation Value")
        .description("Concatenates provided value to the incoming flowfile")
        .required(true)
        .addValidator(StandardValidators.NON_EMPTY_VALIDATOR)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

Refactoring My_Relationship

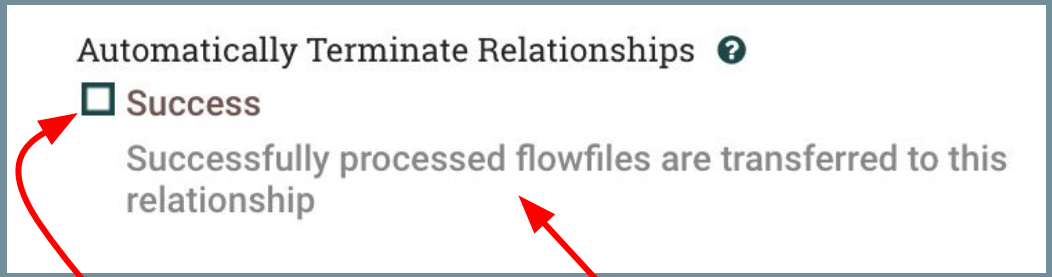
Renaming My Relationship

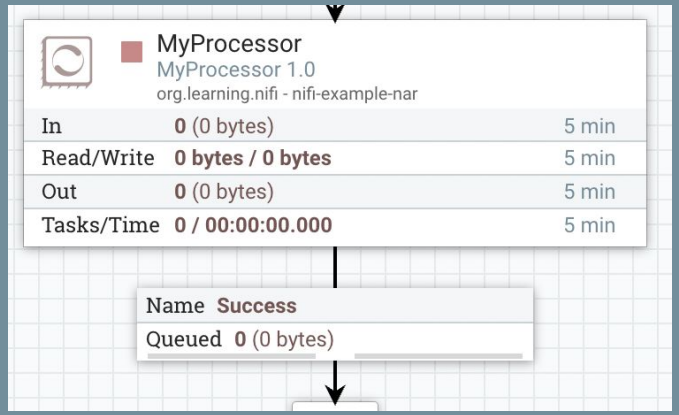





```
2 usages
public static final Relationship MY_RELATIONSHIP = new Relationship.Builder()
        .name("MY_RELATIONSHIP")
        .description("Example relationship")
        .build();
```

Renaming My Relationship





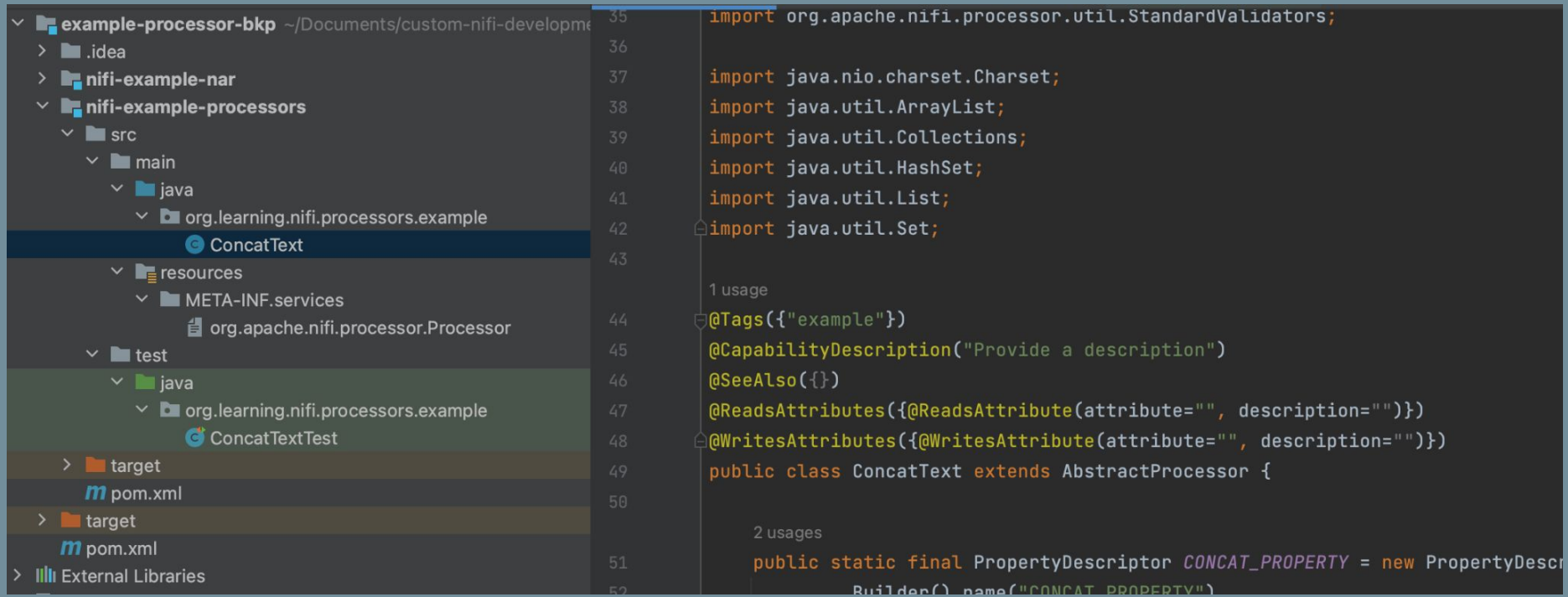

```
2 usages
public static final Relationship SUCCESS = new Relationship.Builder()
        .name("Success")
        .description("Successfully processed flowfiles are transferred to this relationship")
        .build();
```

Renaming MY_Processor

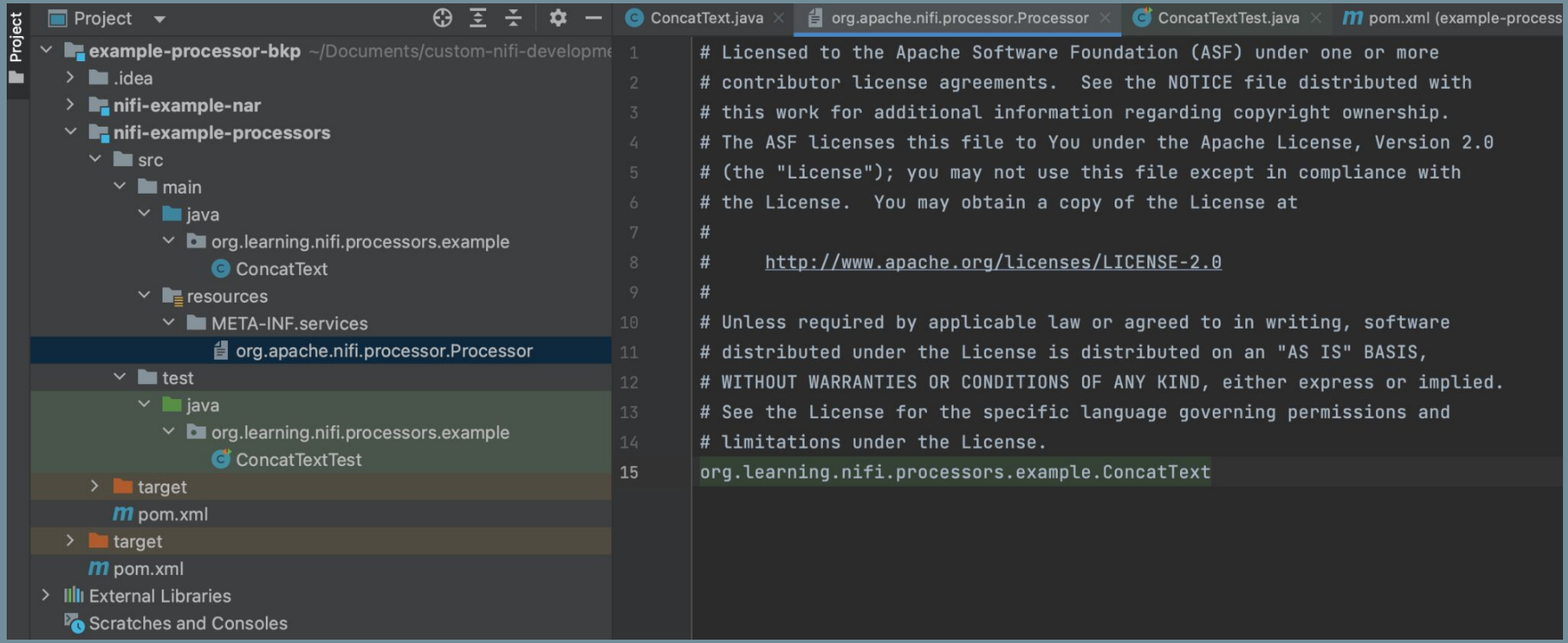
Renaming My Relationship


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

    2 usages
```

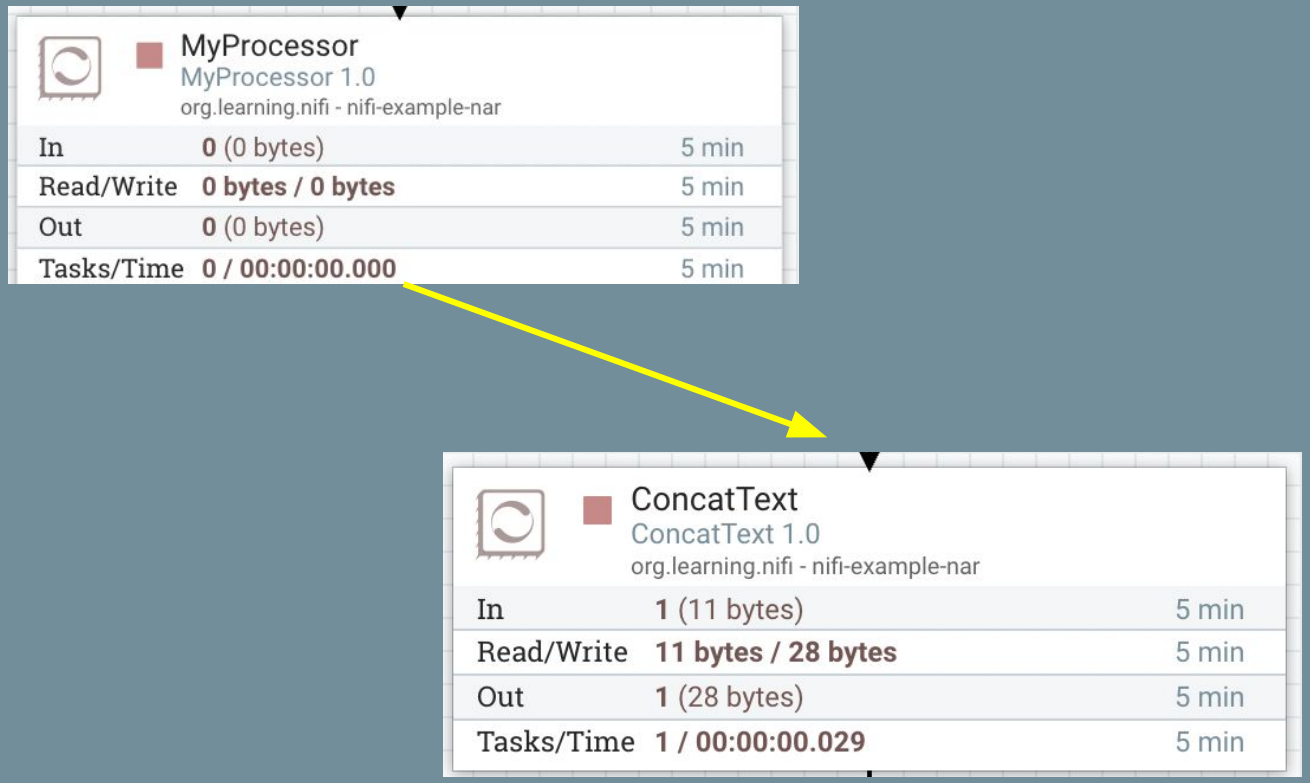
Renaming My Relationship



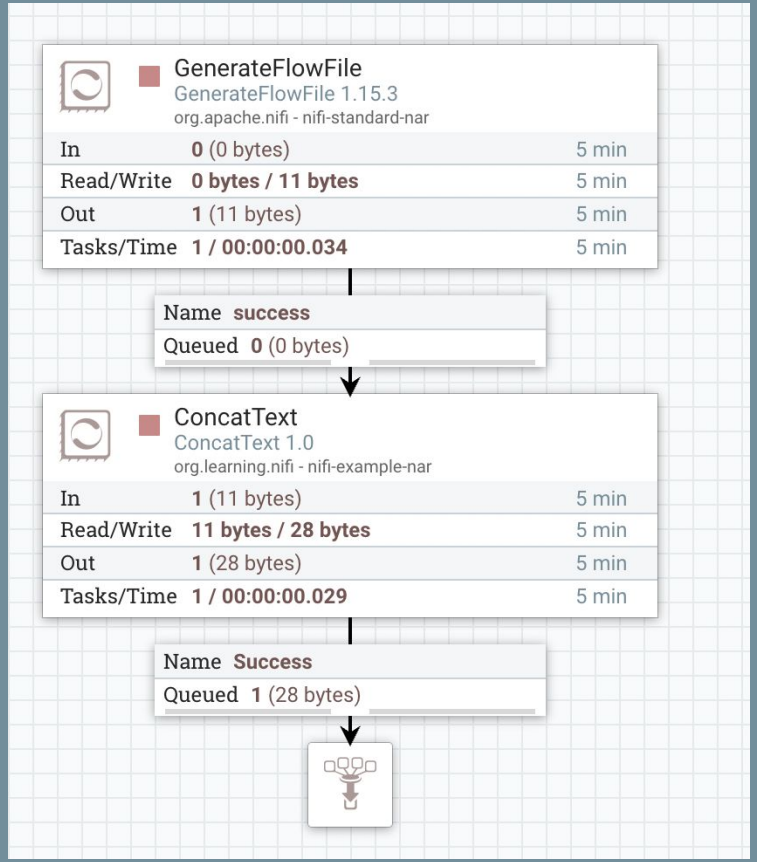
Another Important Change



Renaming My Relationship



ConcatText Processor



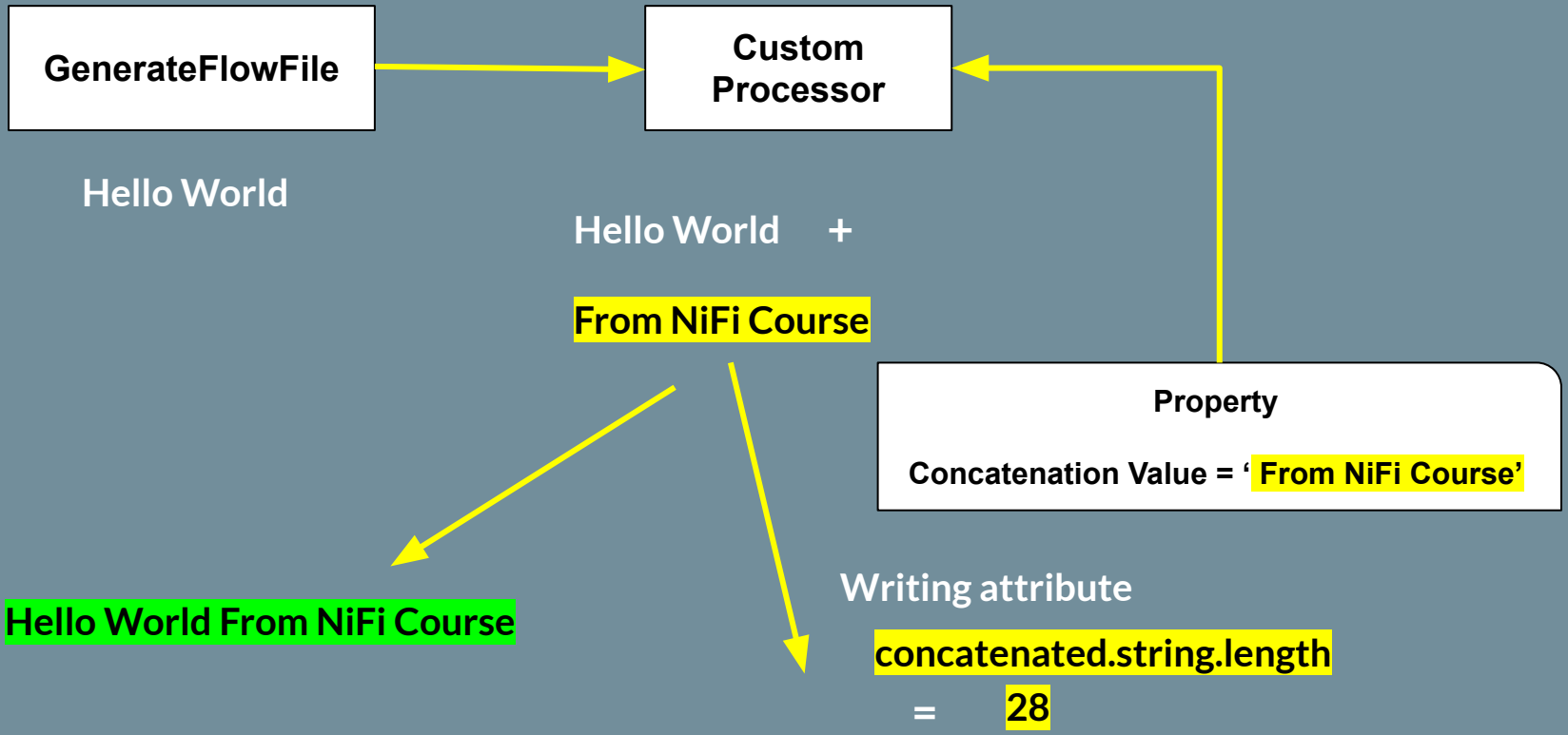
Reading and Writing FlowFile Attributes

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
concatenated.string.length	Length of concatenated string written as attribute to the

FlowFile

DETAILS | ATTRIBUTES

Attribute Values

concatenated.string.length
28

filename
dd811581-e4dc-40c9-832c-1bc06d47d070

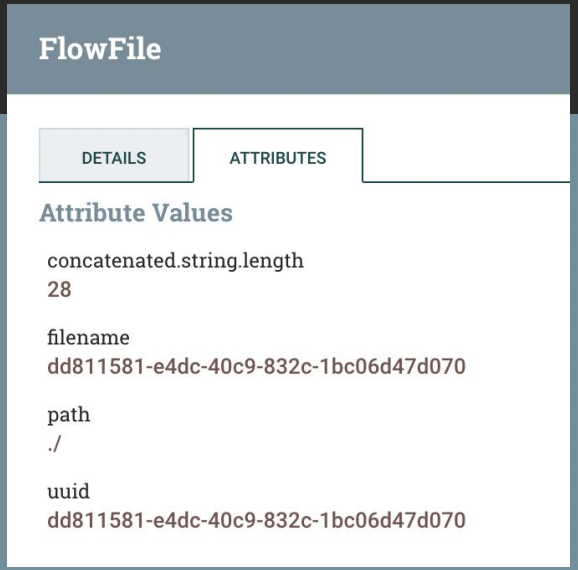
path
./

uuid
dd811581-e4dc-40c9-832c-1bc06d47d070

```
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

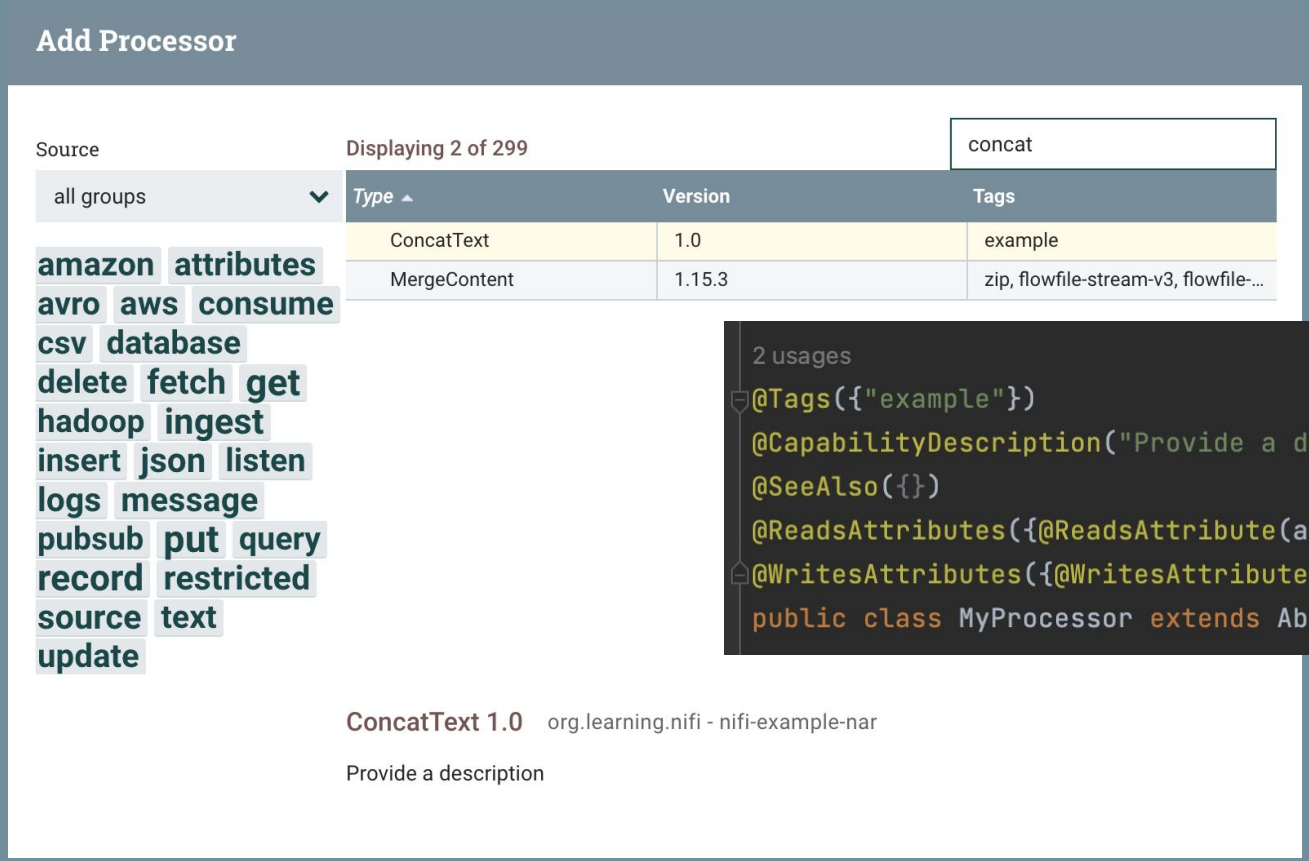
```
    // performing concat on incoming content
    final String resultantString = content.concat(property);

    // 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);
}
```



```java
@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);
}
```

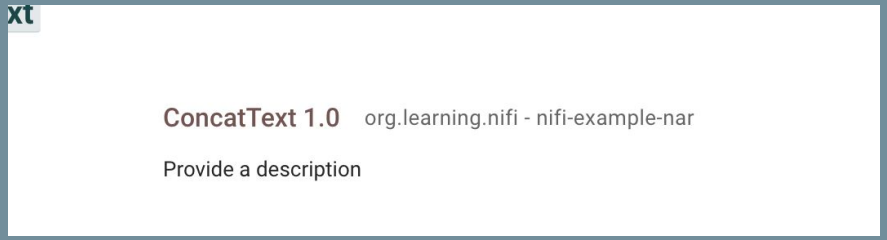
Writing Processor Description

Writing Processor Description




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

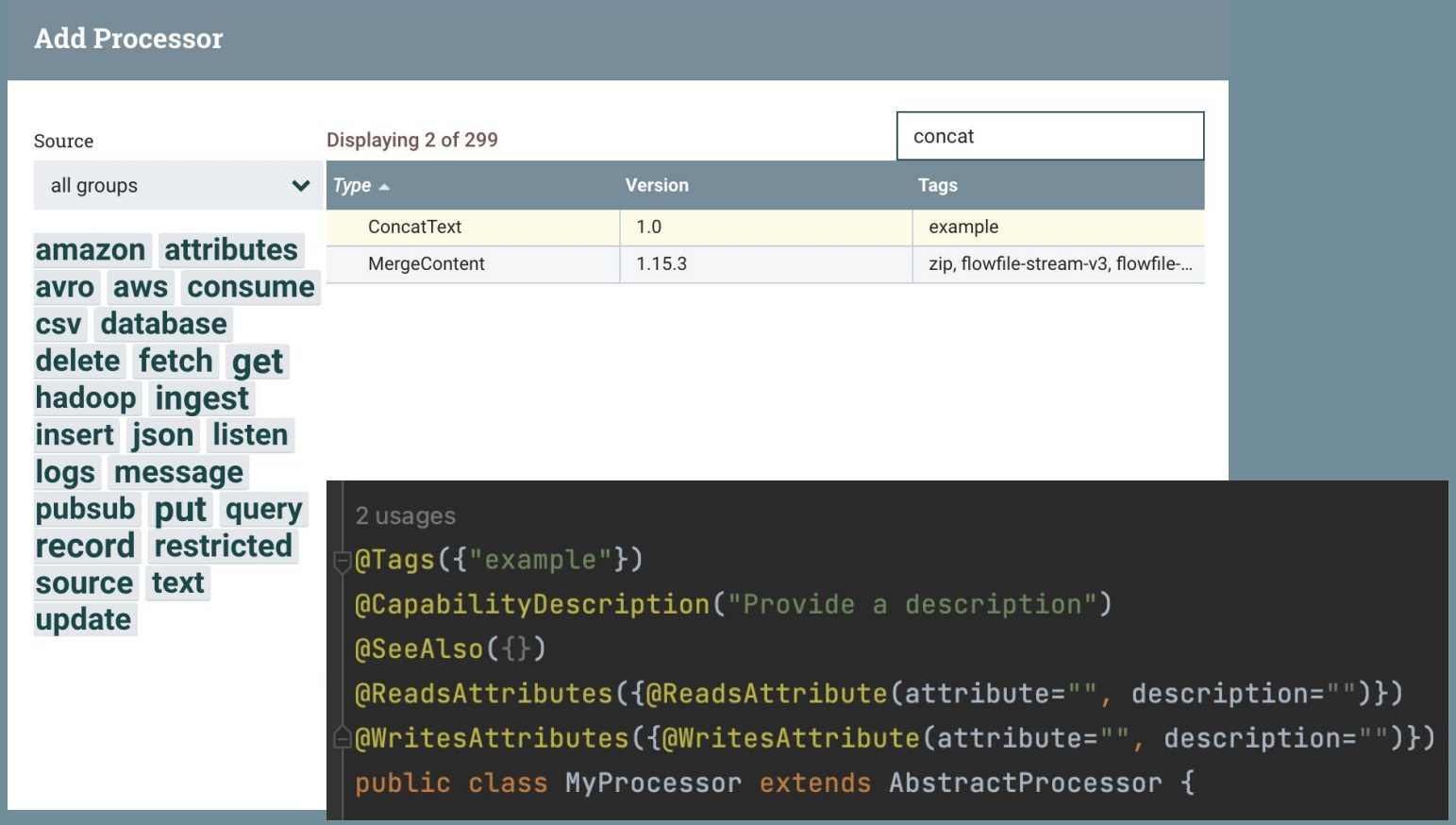
Writing Processor Description



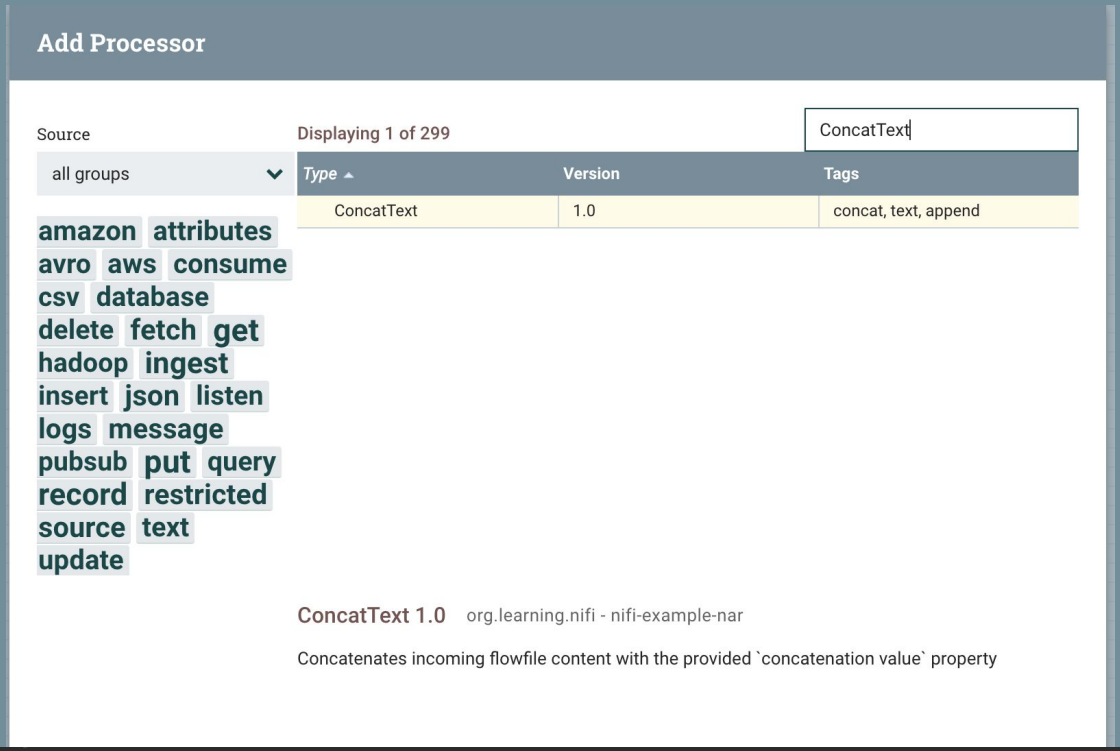

```
2 usages
@Tags({"example"})
@CapabilityDescription("Provide a description")
@SeeAlso({})
@ReadsAttributes({@ReadsAttribute(attribute="", description="")})
@WritesAttributes({@WritesAttribute(attribute="", description="")})
public class MyProcessor extends AbstractProcessor {

    2 usages
```

Processor Tags

Processor Tags



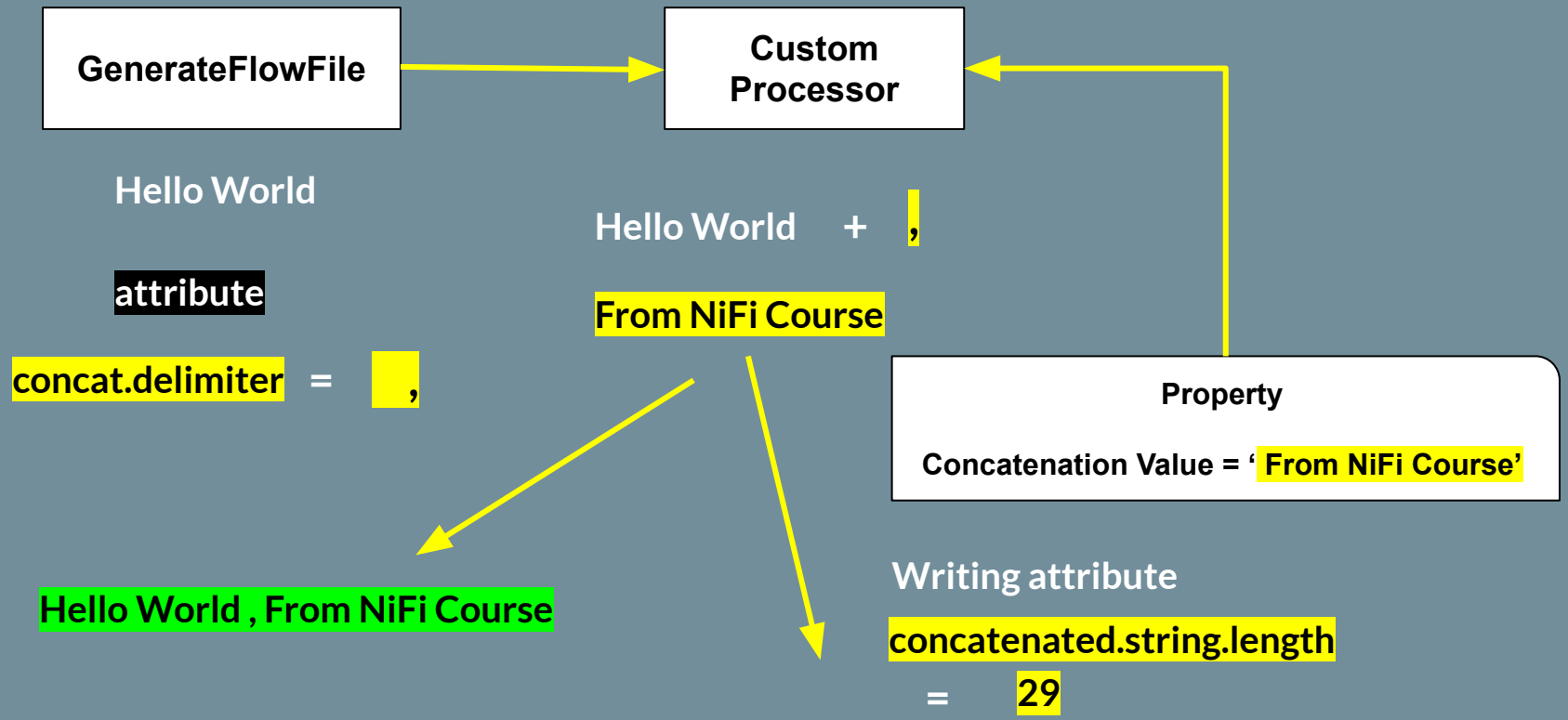
Processor Tags




```
@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

```java
@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.

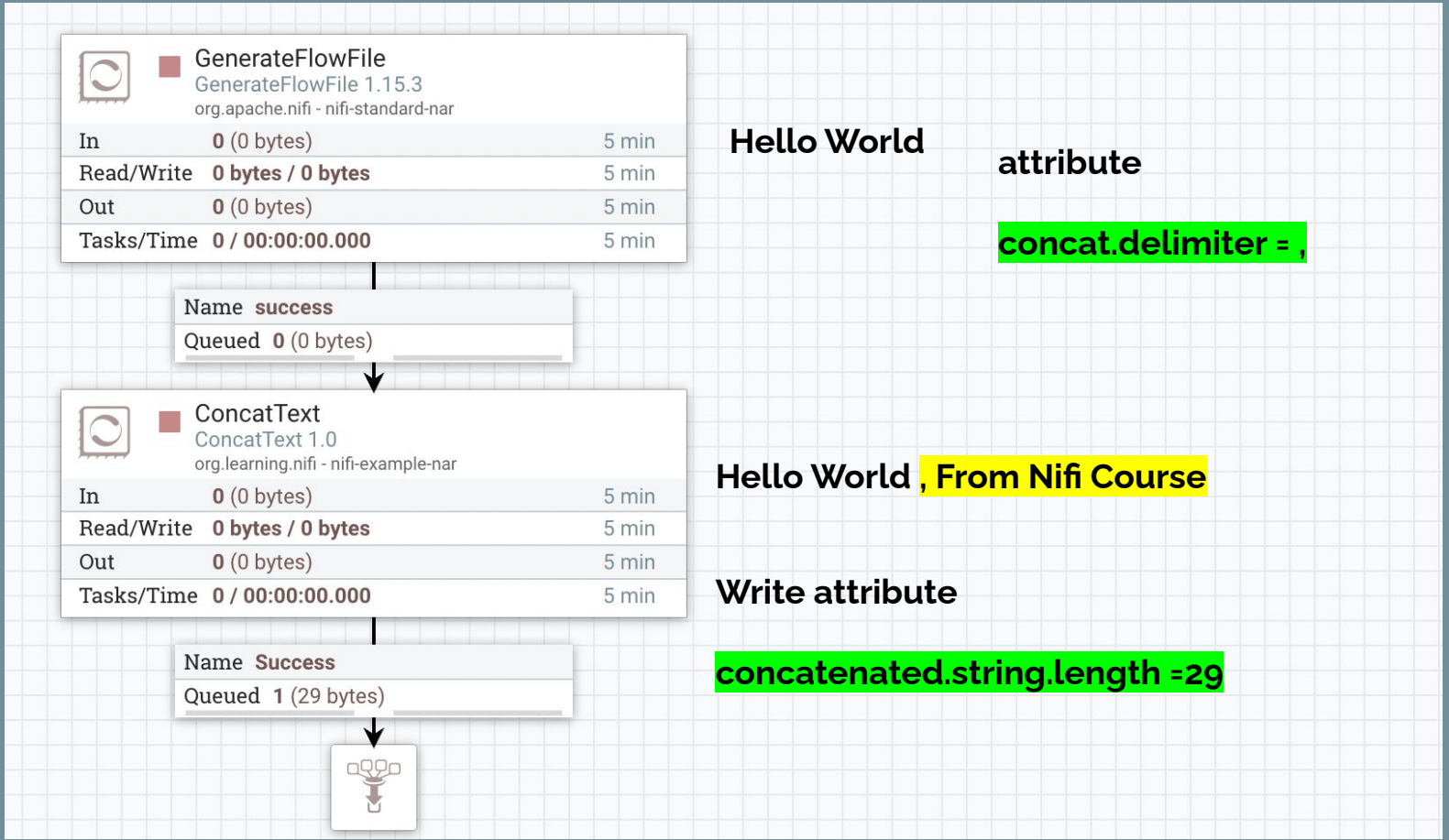
If the attribute doesn't exist, it will return null, in that providing empty string “”

```
// 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

```java
// 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



ConcatText Processor Result