TECO Research Group

Marcel Köpke
Matthias Budde
Till Riedel

Entwurfsdokument
Version 0.1

Visualizing & Mining of Geospatial Sensorstreams with Apache Kafka

---

Jean Baumgarten
Thomas Frank
Oliver Liu
Patrick Ries
Erik Wessel

1. Juli 2018

Inhaltsverzeichnis

1 Einleitung

2 Sequenzdiagramme

Die folgenden Sequenzdiagramme sollen den Ablauf von einzelnen Anwendungsfällen im PaVoS-System illustrieren. Die Interaktionen der Klassen miteinander in verschiedenen Situationen wird somit verdeutlicht.

2.1 Bridge

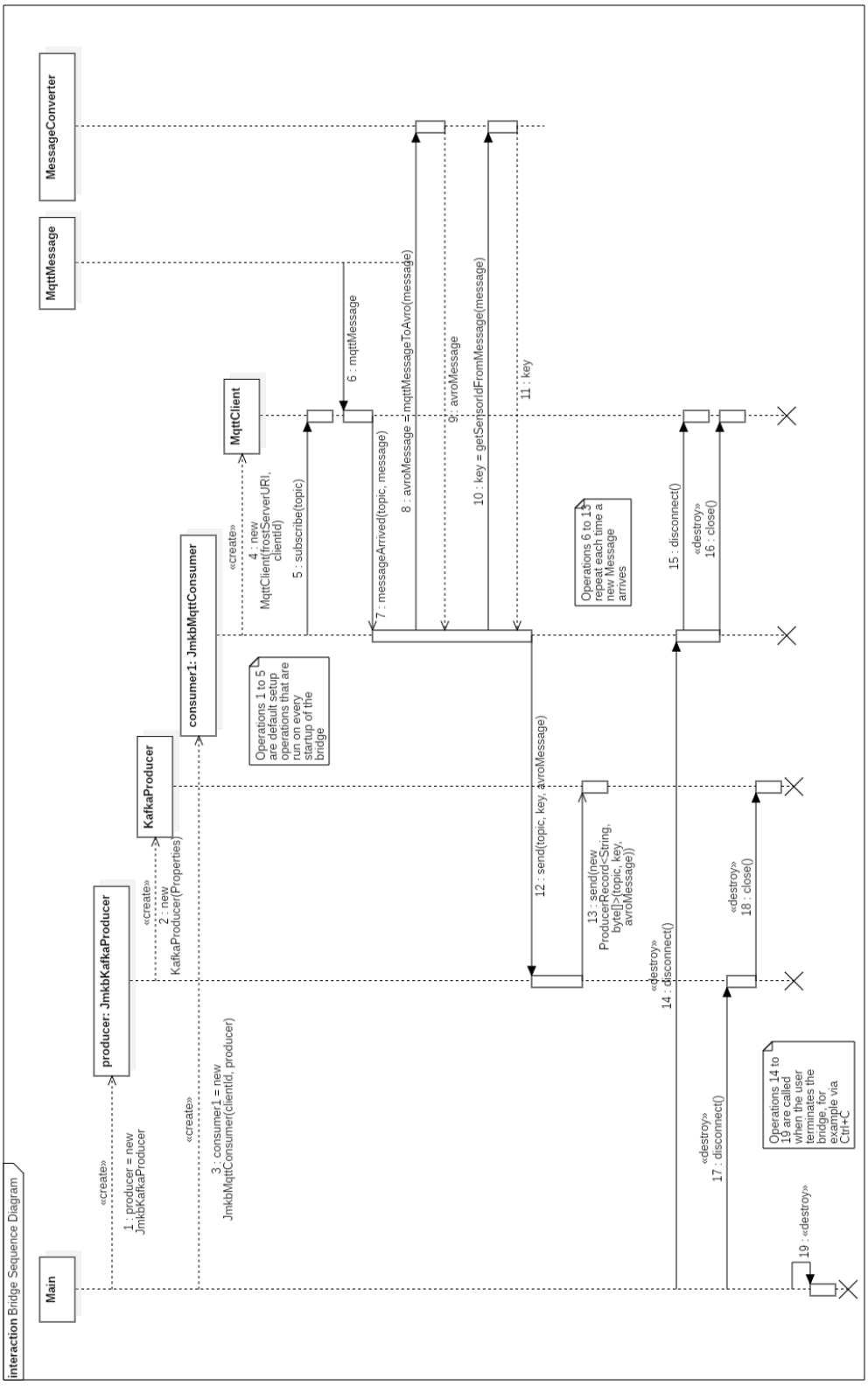
In diesem Sequenzdiagramm wird der Ablauf der Bridge beschrieben, die MQTT-Nachrichten in Records umwandelt und diese an Kafka weiterleitet. Die Bridge läuft komplett unabhängig vom restlichen System.
Die Bridge kann sich in einer von drei Phasen befinden:

1. **Aufbauphase:** Hier findet die Prüfung der Parameter und das Initialisieren der benötigten Klassen statt.
2. **Bereitschaftsphase:** Hier ist die Bridge bereit, Nachrichten von MQTT anzunehmen, zu konvertieren und an Kafka weiter zu senden.
3. **Abbauphase:** Hier werden die Verbindungen zu MQTT und Kafka getrennt, anschließend wird die Bridge beendet.

In der Aufbauphase (in diesem Diagramm Operationen 1-5) wird zunächst ein `JmkbKafkaProducer` erstellt, der intern einen `KafkaProducer` mit bestimmten Einstellungen initialisiert und eine Verbindung zum Kafka Broker aufbaut. Danach wird ein `JmkbMqttConsumer` erstellt, der intern einen `MqttClient` mit bestimmten Einstellungen initialisiert, welcher eine Verbindung zum MQTT-Server aufbaut und die Topics abonniert, die vom FROST-Server angeboten werden.

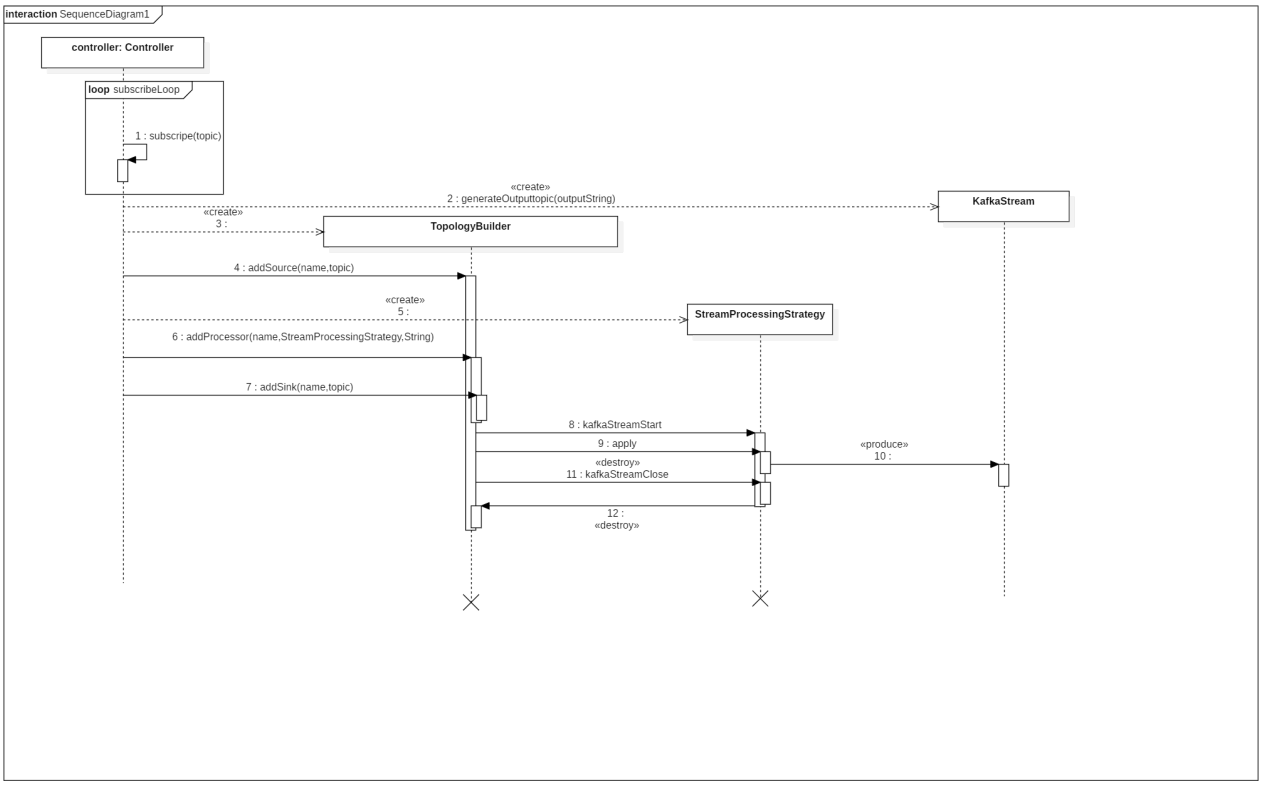
Nun beginnt die Bereitschaftsphase. Sobald eine Nachricht beim MqttClient ankommt, wird die Methode `messageArrived` des `JmkbMqttConsumer`s aufgerufen. In dieser Methode wird aus der erhaltenen Nachricht die IOT-ID des Sensors gefiltert und die Nachricht wird in das Avro-Format konvertiert. Diese zwei Daten sind dann key und value für das Kafka `ProducerRecord` und werden über einen Aufruf der `send`-Methode des `JmkbKafkaProducer`s in ein solches Format gewandelt. Anschließend wird das Record durch den KafkaProducer an Kafka gesendet.

In der Abbauphase werden die `disconnect`-Methoden von `JmkbMqttConsumer` und `JmkbKafkaProducer` aufgerufen, die jeweils die Verbindungen zu MQTT und Kafka sauber trennen und die Clients schließen. Die Abbauphase beginnt nur dann, wenn der Nutzer des Programms es willkürlich schließt oder das System es beendet.


interaction Bridge Sequence Diagram
Main
producer: JmkbKafkaProducer
KafkaProducer
consumer1: JmkbMqttConsumer
MqttClient
MqttMessage
MessageConverter
«create»
1 : producer = new JmkbKafkaProducer
«create»
2 : new KafkaProducer(Properties)
«create»
3 : consumer1 = new JmkbMqttConsumer(clientId, producer)
«create»
4 : new MqttClient(hostServerURI, clientId)
5 : subscribe(topic)
6 : mqttMessage
7 : messageArrived(topic, message)
8 : avroMessage = mqttMessageToAvro(message)
9: avroMessage
10 : key = getSensorIdFromMessage(message)
11 : key
12 : send(topic, key, avroMessage)
13 : send(new ProducerRecord<String, byte[]>(topic, key, avroMessage))
«destroy»
14 : disconnect()
«destroy»
15 : disconnect()
«destroy»
16 : close()
«destroy»
17 : disconnect()
«destroy»
18 : close()
19 : «destroy»
Operations 1 to 5 are default setup operations that are run on every startup of the bridge
Operations 6 to 13 repeat each time a new Message arrives
Operations 14 to 19 are called when the user terminates the bridge, for example via Ctrl+C

2.2 Core

Core

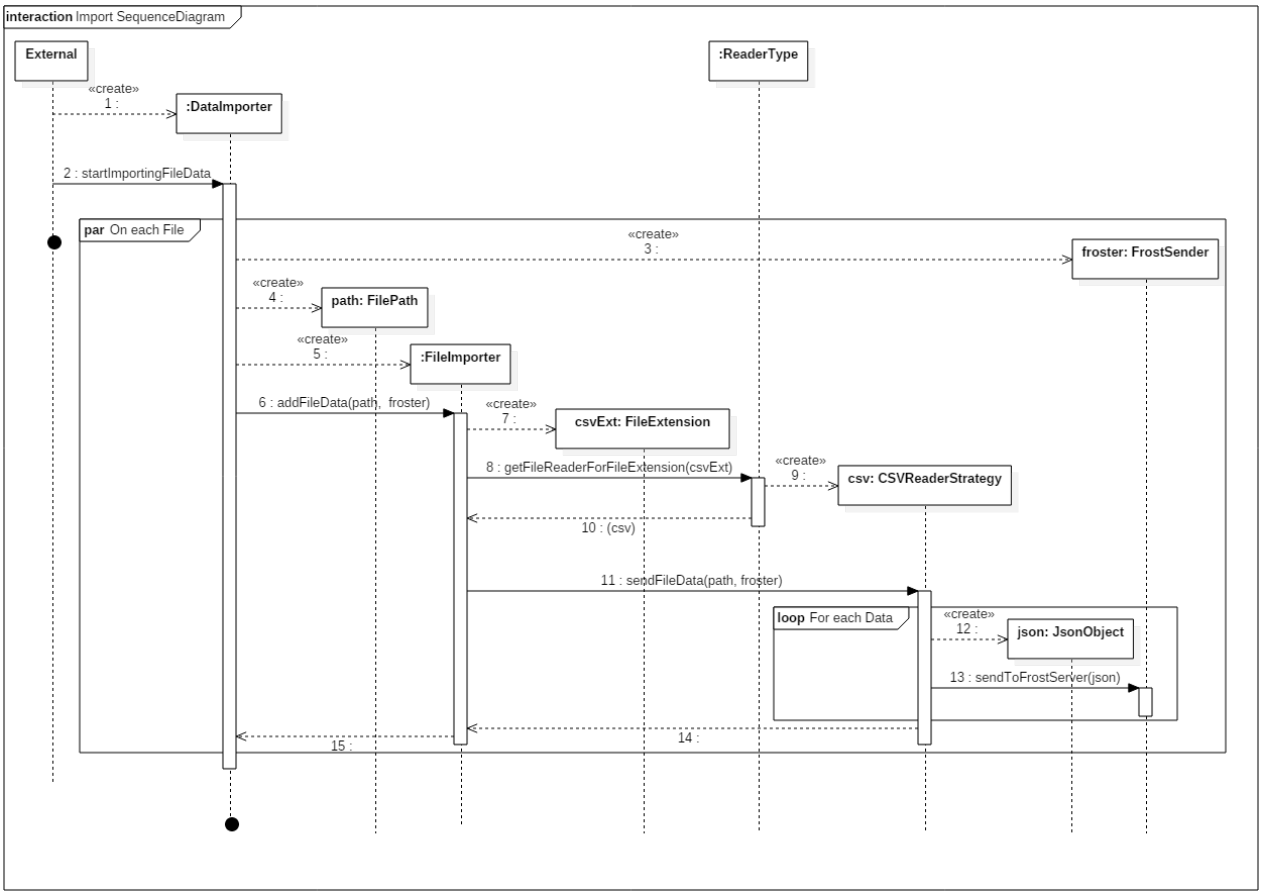

interaction SequenceDiagram1
controller: Controller
loop subscribeLoop
1 : subscripe(topic)
«create»
2 : generateOutputtopic(outputString)
KafkaStream
«create»
3 :
TopologyBuilder
4 : addSource(name,topic)
«create»
5 :
StreamProcessingStrategy
6 : addProcessor(name,StreamProcessingStrategy,String)
7 : addSink(name,topic)
8 : kafkaStreamStart
9 : apply
«produce»
10 :
«destroy»
11 : kafkaStreamClose
12 :
«destroy»

2.3 Import

Bei dem Import wird zuerst in dem Importordner nach Dateien gesucht und danach für jede vorhandene Datei ein separater Importprozess gestartet. Das folgende Sequenzdiagram stellt diesen Vorgang dar. Hier wird ausschließlich der Import behandelt, wer diesen Anstößt soll nicht Teil des Diagrams sein. `External` soll hier das Element darstellen, das den Import aufruft. Dazu wird ein `DataImporter` erstellt und seine Methode `startImportingFileData` aufgerufen, womit der Importvorgang startet.

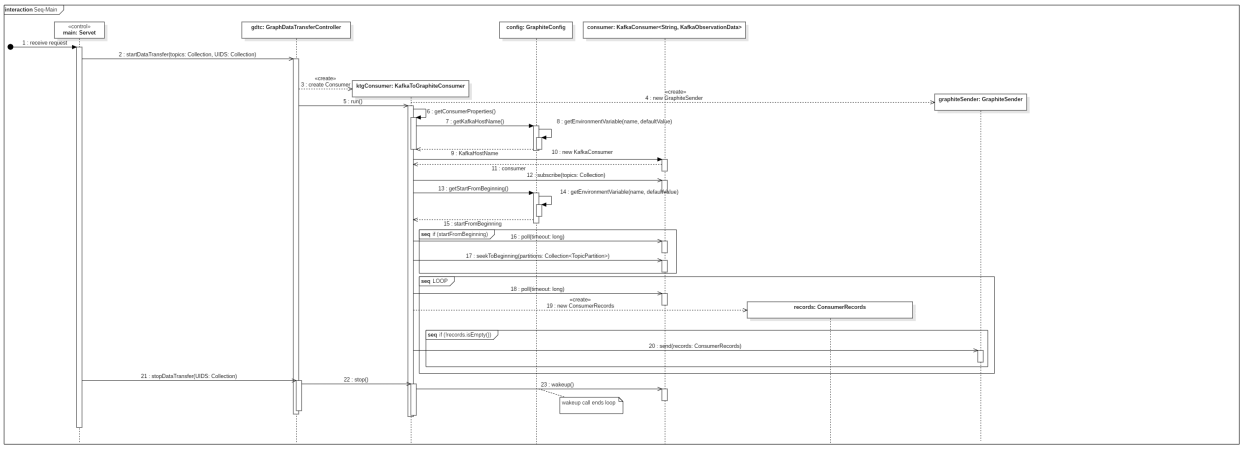
Für jede Datei in dem Importordner wird nun ein `FrostSender` und einen `FilePath` der zum Pfad der Datei passt. Ist dies geschehen wird der `FileImporter` für diese Datei erschaffen und mit `addFileData` gestartet. Dazu wird der Pfad und der `FrostSender` mitübergeben. Aus dem Pfad wird jetzt eine `FileExtension` generiert, die dazu genutzt wird über den `ReaderType` eine Instanz einer Implementierung der `FileReaderStrategy` zu erhalten. Ist die `FileExtension` nicht bekannt würde es hier zu einer Exception kommen und der Import für diese Datei beendet.

In diesem Fall wurde als Beispiel eine `CSVReaderStrategy` genommen. Diese übernimmt den tatsächlichen Import der Daten aus der Datei zum FROST-Server vor. Dazu werden nach und nach einzelne Datensätze aus der Datei ausgelesen und über den `FrostSender` an den Server gesendet.

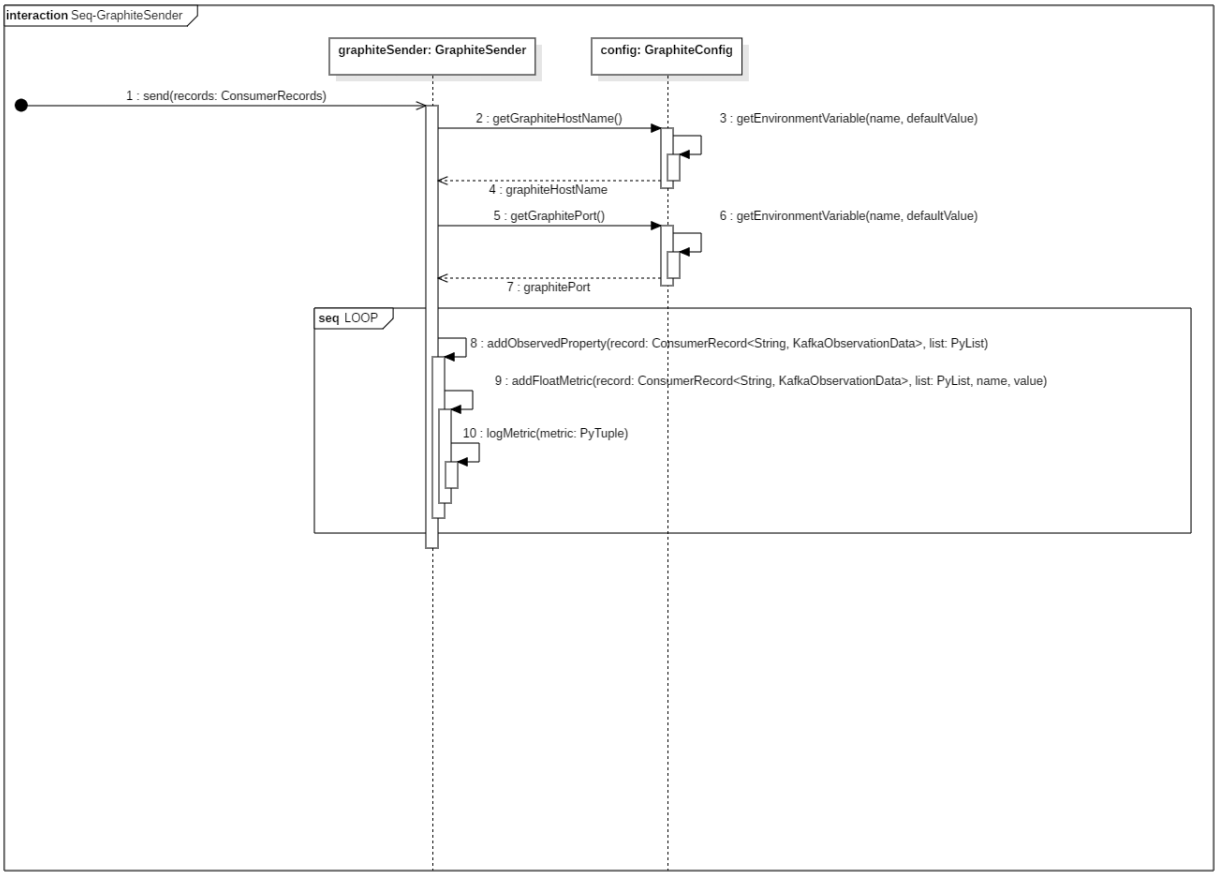

interaction Import SequenceDiagram
External
:ReaderType
«create»
1 :
:DataImporter
2 : startImportingFileData
par On each File
«create»
3 :
froster: FrostSender
«create»
4 :
path: FilePath
«create»
5 :
:FileImporter
6 : addFileData(path, froster)
«create»
7 :
csvExt: FileExtension
8 : getFileReaderForFileExtension(csvExt)
«create»
9 :
csv: CSVReaderStrategy
10 : (csv)
11 : sendFileData(path, froster)
loop For each Data
«create»
12 :
json: JsonObject
13 : sendToFrostServer(json)
14 :
15 :

2.4 Graphite

Graphite Main

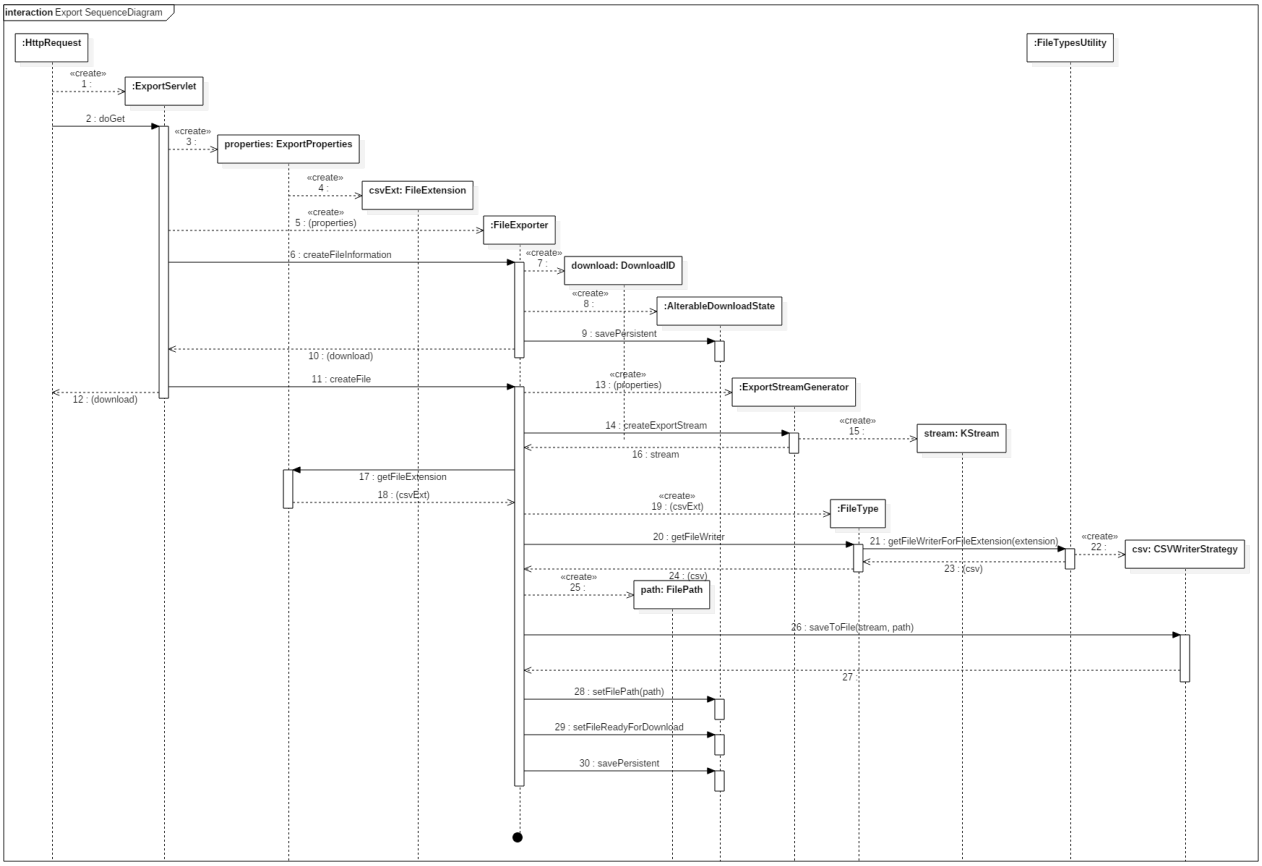


Graphite Sender



2.5 Export

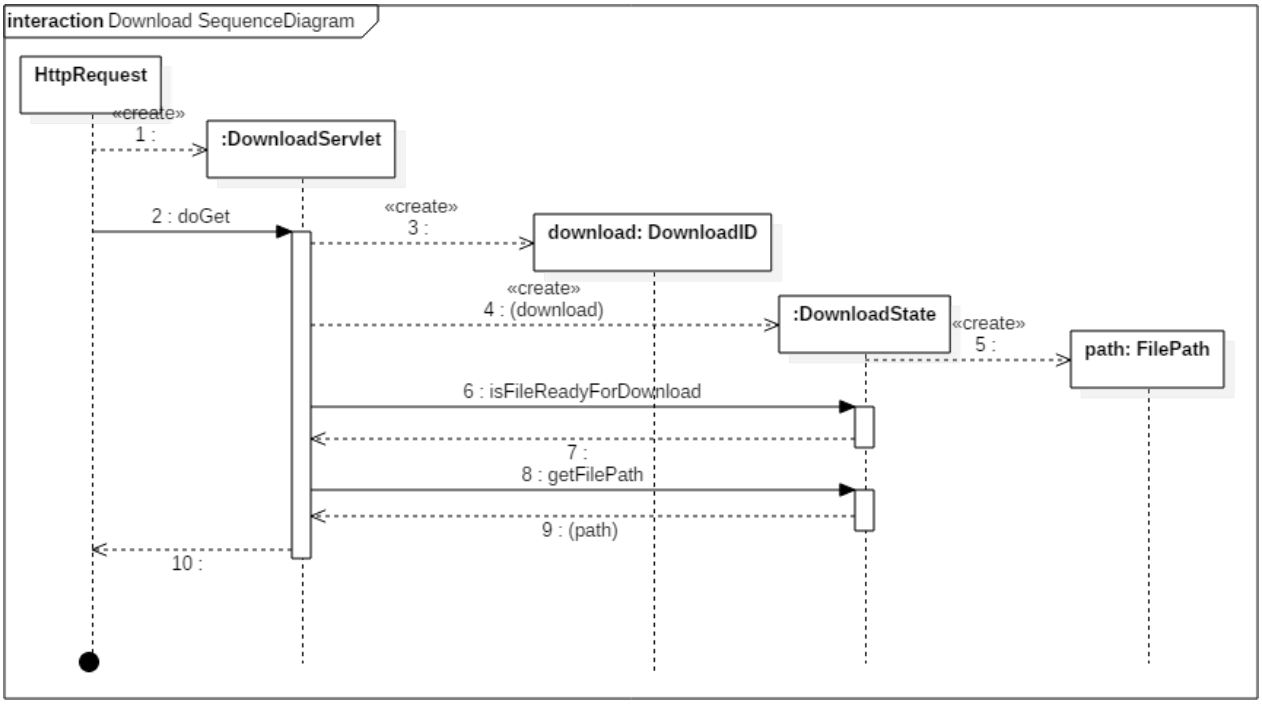
Export



Ein Download wird grundsätzlich von einem Nutzer aus einem Browserfenster angefragt. Dazu wird das `DownloadServlet` benutzt. Diese wird vom Server erstellt sobald eine Anfrage des Nutzer einkommt. Dann wird `doGet` aufgerufen und das Servlet beginnt mit seiner Aufgabe, die in diesem Sequenzdiagram dargestellt wird.

Die Anfrage des nutzers enthält eine DownloadID, die für eine bestimmte Datei auf dem Server steht. Diese wird benutzt um eine `DownloadID` Objekt zu erstellen, das dazu dient einen `DownloadState` zu konstruieren. Dieser holt sich, sobald er erstellt wurde, die Informationen zu dem betreffenden Download. Diese Informationen könnten in einer Datei liegen. Nun wird zuerst geprüft, ob die Datei bereit für den Download ist, dazu dient die Methode `isFileReadyForDownload`. Ist dies der Fall kann nun mit der `getFilePath`-Methode nach dem Pfad der Datei gefragt werden. Dieser wird nun vom `DownloadServlet` genutzt, um die Datei dem Nutzer zu schicken.

Der Vorgang bei dem `StatusServlet` ist sehr Ähnlich. Dort geht es darum in Erfahrung zu bringen, ob eine Download bereit ist, um zum Beispiel zu wissen, ob dem Nutzer bereits ein Download-Button gezeigt werden kann. Der einzige Unterschied liegt darin, dass dort nicht nach dem Pfad gesucht wird, sobndern gleich das Ergebnis der `isFileReadyForDownload` zurückgeschickt wird. Aus diesem Grund wurde darauf verzichtet ein separates Sequenzdiagram dafür zu erstellen.



3 Klassendiagramme

Class Hierarchy

Classes

- java.lang.Object
  - Bridge.JmkbKafkaProducer (in 3.1.1, page 15)
  - Bridge.JmkbMqttConsumer (in 3.1.2, page 16)
  - Bridge.MessageConverter (in 3.1.3, page 18)
  - Bridge.PropertiesFileReader (in 3.1.4, page 19)
  - Bridge.SchemaRegistryConnector (in 3.1.5, page 20)

3.1 Package Bridge

Package Contents *Page*

Classes

3.1.1 Class JmkbKafkaProducer

This class creates a Kafka producer using defined settings and publishes records to the Kafka Cluster.

Declaration

```
public class JmkbKafkaProducer
 extends java.lang.Object
```

Constructor summary

JmkbKafkaProducer() Default constructor

Method summary

disconnect() Disconnects this Kafka producer from the Kafka Cluster and closes the producer.
send(String, byte[]) Asynchronously sends a record to the topic.

Constructors

- **JmkbKafkaProducer**

  ```
  public JmkbKafkaProducer()
  ```

  - **Description**

    Default constructor

Methods

- **disconnect**

  ```
  public void disconnect()
  ```

  - **Description**

    Disconnects this Kafka producer from the Kafka Cluster and closes the producer.

- **send**

  ```
  public void send(java.lang.String topic,byte[] avroMessage)
  ```

  - **Description**

    Asynchronously sends a record to the topic.

  - **Parameters**

    * `topic` – The topic.
    * `avroMessage` – The message to send.

3.1.2 Class JmkbMqttConsumer

This class serves as an MqttClient that consumes messages from the specified FROST-Server address. On message arrival, it will initiate the conversion of the message to a desired format via MqttMessageConverter and supply the converted message to a JmkbKafkaProducer. An instance of this class should be destroyed with a call to the disconnect() method.

Declaration

```
public class JmkbMqttConsumer
 extends java.lang.Object
```

Constructor summary

JmkbMqttConsumer() Default constructor

Method summary

connectionLost(Throwable) This method is called when the connection to the server is lost.
deliveryComplete(IMqttDeliveryToken) Called when delivery for a message has been completed, and all acknowledgments have been received.
disconnect() Disconnects client from MQTT and closes the client.
JmkbMqttConsumer(String, JmkbKafkaProducer) This constructor for this class.
messageArrived(String, MqttMessage) This method is called when a message arrives from the server.

Constructors

- **JmkbMqttConsumer**

  ```
  public JmkbMqttConsumer()
  ```

- **Description**

  Default constructor

Methods

- **connectionLost**

  ```
  public void connectionLost(java.lang.Throwable cause)
  ```

  - **Description**

    This method is called when the connection to the server is lost.

  - **Parameters**

    * `cause` – the reason behind the loss of connection.

- **deliveryComplete**

  ```
  public void deliveryComplete(IMqttDeliveryToken token)
  ```

  - **Description**

    Called when delivery for a message has been completed, and all acknowledgments have been received. In this implementation of this method, nothing happens.

  - **Parameters**

    * `token` – the delivery token associated with the message.

- **disconnect**

  ```
  public void disconnect()
  ```

  - **Description**

    Disconnects client from MQTT and closes the client.

- **JmkbMqttConsumer**

  ```
  public void JmkbMqttConsumer(java.lang.String clientId,
      JmkbKafkaProducer producer)
  ```

  - **Description**

    This constructor for this class. Creates a new MqttClient and subscribes to the topics specified in the SensorThings API standard. A unique identifier and a JmkbKafkaProducer should be supplied.

- **Parameters**
  - `clientId` – The unique identifier for the MqttClient.
  - `producer` – A JmkbKafkaProducer.

- **messageArrived**

```
public void messageArrived(java.lang.String topic,MqttMessage
    message)
```

  - **Description**

    This method is called when a message arrives from the server. This method is invoked synchronously by the MQTT client. An acknowledgment is not sent back to the server until this method returns cleanly. Any additional messages which arrive while this method is running will build up in memory, and will then back up on the network. When this method is called, the supplied message will be converted to an Avro message and forwarded to an instance of JmkbKafkaProducer, which will then send the message to the Kafka Cluster.

  - **Parameters**
    - `topic` – name of the topic on the message was published to
    - `message` – the actual message.

3.1.3 Class MessageConverter

This convenience class provides static methods to convert a given message to another format.

Declaration

```
public class MessageConverter
 extends java.lang.Object
```

Constructor summary

MessageConverter() Default constructor

Method summary

getSensorIdFromMessage(byte[]) This method returns the sensor ID that has supplied the information in the message.

mqttMessageToAvro(MqttMessage) This method converts a given MqttMessage, which contains information in the JSON format, to an Avro message in a byte array.

Constructors

- **MessageConverter**

  ```
  public MessageConverter()
  ```

  - **Description**

    Default constructor

Methods

- **getSensorIdFromMessage**

  ```
  public static java.lang.String getSensorIdFromMessage(byte[] message
      )
  ```

  - **Description**

    This method returns the sensor ID that has supplied the information in the message. In detail, this method searches for the key 'iot.id' in the message and returns the value associated with the key.

  - **Parameters**
    - `message` – The message from which to extract the sensor ID.
  - **Returns** – The sensor ID.

- **mqttMessageToAvro**

  ```
  public static byte[] mqttMessageToAvro(MqttMessage message)
  ```

  - **Description**

    This method converts a given MqttMessage, which contains information in the JSON format, to an Avro message in a byte array.

  - **Parameters**
    - `message` – The message to convert.
  - **Returns** – The message in Avro format.

3.1.4 Class PropertiesFileReader

A class that reads properties from the configuration file (jmkb.properties) and provides a method for getting a property by key.

Declaration

```
public class PropertiesFileReader
 extends java.lang.Object
```

Constructor summary

PropertiesFileReader() Default constructor

Method summary

getProperty(String) Searches for the property with the specified key in jmkb.property.

Constructors

- **PropertiesFileReader**

  ```
  public PropertiesFileReader()
  ```

  - **Description**

    Default constructor

Methods

- **getProperty**

  ```
  public void getProperty(java.lang.String key)
  ```

  - **Description**

    Searches for the property with the specified key in jmkb.property.
  - **Parameters**
    - `key` – The value associated with the key or null if the key is not found.

3.1.5 Class SchemaRegistryConnector

Convenience class which provides methods for interacting with the schema registry.

Declaration

```
public class SchemaRegistryConnector
 extends java.lang.Object
```

Constructor summary

SchemaRegistryConnector() Default constructor

Method summary

getSchemaById(int) Requests the schema associated with the schema ID from the schema registry.

getSchemaBySubject(String) Requests the latest version of the schema associated with the given subject from the schema registry.

getSchemaBySubject(String, int) Requests the given version of the schema associated with the given subject from the schema registry.

Constructors

- **SchemaRegistryConnector**

  ```
  public SchemaRegistryConnector()
  ```

  - **Description**

    Default constructor

Methods

- **getSchemaById**

  ```
  public java.lang.String getSchemaById(int id)
  ```

  - **Description**

    Requests the schema associated with the schema ID from the schema registry. Returns the schema if successful, null if not.

  - **Parameters**

    * `id` – The schema id.

  - **Returns** – The schema if successful, null if not.

- **getSchemaBySubject**

  ```
  public java.lang.String getSchemaBySubject(java.lang.String subject)
  ```

  - **Description**

    Requests the latest version of the schema associated with the given subject from the schema registry. Returns the schema if successful, null if not.

  - **Parameters**

    * `subject` – The subject of the schema.

  - **Returns** – The schema if successful, null if not.

- **getSchemaBySubject**

  ```
  public java.lang.String getSchemaBySubject(java.lang.String subject,
      int version)
  ```

  - **Description**

    Requests the given version of the schema associated with the given subject from the schema registry. Returns the schema if successful, null if not.

  - **Parameters**
    - `subject` – The subject of the schema.
    - `version` – The schema version.
  - **Returns** – the schema if successful, null if not.

Class Hierarchy

Classes

- java.lang.Object
    - CommandRequestPattern.GetClusterCommand
    - CommandRequestPattern.GetSensorCommand
    - CommandRequestPattern.GetTileCommand
    - CommandRequestPattern.Replier
    - CommandRequestPattern.Requestor
    - ConfigGUI.JFrame
        - ConfigGUI.DeleteFrame
        - ConfigGUI.MainFrame
        - ConfigGUI.SensorFrame
    - Controller.ClusterProcessStrategy
    - Controller.CombinerProcessStrategy
    - Controller.Controller
    - Controller.ExportProcessStrategy
    - Controller.GraphiteProcessStrategy
    - Controller.TopologyBuilder
    - Controller.UncaughtExceptionHandler
    - Properties.PropertiesFile

Interfaces

- CommandRequestPattern.RequestCommand
- CommandRequestPattern.StreamProcessingStrategy
- Properties.PropertiesFileInterface

3.2 Package CommandRequestPattern

Package Contents *Page*

Interfaces

Classes

This Command request a Sensor in the System.

GetTileCommand ..... ??
This Command request a Tile in the System.

Replier ..... ??
This Class handels the Requests and Replies to them

Requestor ..... ??
The Implemente this class and request something to the System and a Replier answer to it.

3.2.1 Interface RequestCommand

All CommandsRequest implements this Interface. CommandRequest are sendet form the View to request something out of the System.

Declaration

```
public interface RequestCommand
```

All known subinterfaces

GetTileCommand , GetSensorCommand , GetClusterCommand

All classes known to implement interface

GetTileCommand , GetSensorCommand , GetClusterCommand

Method summary

execute() This is the Execution form the requested Command
getObject() This Method Return the Requested Object

Methods

- **execute**

  ```
  void execute()
  ```

  - **Description**

    This is the Execution form the requested Command

- **getObject**

  ```
  void getObject()
  ```

- **Description**

  This Method Return the Requested Object

3.2.2 Interface StreamProcessingStrategy

This Class is a Interface for the Stream Builder Applications which genereates an Output topic to provides data transformations. The ProcessingApplication will use Kafka DSL API to process the data.

Declaration

```
public interface StreamProcessingStrategy
```

Method summary

apply() This Methode definite the Process of the Application.
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Methods

- **apply**

  ```
  boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Process got Successfully worked

- **kafkaStreamClose**

  ```
  boolean kafkaStreamClose()
  ```

  - **Description**

    This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.
  - **Returns** – true if the Kafka Stream closed, false otherwise

- **kafkaStreamStart**

  ```
  boolean kafkaStreamStart()
  ```

- **Description**

  This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.

- **Returns** – true if the Kafka Stream Started false otherwise

3.2.3 Class GetClusterCommand

This Command request a Cluster in the System.

Declaration

```
public class GetClusterCommand
 extends java.lang.Object implements RequestCommand
```

Constructor summary

GetClusterCommand() Default constructor

Method summary

execute() This is the Execution form the requested Command.
getObject() This Method Return the Requested Cluster as a KStream

Constructors

- **GetClusterCommand**

  ```
  public GetClusterCommand()
  ```

  - **Description**

    Default constructor

Methods

- **execute**

  ```
  public void execute()
  ```

  - **Description**

    This is the Execution form the requested Command. So it will search for the Cluster

- **getObject**

```
public void getObject()
```

- **Description**

  This Method Return the Requested Cluster as a KStream

3.2.4 Class GetSensorCommand

This Command request a Sensor in the System.

Declaration

```
public class GetSensorCommand
 extends java.lang.Object implements RequestCommand
```

Constructor summary

GetSensorCommand() Default constructor

Method summary

execute() This is the Execution form the requested Command.
getObject() This Method Return the Requested Sensor as a KStream

Constructors

- **GetSensorCommand**

```
public GetSensorCommand()
```

  - **Description**

    Default constructor

Methods

- **execute**

```
public void execute()
```

  - **Description**

    This is the Execution form the requested Command. So it will search for the Sensor Uid

- **getObject**

```
public void getObject()
```

- **Description**

  This Method Return the Requested Sensor as a KStream

3.2.5 Class GetTileCommand

This Command request a Tile in the System.

Declaration

```
public class GetTileCommand
 extends java.lang.Object implements RequestCommand
```

Constructor summary

GetTileCommand() Default constructor

Method summary

execute() This is the Execution form the requested Command.
getObject() This Method Return the Requested Tile as a KStream

Constructors

- **GetTileCommand**

  ```
  public GetTileCommand()
  ```

  - **Description**

    Default constructor

Methods

- **execute**

  ```
  public void execute()
  ```

  - **Description**

    This is the Execution form the requested Command. So it will search for the Tile

- **getObject**

  ```
  public void getObject()
  ```

  - **Description**

    This Method Return the Requested Tile as a KStream

3.2.6 Class Replier

This Class handels the Requests and Replies to them

Declaration

```
public class Replier
 extends java.lang.Object
```

Constructor summary

Replier() Default constructor

Method summary

initialize(Connection, String) This is the initialisation Method for the Replier to connect to different Requestors

onMessage(Message, RequestCommand) This Methode triggers something in the System waht has to be done

Constructors

- **Replier**

  ```
  public Replier()
  ```

  - **Description**

    Default constructor

Methods

- **initialize**

  ```
  public void initialize(Connection connection,java.lang.String
      requestQueueName)
  ```

  - **Description**

    This is the initialisation Method for the Replier to connect to different Requestors

  - **Parameters**

    * `connection` – This is the Connection parameter, so taht the replier knows where he answers
    * `requestQueueName` – This a Simple name for the request Queue

- **onMessage**

  ```
  public void onMessage(Message message,RequestCommand request)
  ```

  - **Description**

    This Methode triggers something in the System waht has to be done

  - **Parameters**

    * `message` – This is a simple Message parameter
    * `request` – This is the RequestCommand Object wich Contains the Real request.

3.2.7 Class Requestor

The Implemente this class and request something to the System and a Replier answer to it.

Declaration

```
public class Requestor
 extends java.lang.Object
```

Constructor summary

Requestor() Default constructor

Method summary

initialize(Connection)
receiveSync(RequestCommand) This Methode is there to got the Request again when it get lost or something
send(RequestCommand)

Constructors

- **Requestor**

  ```
  public Requestor()
  ```

  - **Description**

    Default constructor

Methods

- **initialize**

  ```
  public void initialize(Connection connection)
  ```

  - **Parameters**
    - `connection` – This is the Connection parameter, so taht the repuestor knows where he requests something

- **receiveSync**

  ```
  public RequestCommand receiveSync(RequestCommand request)
  ```

  - **Description**

    This Methode is there to got the Request again when it get lost or something
  - **Parameters**
    - `request` – It Returns the Requested RequestCommand
  - **Returns** – A RequestCommand which contains a Request for a RequestCommand

- **send**

  ```
  public boolean send(RequestCommand request)
  ```

  - **Parameters**
    - `request` – This is the RequestCommand Object wich Conntains the Real request.
  - **Returns** – true if the RequestCommand got send and false otherwise

3.3 Package ConfigGUI

Package Contents *Page*

Classes

3.3.1 Class DeleteFrame

This Frame is the Delete Frame, where you delete Topics out of the Programm

Declaration

```
public class DeleteFrame
 extends ConfigGUI.JFrame
```

Constructor summary

DeleteFrame() Default constructor

Constructors

- **DeleteFrame**

  ```
  public DeleteFrame()
  ```

  - **Description**

    Default constructor

3.3.2 Class JFrame

This is the Basic Interface from Java for building a Frame.

Declaration

```
public class JFrame
 extends java.lang.Object
```

All known subclasses

SensorFrame , MainFrame , DeleteFrame

Constructor summary

JFrame() Default constructor

Constructors

- **JFrame**

  ```
  public JFrame()
  ```

- **Description**

  Default constructor

3.3.3 Class MainFrame

This Class holds the main functionality of the PaVoS program. It starts/stops the whole System and manages the export/import.

Declaration

```
public class MainFrame
 extends ConfigGUI.JFrame
```

Constructor summary

MainFrame() Default constructor

Constructors

- **MainFrame**

  ```
  public MainFrame()
  ```

  - **Description**

    Default constructor

3.3.4 Class SensorFrame

This Frame hold the data of all possible Sensors in the System.

Declaration

```
public class SensorFrame
 extends ConfigGUI.JFrame
```

Constructor summary

SensorFrame() Default constructor

Constructors

- **SensorFrame**

  ```
  public SensorFrame()
  ```

– **Description**

Default constructor

3.4 Package Controller

Package Contents *Page*

Classes

3.4.1 Class ClusterProcessStrategy

This Class is for the generation of the Clusters for the View. It Generates a Cluster Outputtopic

Declaration

```
public class ClusterProcessStrategy
 extends java.lang.Object
```

Constructor summary

ClusterProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.

kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **ClusterProcessStrategy**

  ```
  public ClusterProcessStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Cluster Process got Successfully worked, false otherwise

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

  - **Description**

    This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.
  - **Returns** – true if the Kafka Stream closed false otherwise

- **kafkaStreamStart**

  ```
  public boolean kafkaStreamStart()
  ```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.
  - **Returns** – true if the Kafka Stream Started, false otherwise

3.4.2 Class CombinerProcessStrategy

This Class does combinate the Clusters to bigger Cluster for the Different Zoom Levels

Declaration

```
public class CombinerProcessStrategy
 extends java.lang.Object
```

Constructor summary

CombinerProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **CombinerProcessStrategy**

  ```
  public CombinerProcessStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Combiner Process got Successfully worked

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

- **Description**

  This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.

- **Returns** – true if the Kafka Stream closed, false otherwise

- **kafkaStreamStart**

```
public boolean kafkaStreamStart()
```

- **Description**

  This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.

- **Returns** – true if the Kafka Stream Started false otherwise

3.4.3 Class Controller

This Class is the ControllerClass which manages the Requests and start new TopologyBuilders to start new Processing Application.

Declaration

```
public class Controller
 extends java.lang.Object
```

Constructor summary

Controller() Default constructor

Method summary

generateOutputtopic(String) This Method generates a Output Topic, which uses a ProcessApplikation as OutputSink.
init() This Method initialise the Controler
setProperties(PropertiesFileInterface) This Method sets the Properties File
setTopolgyBuilder(StreamProcessingStrategy, String, String) Thsi Method starts a TopolgyBuilder to start a Kafka Stream Process.
subscribe(String) This method subscribe the controller to the Input Kafka Stream
workRequest(RequestCommand) This Method process the single Reuqest form the View

Constructors

- **Controller**

  ```
  public Controller()
  ```

  - **Description**

    Default constructor

Methods

- **generateOutputtopic**

  ```
  public boolean generateOutputtopic(java.lang.String topic)
  ```

  - **Description**

    This Method generates a Output Topic, which uses a ProcessApplikation as OutputSink. This will use Apache Avro Format.
  - **Parameters**
    - `topic` – topic name of the new Topic in Kafka
  - **Returns** – true when the Output Topic got successful generated

- **init**

  ```
  public boolean init()
  ```

  - **Description**

    This Method initialise the Controler
  - **Returns** – true when the initialise was successful and false otherwise

- **setProperties**

  ```
  public void setProperties(PropertiesFileInterface props)
  ```

  - **Description**

    This Method sets the Properties File
  - **Parameters**
    - `props` – props is the Propertyfile form where the controller reads his Settings

- **setTopolgyBuilder**

```
public void setTopolgyBuilder(StreamProcessingStrategy process,java.
    lang.String inputTopic,java.lang.String outputTopic)
```

- **Description**

  Thsi Method starts a TopolgyBuilder to start a Kafka Stream Process.

- **Parameters**
  - `process` – process name of the Process Application
  - `inputTopic` – inputTopic of the Kafka Topic
  - `outputTopic` – outputTopic of the Kafka Topic

- **subscribe**

```
public void subscribe(java.lang.String topic)
```

- **Description**

  This method subscribe the controller to the Input Kafka Stream

- **Parameters**
  - `topic` – The Name of the Topic which you want to subscribe

- **workRequest**

```
public void workRequest(RequestCommand command)
```

- **Description**

  This Method process the single Reuqest form the View

- **Parameters**
  - `command` – command is Instance of the RequestCommand Interface which contains a Job Request

3.4.4 Class ExportProcessStrategy

This Class is for The Processing of the Export Stream and it generates a Output Stream

Declaration

```
public class ExportProcessStrategy
 extends java.lang.Object
```

Constructor summary

ExportProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.
ExportApplication(ExportProperties) This is the default Contructer for the Export Process
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **ExportProcessStrategy**

  ```
  public ExportProcessStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Export Process got Successfully worked.

- **ExportApplication**

  ```
  public void ExportApplication(ExportProperties props)
  ```

  - **Description**

    This is the default Contructer for the Export Process
  - **Parameters**
    - `props` – ExportProperties is the Properties Object for the Application

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

- **Description**

  This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.

- **Returns** – true if the Kafka Stream Started false otherwise

- **kafkaStreamStart**

```
public boolean kafkaStreamStart()
```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.

  - **Returns** – true if the Kafka Stream Started false otherwise

3.4.5 Class GraphiteProcessStrategy

This Class is for The Processing of the Data for Graphite, to represente the Sensors. It Generates a Graphite Output Stream

Declaration

```
public class GraphiteProcessStrategy
 extends java.lang.Object
```

Constructor summary

GraphiteProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **GraphiteProcessStrategy**

```
public GraphiteProcessStrategy()
```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Graphite Process got Successfully worked

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

  - **Description**

    This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.
  - **Returns** – true if the Kafka Stream closed, false otherwise

- **kafkaStreamStart**

  ```
  public boolean kafkaStreamStart()
  ```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.
  - **Returns** – true if the Kafka Stream Started false otherwise

3.4.6 Class TopologyBuilder

A component that is used to build a ProcessorTopology. A topology contains an acyclic graph of sources, processors, and sinks. A source is a node in the graph that consumes one or more Kafka topics and forwards them to its child nodes. A processor is a node in the graph that receives input records from upstream nodes, processes that records, and optionally forwarding new records to one or all of its children. Finally, a sink is a node in the graph that receives records from upstream nodes and writes them to a Kafka topic. This builder allows you to construct an acyclic graph of these nodes, and the builder is then passed into a new KafkaStreams instance that will then begin consuming, processing, and producing records

Declaration

```
public class TopologyBuilder
 extends java.lang.Object
```

Constructor summary

TopologyBuilder() Default constructor

Method summary

addProcessor(String, StreamProcessingStrategy, String) Add a new processor node that receives and processes records output by one or more parent source or processor node.

addSink(String, String) Add a new sink that forwards records from upstream parent processor and/or source nodes to the named Kafka topic.

addSource(String, topic) Add a new source that consumes from topics matching the given pattern and forward the records to child processor and/or sink nodes.

Constructors

- **TopologyBuilder**

  ```
  public TopologyBuilder()
  ```

  - **Description**

    Default constructor

Methods

- **addProcessor**

  ```
  public void addProcessor(java.lang.String name,
    StreamProcessingStrategy supplier,java.lang.String input)
  ```

  - **Description**

    Add a new processor node that receives and processes records output by one or more parent source or processor node.

  - **Parameters**

    * `name` – is the name of the Processor Stratgie
    * `supplier` – supplier is the supplier of the Process instant to generate more then 1 Process
    * `input` – input Topic Stream name

- **addSink**

  ```
  public void addSink(java.lang.String name,java.lang.String topic)
  ```

  - **Description**

    Add a new sink that forwards records from upstream parent processor and/or source nodes to the named Kafka topic.

  - **Parameters**

    * `name` – name of the Sink
    * `topic` – name of the Topic Stream

- **addSource**

  ```
  public void addSource(java.lang.String name,topic topicPattern)
  ```

  - **Description**

    Add a new source that consumes from topics matching the given pattern and forward the records to child processor and/or sink nodes.

  - **Parameters**

    * `name` – name of the Input Topic Stream
    * `topicPattern` – topicPattern is a Pattern to filter the data from the Input Topic Stream

3.4.7 Class UncaughtExceptionHandler

To catch any unexpected exceptions, you can set before you start the application. This handler is called whenever a stream thread is terminated by an unexpected exception.

Declaration

```
public class UncaughtExceptionHandler
 extends java.lang.Object
```

Constructor summary

UncaughtExceptionHandler() Default constructor

Method summary

getMessage() Returns the detail message string of this throwable.

Constructors

- **UncaughtExceptionHandler**

  `public UncaughtExceptionHandler()`

  - **Description**

    Default constructor

Methods

- **getMessage**

  `public java.lang.String getMessage()`

  - **Description**

    Returns the detail message string of this throwable.
  - **Returns** – String with the error Message

3.5 Package Properties

Package Contents *Page*

Interfaces

PropertiesFileInterface ..... **??**
The Properties Interface is a special form of associative memory in which key-value pairs are always of type string.

Classes

PropertiesFile ..... **??**
The Properties class is a special form of associative memory in which key-value pairs are always of type string.

3.5.1 Interface PropertiesFileInterface

The Properties Interface is a special form of associative memory in which key-value pairs are always of type string. Since the entries can be stored in a file and read out again, hardwired character strings can be externalized from the program text so that the values can be easily changed without retranslation.

Declaration

```
public interface PropertiesFileInterface
```

All known subinterfaces

PropertiesFile

All classes known to implement interface

PropertiesFile

Method summary

getPropValues(String) This Methodes returns the requestet propertie Value
putProperty(String, String) The Method adds a key-value pair to the Properties object.
save(boolean) This Method saves the PropertiesFile with the Option to do a Backup of the File

Methods

- **getPropValues**

  ```
  java.lang.String getPropValues(java.lang.String propertyName)
  ```

  - **Description**

    This Methodes returns the requestet propertie Value

  - **Parameters**

    * `propertyName` – propertyName is the name of the Requested Property

  - **Returns** – Return the Value to the Requested Property

- **putProperty**

  ```
  boolean putProperty(java.lang.String propertyName,java.lang.String propertyValue)
  ```

  - **Description**

    The Method adds a key-value pair to the Properties object. To get back to the value later, is called with the key and then return

  - **Parameters**

    * `propertyName` – propertyName is the Name of the Property which you want to edit
    * `propertyValue` – propertyValue is the Value of the Property which you want to edit

  - **Returns** – true wenn the property got set false otherwise

- **save**

  ```
  boolean save(boolean makeBackup)
  ```

  - **Description**

    This Method saves the PropertiesFile with the Option to do a Backup of the File
  - **Parameters**
    * `makeBackup` – true if you want to make a Bachup
  - **Returns** – true when the file got saved, false otherwise

3.5.2 Class PropertiesFile

The Properties class is a special form of associative memory in which key-value pairs are always of type string. Since the entries can be stored in a file and read out again, hardwired character strings can be externalized from the program text so that the values can be easily changed without retranslation.

Declaration

```
public class PropertiesFile
 extends java.lang.Object implements PropertiesFileInterface
```

Constructor summary

PropertiesFile() Default constructor

Method summary

getPropValues(String) This Methodes returns the requestet propertie Value
putProperty(String, String) The Method adds a key-value pair to the Properties object.
save(boolean) This Method saves the PropertiesFile with the Option to do a Backup of the File

Constructors

- **PropertiesFile**

  ```
  public PropertiesFile()
  ```

  - **Description**

    Default constructor

Methods

- **getPropValues**

  ```
  public java.lang.String getPropValues(java.lang.String propertyName)
  ```

  - **Description**

    This Methodes returns the requestet propertie Value

  - **Parameters**
    - `propertyName` – propertyName is the name of the Requested Property
  - **Returns** – Return the Value to the Requested Property

- **putProperty**

  ```
  public boolean putProperty(java.lang.String propertyName,java.lang.
      String propertyValue)
  ```

  - **Description**

    The Method adds a key-value pair to the Properties object. To get back to the value later, is called with the key and then return

  - **Parameters**
    - `propertyName` – propertyName is the Name of the Property which you want to edit
    - `propertyValue` – propertyValue is the Value of the Property which you want to edit
  - **Returns** – true wenn the property got set false otherwise

- **save**

  ```
  public boolean save(boolean makeBackup)
  ```

  - **Description**

    This Method saves the PropertiesFile with the Option to do a Backup of the File

  - **Parameters**
    - `makeBackup` – true if you want to make a Bachup
  - **Returns** – true when the file got saved, false otherwise

Class Hierarchy

Classes

- java.lang.Object
  - Import.CSVReaderStrategy (in 3.2.2, page 24)
  - Import.DataImporter (in 3.2.3, page 26)
  - Import.FileImporter (in 3.2.4, page 27)
  - Import.FrostSender (in 3.2.5, page 28)
  - Import.NetCDFReaderStrategy (in 3.2.6, page 29)
  - Import.ReaderType (in 3.2.7, page 31)

Interfaces

- Import.FileReaderStrategy (in 3.2.1, page 23)

3.6 Package Import

Package Contents *Page*

Interfaces

Classes

3.6.1 Interface FileReaderStrategy

Interface for the FileReaderStrategy classes. Realization of a Strategy to be able to swap out the way a File has to be read.

Declaration

```
public interface FileReaderStrategy
```

All known subinterfaces

NetCDFReaderStrategy (in 3.2.6, page 29), CSVReaderStrategy (in 3.2.2, page 24)

All classes known to implement interface

NetCDFReaderStrategy (in 3.2.6, page 29), CSVReaderStrategy (in 3.2.2, page 24)

Method summary

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

Methods

- **sendFileData**

  ```
  void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**

    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.2 Class CSVReaderStrategy

Implementation of the FileReaderStrategy interface for CSV files.

Declaration

```
public class CSVReaderStrategy
 extends java.lang.Object implements FileReaderStrategy
```

Constructor summary

CSVReaderStrategy() Default constructor

Method summary

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

Constructors

- **CSVReaderStrategy**

  ```
  public CSVReaderStrategy()
  ```

  - **Description**

    Default constructor

Methods

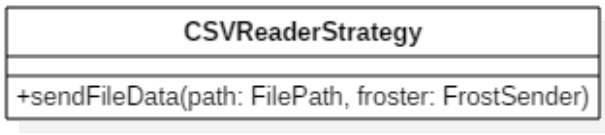
- **sendFileData**

  ```
  public void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**



    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

- **sendFileData**

```
public void sendFileData(FilePath path,FrostSender froster)
```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**
    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.3 Class DataImporter

Importer for data that should be added to PaVoS. Import takes place for files in a specified folder of the server.

Declaration

```
public class DataImporter
 extends java.lang.Object
```

Constructor summary

DataImporter() Default constructor

Method summary

startImportingFileData() Checks for files in the specified import folder and opens a new thread for each of them, where a FileImporter is started to import the contained data.

Constructors

- **DataImporter**

```
public DataImporter()
```

  - **Description**

    Default constructor

Methods

- **startImportingFileData**

```
public void startImportingFileData()
```

  - **Description**

    Checks for files in the specified import folder and opens a new thread for each of them, where a FileImporter is started to import the contained data.

3.6.4 Class FileImporter

Importer for the Data contained in a File. Takes the Data and sends them to the FROST-Server.

Declaration

```
public class FileImporter
 extends java.lang.Object
```

Constructor summary

FileImporter() Default constructor

Method summary

addFileData(FilePath, FrostSender) Adds the Data of a File at a specified FilePath to the FROST-Server.

Constructors

- **FileImporter**

```
public FileImporter()
```

  - **Description**

    Default constructor

Methods

- **addFileData**

```
public void addFileData(FilePath path,FrostSender froster)
```

  - **Description**

    Adds the Data of a File at a specified FilePath to the FROST-Server. To do so, the FileExtension of the File is determined.With help of the readerTypeClass the matching implementation of the FileReaderStrategy interface for the FileExtension is generated and can be used to get the Data from then File.

  - **Parameters**
    - `path` – Is the FilePath of the File to Import.
    - `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.5 Class FrostSender

sends Data to the FROST-Server.

Declaration

```
public class FrostSender
 extends java.lang.Object
```

Constructor summary

FrostSender() Default constructor

Method summary

sendToFrostServer(JsonObject) Sends the given JsonObject to the FROST-Server.

Constructors

- **FrostSender**

  ```
  public FrostSender()
  ```

  - **Description**

    Default constructor

Methods

- **sendToFrostServer**

  ```
  public void sendToFrostServer(JsonObject json)
  ```

  - **Description**

    Sends the given JsonObject to the FROST-Server.
  - **Parameters**
    - `json` – Represents a single ObservedProperty.

3.6.6 Class NetCDFReaderStrategy

Implementation of the FileReaderStrategy interface for NetCDF files.

Declaration

```
public class NetCDFReaderStrategy
 extends java.lang.Object implements FileReaderStrategy
```

Constructor summary

NetCDFReaderStrategy() Default constructor

Method summary

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

Constructors

- **NetCDFReaderStrategy**

  ```
  public NetCDFReaderStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **sendFileData**

  ```
  public void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**

    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

- **sendFileData**

```
public void sendFileData(FilePath path,FrostSender froster)
```

- **Description**

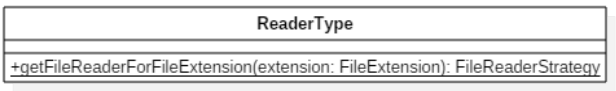
  Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

- **Parameters**

  * `path` – Is the FilePath of the File to Import.
  * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.7 Class ReaderType

Is like a chooser for the right FileReaderStrategy. If a new Strategy is added, this class needs some changes to use the new Strategy.



Declaration

```
public class ReaderType
 extends java.lang.Object
```

Constructor summary

ReaderType() Default constructor

Method summary

getFileReaderForFileExtension(FileExtension) Gives a new Instance of a FileReaderStrategy for the specified FileExtension.

Constructors

- **ReaderType**

```
public ReaderType()
```

- **Description**

  Default constructor

Methods

- **getFileReaderForFileExtension**

```
public static FileReaderStrategy getFileReaderForFileExtension(
    FileExtension extension)
```

  - **Description**

    Gives a new Instance of a FileReaderStrategy for the specified FileExtension.

  - **Parameters**

    * `extension` – is the FileExtension for which a FileReaderStrategy has to be generated.

  - **Returns** – An instance of an implementation of the FileReaderStrategy interface.

Class Hierarchy

Classes

- java.lang.Object
  - DatabaseConnection.ClusterID (in 3.3.1, page 34)
  - DatabaseConnection.Facade (in 3.3.3, page 36)
  - DatabaseConnection.HttpServlet (in 3.3.5, page 39)
    - DatabaseConnection.GridDataServlet (in 3.3.4, page 38)
    - DatabaseConnection.SensorListServlet (in 3.3.9, page 43)
  - DatabaseConnection.KafkaToStorageProcessor (in 3.3.6, page 40)
  - DatabaseConnection.Maintainer (in 3.3.7, page 41)
    - DatabaseConnection.DataMaintainer (in 3.3.2, page 35)
    - DatabaseConnection.SensorMaintainer (in 3.3.10, page 44)
  - DatabaseConnection.MaintenanceManager (in 3.3.8, page 42)
  - DatabaseConnection.ZoomLevel (in 3.3.11, page 45)

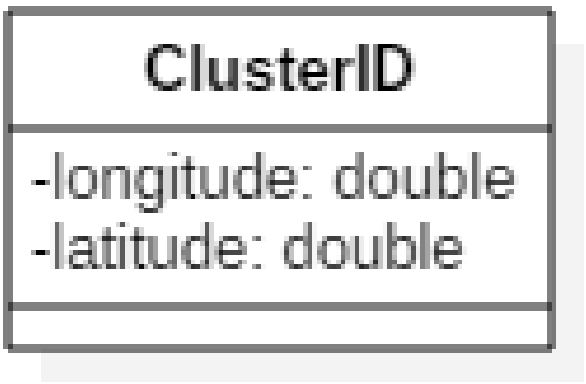
3.7 Package DatabaseConnection

Package Contents *Page*

Classes

3.7.1 Class ClusterID

This class describes a unique identification of a cluster via longitude and latitude.



Declaration

```
public class ClusterID
 extends java.lang.Object
```

Constructor summary

ClusterID() Default constructor

Constructors

- **ClusterID**

  ```
  public ClusterID()
  ```

  – **Description**

  Default constructor

3.7.2 Class DataMaintainer

This class maintains the sensordata in the StorageSolution.

Declaration

```
public class DataMaintainer
 extends DatabaseConnection.Maintainer
```

Constructor summary

DataMaintainer() Default constructor

Method summary

summarize(TimeUnit) This method takes data of a certain TimeUnit and summarizes it into the next higher TimeUnit.

Constructors

- **DataMaintainer**

  ```
  public DataMaintainer()
  ```

  - **Description**

    Default constructor

Methods

- **summarize**

  ```
  public void summarize(TimeUnit timeUnit)
  ```

- **Description**

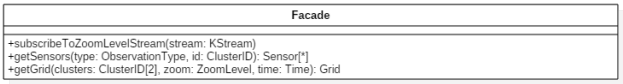
  This method takes data of a certain TimeUnit and summarizes it into the next higher TimeUnit. The summarized data is then saved back into the StorageSolution. The original data of the lower TimeUnit is then deleted from the database.

- **Parameters**

  * `timeUnit` – The TimeUnit to summarize.

3.7.3 Class Facade

A facade to simplify access to a StorageSolution, such as a database. Through the methods, data can be inserted into the StorageSolution and certain information about its content requested.



Declaration

```
public class Facade
 extends java.lang.Object
```

Constructor summary

Facade() Default constructor

Method summary

getGrid(ClusterID[], ZoomLevel, Time) Returns an appropriate grid of clusters in the requested grid section for the specified ZoomLevel and time.

getSensors(ObservationType, ClusterID) Fetches all sensors from the given cluster that observe the given ObservedProperty and returns an array of sensors.

subscribeToZoomLevelStream(KStream) Subscribes to the given KafkaStream, which contains ZoomLevel-specific data and initiates processing of its records.

Constructors

- **Facade**

  ```
  public Facade()
  ```

  - **Description**

    Default constructor

Methods

- **getGrid**

  ```
  public Grid getGrid(ClusterID[] clusters,ZoomLevel zoom,Time time)
  ```

  - **Description**

    Returns an appropriate grid of clusters in the requested grid section for the specified ZoomLevel and time. The (first) two values of the ClusterID array define the grid section from which to get the data.

  - **Parameters**
    - `clusters` – An array of ClusterIDs from which the first two entries are taken to compute the section of the Grid to get the data from.
    - `zoom` – The ZoomLevel from which to get the data.
    - `time` – The point in time.
  - **Returns** – A grid with the computed data.

- **getSensors**

  ```
  public java.util.Set getSensors(ObservationType type,ClusterID id)
  ```

  - **Description**

    Fetches all sensors from the given cluster that observe the given ObservedProperty and returns an array of sensors.

  - **Parameters**
    - `type` – The ObservationType of the requested sensors.
    - `id` – The ID of the cluster.
  - **Returns** – An array of sensors.

- **subscribeToZoomLevelStream**

  ```
  public void subscribeToZoomLevelStream(KStream stream)
  ```

  - **Description**

    Subscribes to the given KafkaStream, which contains ZoomLevel-specific data and initiates processing of its records.

  - **Parameters**
    - `stream` – The stream to subscribe to.

3.7.4 Class GridDataServlet

An HTTPServlet for requesting Grid data.

Declaration

```
public class GridDataServlet
 extends DatabaseConnection.HttpServlet
```

Constructor summary

GridDataServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) This method calls the getGrid method of the Facade to get a Grid of clusters at a certain ZoomLevel and Time .

Constructors

- **GridDataServlet**

  ```
  public GridDataServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    This method calls the getGrid method of the Facade to get a Grid of clusters at a certain ZoomLevel and Time . This saves the Grid into res.

  - **Parameters**

* `req` – An HttpServletRequest object that contains the request the client has made of the servlet.
* `res` – An HttpServletResponse object that contains the response the servlet sends to the client.

Members inherited from class HttpServlet

`DatabaseConnection.HttpServlet` (in 3.3.5, page 39)

- `public void` **doGet**`(HttpServletRequest` **req**`, HttpServletResponse` **res**`)`

3.7.5 Class HttpServlet

An abstract HTTPServlet.

Declaration

```
public class HttpServlet
 extends java.lang.Object
```

All known subclasses

SensorListServlet (in 3.3.9, page 43), GridDataServlet (in 3.3.4, page 38)

Constructor summary

HttpServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Called by the server (via the service method) to allow a servlet to handle a GET request.

Constructors

- **HttpServlet**

```
public HttpServlet()
```

- **Description**

  Default constructor

Methods

- **doGet**

```
public void doGet(HttpServletRequest req,HttpServletResponse res)
```

  - **Description**

    Called by the server (via the service method) to allow a servlet to handle a GET request.

  - **Parameters**

    * `req` – An HttpServletRequest object that contains the request the client has made of the servlet.
    * `res` – An HttpServletResponse object that contains the response the servlet sends to the client.

3.7.6 Class KafkaToStorageProcessor

This class converts KafkaStream records to data that can be inserted into the StorageSolution.

Declaration

```
public class KafkaToStorageProcessor
 extends java.lang.Object
```

Constructor summary

KafkaToStorageProcessor() Default constructor

Method summary

subscribe(KStream) Subscribes to the given KafkaStream and converts the data to the appropriate format for the StorageSolution.

Constructors

- **KafkaToStorageProcessor**

  ```
  public KafkaToStorageProcessor()
  ```

  - **Description**

    Default constructor

Methods

- **subscribe**

  ```
  public void subscribe(KStream stream)
  ```

  - **Description**

    Subscribes to the given KafkaStream and converts the data to the appropriate format for the StorageSolution. If a stream is already subscribed to, unsubscribes from the old stream and subscribes to the new one.

  - **Parameters**

    * `stream` – The KStream to subscribe to.

3.7.7 Class Maintainer

An abstract class describing a Maintainer, which performs maintenance on certain data in the StorageSolution.

Declaration

```
public class Maintainer
 extends java.lang.Object
```

All known subclasses

SensorMaintainer (in 3.3.10, page 44), DataMaintainer (in 3.3.2, page 35)

Constructor summary

Maintainer() Default constructor

Constructors

- **Maintainer**

```
public Maintainer()
```

  - **Description**

    Default constructor

3.7.8 Class MaintenanceManager

This class manages the way the methods of Maintainers are called to make sure the StorageSolution content is maintained.

Declaration

```
public class MaintenanceManager
 extends java.lang.Object
```

Constructor summary

MaintenanceManager() Default constructor

Method summary

startMaintenance() This method should be called as soon as the database is started.

Constructors

- **MaintenanceManager**

```
public MaintenanceManager()
```

  - **Description**

    Default constructor

Methods

- **startMaintenance**

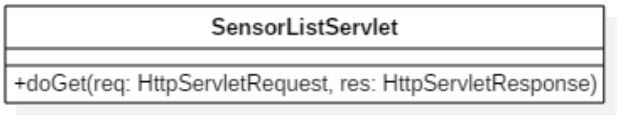
```
public void startMaintenance()
```

  - **Description**

    This method should be called as soon as the database is started. Through calls to instances of Maintainers, summarizes data in the database and deletes data that has become obsolete as a result of the summarization.

3.7.9 Class SensorListServlet

An HTTPServlet for requesting a list of sensors.



Declaration

```
public class SensorListServlet
 extends DatabaseConnection.HttpServlet
```

Constructor summary

SensorListServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) This method calls the getSensors method of the Facade to get a list of Sensors that are in a certain cluster.

Constructors

- **SensorListServlet**

  ```
  public SensorListServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req, HttpServletResponse res)
  ```

  - **Description**

    This method calls the getSensors method of the Facade to get a list of Sensors that are in a certain cluster.

  - **Parameters**

    * `req` – An HttpServletRequest object that contains the request the client has made of the servlet.
    * `res` – An HttpServletResponse object that contains the response the servlet sends to the client.

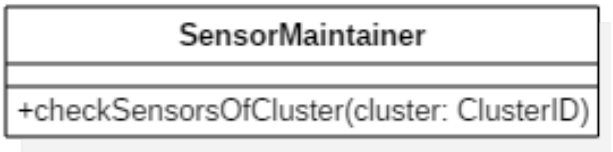
Members inherited from class HttpServlet

`DatabaseConnection.HttpServlet` (in 3.3.5, page 39)

- `public void` **doGet**`(HttpServletRequest req, HttpServletResponse res)`

3.7.10 Class SensorMaintainer

This class maintains the list of sensors saved in the StorageSolution.



Declaration

```
public class SensorMaintainer
 extends DatabaseConnection.Maintainer
```

Constructor summary

SensorMaintainer() Default constructor

Method summary

checkSensorsOfCluster(ClusterID) This method checks if the sensors registered to the given cluster are up to date.

Constructors

- **SensorMaintainer**

  ```
  public SensorMaintainer()
  ```

  - **Description**

    Default constructor

Methods

- **checkSensorsOfCluster**

  ```
  public void checkSensorsOfCluster(ClusterID cluster)
  ```

  - **Description**

    This method checks if the sensors registered to the given cluster are up to date. A sensor is up to date if data has been received from it in the last 24 hours. If this requirement is not met, the sensor is deleted from the database.

  - **Parameters**

    * `cluster` – The cluster to check.

3.7.11 Class ZoomLevel

This class describes a zoom level for the map.

Declaration

```
public class ZoomLevel
 extends java.lang.Object
```

Constructor summary

ZoomLevel() Default constructor

Constructors

- **ZoomLevel**

  ```
  public ZoomLevel()
  ```

  - **Description**

    Default constructor

Class Hierarchy

Classes

- java.lang.Object
  - Download.DownloadID (in 3.5.2, page 64)
  - Download.DownloadState (in 3.5.3, page 65)
    - Download.AlterableDownloadState (in 3.5.1, page 62)
  - Export.AbstractExporter (in 3.4.2, page 49)
    - Export.FileExporter (in 3.4.6, page 55)
  - Export.CSVWriterStrategy (in 3.4.3, page 50)
  - Export.ExportProperties (in 3.4.4, page 52)
  - Export.ExportStreamGenerator (in 3.4.5, page 54)
  - Export.FileExtension (in 3.4.7, page 56)
  - Export.FileType (in 3.4.8, page 57)
  - Export.FileTypesUtility (in 3.4.9, page 59)
  - Export.NetCDFWriterStrategy (in 3.4.10, page 60)
  - ExportDownloadCommunication.HttpServlet (in 3.6.4, page 70)
    - ExportDownloadCommunication.DownloadServlet (in 3.6.1, page 66)
    - ExportDownloadCommunication.ExportServlet (in 3.6.2, page 68)
    - ExportDownloadCommunication.FileExtensionServlet (in 3.6.3, page 69)
    - ExportDownloadCommunication.StatusServlet (in 3.6.5, page 71)

Interfaces

- Export.FileWriterStrategy (in 3.4.1, page 48)

3.8 Package Export

Package Contents *Page*

Interfaces

Classes

Generates a Stream for the Export by asking for one at the PaVoS Core and Subscribing to it.

FileExporter 55
Exporter of Data from Kafka to a File.

FileExtension 56
Represents the FileExtension of a File.

FileType 57
Is used to store a FileExtension information and give the right FileWriter for this FileExtension.

FileTypesUtility 59
Utility class that provides static methods to get all supported FileExtensions and one to get a new Instance of the FileWriter associated with a given FileExtension.

NetCDFWriterStrategy 60
Implementation of the FileWriterStrategy interface for NetCDF files.

3.8.1 Interface FileWriterStrategy

Interface for the FileWriterStrategy classes. Realization of a Strategy to be able to swap out the way a File has to be saved.

Declaration

`public interface FileWriterStrategy`

All known subinterfaces

NetCDFWriterStrategy (in 3.4.10, page 60), CSVWriterStrategy (in 3.4.3, page 50)

All classes known to implement interface

NetCDFWriterStrategy (in 3.4.10, page 60), CSVWriterStrategy (in 3.4.3, page 50)

Method summary

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

Methods

- **saveToFile**

  ```
  void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.

  - **Parameters**

    * `stream` – is the KStream, that should be exported to a File.
    * `path` – Is the FilePath, where the new File should be created.

3.8.2 Class AbstractExporter

Abstract Exporter of Data to a File.

Declaration

```
public class AbstractExporter
 extends java.lang.Object
```

All known subclasses

FileExporter (in 3.4.6, page 55)

Field summary

properties Contains the Properties of an Export Request.

Constructor summary

AbstractExporter() Default constructor

Method summary

createFile() Generates the File with the desired Data.
createFileInformation() Creates Information for that Export.

Fields

- `public ExportProperties` **properties**
  - Contains the Properties of an Export Request.

Constructors

- **AbstractExporter**

  ```
  public AbstractExporter()
  ```

  - **Description**

    Default constructor

Methods

- **createFile**

  ```
  public void createFile()
  ```

  - **Description**

    Generates the File with the desired Data.

- **createFileInformation**

  ```
  public DownloadID createFileInformation()
  ```

  - **Description**

    Creates Information for that Export. These Information will be used to identifie a File for the WebGUI, that gets the created DownloadID.
  - **Returns** – Is the DownloadID for the started Export.

3.8.3 Class CSVWriterStrategy

Implementation of the FileWriterStrategy interface for CSV files.

Declaration

```
public class CSVWriterStrategy
 extends java.lang.Object implements FileWriterStrategy
```

Constructor summary

CSVWriterStrategy() Default constructor

Method summary

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.
saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

Constructors

- **CSVWriterStrategy**

  ```
  public CSVWriterStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **saveToFile**

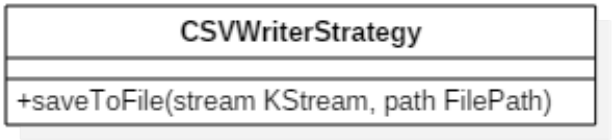
  ```
  public void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

  - **Parameters**

    * `stream` – is the KStream, that should be exported to a File.
    * `path` – Is the FilePath, where the new File should be created.



- **saveToFile**

  ```
  public void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

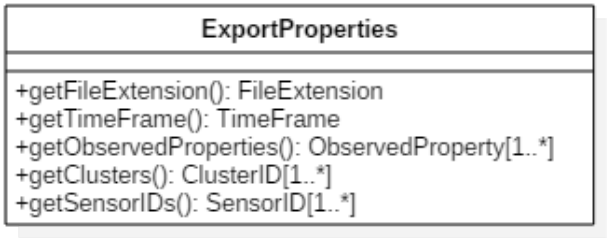
    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.

  - **Parameters**

    * `stream` – is the KStream, that should be exported to a File.
    * `path` – Is the FilePath, where the new File should be created.

3.8.4 Class ExportProperties

Contains the Properties of an Export Request.



Declaration

```
public class ExportProperties
 extends java.lang.Object
```

Constructor summary

ExportProperties() Default constructor

Method summary

getClusters() Get the ClusterIDs that should be exported.
getFileExtension() Get the FileExtension for the Export File.
getObservedProperties() Get the ObservedProperties that should be exported.
getSensorIDs() Get the SensorIDs that should be exported.
getTimeFrame() Get the TimeFrame of the Data that should be exported.

Constructors

- **ExportProperties**

  ```
  public ExportProperties()
  ```

  - **Description**

    Default constructor

Methods

- **getClusters**

  ```
  public java.util.Set getClusters()
  ```

  - **Description**

    Get the ClusterIDs that should be exported. Always only exports a Groupd of Sensors or a Group of Clusters. The other Option is Empty.
  - **Returns** – The Clusters that should be taken in the Export.

- **getFileExtension**

  ```
  public FileExtension getFileExtension()
  ```

  - **Description**

    Get the FileExtension for the Export File.
  - **Returns** – The FileExtension for the File to export.

- **getObservedProperties**

  ```
  public java.util.Set getObservedProperties()
  ```

  - **Description**

    Get the ObsorvedProperties that should be exported.
  - **Returns** – The ObservedProperties that should be used for the export.

- **getSensorIDs**

  ```
  public java.util.Set getSensorIDs()
  ```

- **Description**

  Get the SensorIDs that should be exported. Always only exports a Groupd of Sensors or a Group of Clusters. The other Option is Empty.

- **Returns** – The SensorIDs of the Data that should be exported.

- **getTimeFrame**

```
public TimeFrame getTimeFrame()
```

  - **Description**

    Get the TimeFrame of the Data that should be exported.

  - **Returns** – The TimeFrame of the Data to be exported.

3.8.5 Class ExportStreamGenerator

Generates a Stream for the Export by asking for one at the PaVoS Core and Subscribing to it.

Declaration

```
public class ExportStreamGenerator
 extends java.lang.Object
```

Field summary

properties Contains the Properties of an Export Request.

Constructor summary

ExportStreamGenerator() Default constructor

Method summary

createExportStream() Asks for a KafkaStream and subscribes to it.

Fields

- `public ExportProperties` **properties**
  - Contains the Properties of an Export Request.

Constructors

- **ExportStreamGenerator**

  ```
  public ExportStreamGenerator()
  ```

  - **Description**

    Default constructor

Methods

- **createExportStream**

  ```
  public KStream createExportStream()
  ```

  - **Description**

    Asks for a KafkaStream and subscribes to it. Then gives it through to the needed part for the export.
  - **Returns** – Is a KStream of the Data that should be exported.

3.8.6 Class FileExporter

Exporter of Data from Kafka to a File.

Declaration

```
public class FileExporter
 extends Export.AbstractExporter
```

Constructor summary

FileExporter() Default constructor

Method summary

createFile() Generates the File with the desired Data.
createFileInformation() Creates Information for that Export.

Constructors

- **FileExporter**

  ```
  public FileExporter()
  ```

  - **Description**

    Default constructor

Methods

- **createFile**

  ```
  public void createFile()
  ```

  - **Description**

    Generates the File with the desired Data.

- **createFileInformation**

  ```
  public DownloadID createFileInformation()
  ```

  - **Description**

    Creates Information for that Export. These Information will be used to identifie a File for the WebGUI, that gets the created DownloadID.
  - **Returns** – Is the DownloadID for the started Export.

Members inherited from class AbstractExporter

`Export.AbstractExporter` (in 3.4.2, page 49)

- `public void` **createFile()**
- `public DownloadID` **createFileInformation()**
- `public` **properties**

3.8.7 Class FileExtension

Represents the FileExtension of a File. Is used to match the right FileFormat for an export or import.

Declaration

```
public class FileExtension
 extends java.lang.Object
```

Constructor summary

FileExtension() Default constructor

Constructors

- **FileExtension**

  ```
  public FileExtension()
  ```

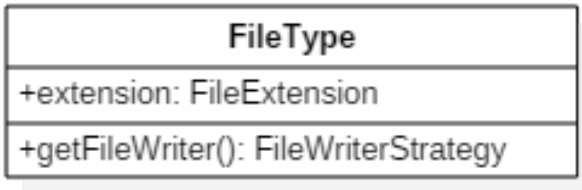
  - **Description**

    Default constructor

3.8.8 Class FileType

Is used to store a FileExtension information and give the right FileWriter for this FileExtension.

Declaration



```
public class FileType
 extends java.lang.Object
```

Field summary

extension The FileExtension is defining the FileType.

Constructor summary

FileType() Default constructor

Method summary

getFileWriter() Gives an instance of the implemented FileWriter that is associated with this FileType, thus this FileExtension.

Fields

- `public FileExtension` **extension**
  - The FileExtension is defining the FileType.

Constructors

- **FileType**

  ```
  public FileType()
  ```

  - **Description**

    Default constructor

Methods

- **getFileWriter**

  ```
  public FileWriterStrategy getFileWriter()
  ```

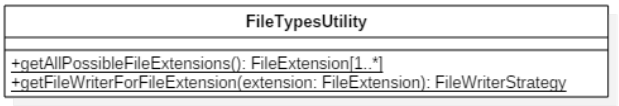
  - **Description**

    Gives an instance of the implemented FileWriter that is associated with this FileType, thus this FileExtension. To do so it uses the static method getFileWriterForFileExtension from the FileTypesUtility class.

  - **Returns** – Is a new instance of an implementation of a FilWriterStrategy.

3.8.9 Class FileTypesUtility

Utility class that provides static methods to get all supported FileExtensions and one to get a new Instance of the FileWriter associated with a given FileExtension. If a new FileWriter is added to PaVoS, this class needs some changed to be able to return the new FileWriter.



Declaration

```
public class FileTypesUtility
 extends java.lang.Object
```

Constructor summary

FileTypesUtility() Default constructor

Method summary

getAllPossibleFileExtensions() Gives all supported FileExtensions in an ArrayList.
getFileWriterForFileExtension(FileExtension) Gives a new Instance of the FileWriter associated witha given FileExtension.

Constructors

- **FileTypesUtility**

  ```
  public FileTypesUtility()
  ```

  - **Description**

    Default constructor

Methods

- **getAllPossibleFileExtensions**

  ```
  public static java.util.Set getAllPossibleFileExtensions()
  ```

  - **Description**

    Gives all supported FileExtensions in an ArrayList.

  - **Returns** – Is an Array of the possible FileExtensions for an Export.

- **getFileWriterForFileExtension**

```
public static FileWriterStrategy getFileWriterForFileExtension(
    FileExtension extension)
```

  - **Description**

    Gives a new Instance of the FileWriter associated witha given FileExtension.

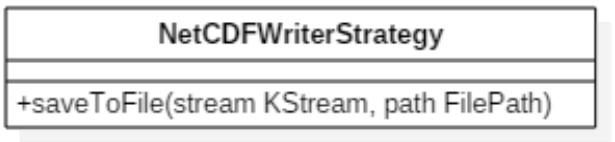
  - **Parameters**

    * `extension` – Is the FileExtension for which a new instance of an Implementation of the FileWriterStrategy is wanted.

  - **Returns** – Is the instance of the implementation of a FileWriterStrategy.

3.8.10 Class NetCDFWriterStrategy

Implementation of the FileWriterStrategy interface for NetCDF files.



Declaration

```
public class NetCDFWriterStrategy
 extends java.lang.Object implements FileWriterStrategy
```

Constructor summary

NetCDFWriterStrategy() Default constructor

Method summary

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

Constructors

- **NetCDFWriterStrategy**

  ```
  public NetCDFWriterStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **saveToFile**

  ```
  public void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.
  - **Parameters**
    - `stream` – is the KStream, that should be exported to a File.
    - `path` – Is the FilePath, where the new File should be created.

- **saveToFile**

  ```
  public void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.
  - **Parameters**
    - `stream` – is the KStream, that should be exported to a File.
    - `path` – Is the FilePath, where the new File should be created.

3.9 Package Download

Package Contents *Page*

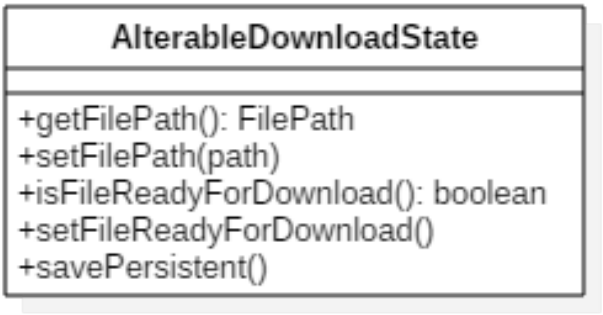
Classes

Is an Identifier for a specific Download, so that the right file can be fount for a requeststed Download.

DownloadState 65
Verifies for the State of a Download.

3.9.1 Class AlterableDownloadState

Verifies for the State of a Download. Can also change it.



Declaration

```
public class AlterableDownloadState
 extends Download.DownloadState
```

Constructor summary

AlterableDownloadState() Default constructor

Method summary

getFilePath() Gives the FilePath associated with this DownloadID.
isFileReadyForDownload() Checks if a File is Ready to be downloaded.
savePersistent() Save the changed Data persistently.
setFilePath(void) Defines the FilePath for the DownloadID.
setFileReadyForDownload() Validate, that the File is ready to be downloaded.

Constructors

- **AlterableDownloadState**

```
public AlterableDownloadState()
```

- **Description**
  Default constructor

Methods

- **getFilePath**

  ```
  public FilePath getFilePath()
  ```

  - **Description**
    Gives the FilePath associated with this DownloadID.
  - **Returns** – The FilePath of the File for the Download.

- **isFileReadyForDownload**

  ```
  public boolean isFileReadyForDownload()
  ```

  - **Description**
    Checks if a File is Ready to be downloaded.
  - **Returns** – A boolean whether the file is downloadable or not.

- **savePersistent**

  ```
  public void savePersistent()
  ```

  - **Description**
    Save the changed Data persistently.

- **setFilePath**

  ```
  public void setFilePath(void path)
  ```

  - **Description**
    Defines the FilePath for the DownloadID.
  - **Parameters**
    * `path` – Is the FilePath to be set.

- **setFileReadyForDownload**

  ```
  public void setFileReadyForDownload()
  ```

  - **Description**
    Validate, that the File is ready to be downloaded.

Members inherited from class DownloadState

`Download.DownloadState` (in 3.5.3, page 65)

- `public` **downloadID**
- `public FilePath` **getFilePath()**
- `public boolean` **isFileReadyForDownload()**

3.9.2 Class DownloadID

Is an Identifier for a specific Download, so that the right file can be fount for a requeststed Download.

Declaration

```
public class DownloadID
 extends java.lang.Object
```

Constructor summary

DownloadID() Default constructor

Constructors

- **DownloadID**

  ```
  public DownloadID()
  ```

  – **Description**

  Default constructor

3.9.3 Class DownloadState

Verifies for the State of a Download.

Declaration

```
public class DownloadState
 extends java.lang.Object
```

All known subclasses

AlterableDownloadState (in 3.5.1, page 62)

Field summary

downloadID Is an Identifier for a specific Download.

Constructor summary

DownloadState() Default constructor

Method summary

getFilePath() Gives the FilePath associated with this DownloadID.
isFileReadyForDownload() Checks if a File is Ready to be downloaded.

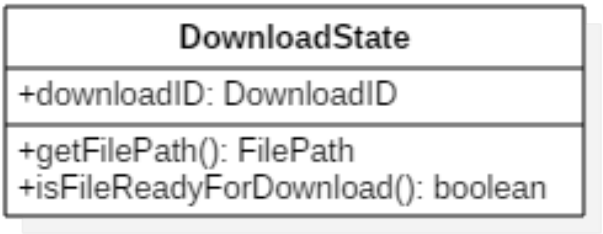
Fields

- `public DownloadID` **downloadID**
  - Is an Identifier for a specific Download.

Constructors

- **DownloadState**

```
public DownloadState()
```



– **Description**
Default constructor

Methods

- **getFilePath**

```
public FilePath getFilePath()
```

  – **Description**
  Gives the FilePath associated with this DownloadID.
  – **Returns** – The FilePath of the File for the Download.

- **isFileReadyForDownload**

```
public boolean isFileReadyForDownload()
```

  – **Description**
  Checks if a File is Ready to be downloaded.
  – **Returns** – A boolean whether the file is downloadable or not.

3.10 Package ExportDownloadCommunication

Package Contents *Page*

Classes

3.10.1 Class DownloadServlet

Servlet to let the WebGUI download a finished Export.

Declaration

```
public class DownloadServlet
 extends ExportDownloadCommunication.HttpServlet
```

Field summary

downloadID Is an Identifier for a specific Download.

Constructor summary

DownloadServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by sending the desired File to the WebGUI.

Fields

- `public DownloadID` **downloadID**
  - Is an Identifier for a specific Download.

Constructors

- **DownloadServlet**

  ```
  public DownloadServlet()
  ```

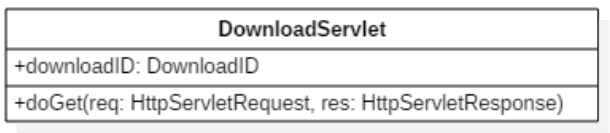
  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```



- **Description**

  Handles a GET request by sending the desired File to the WebGUI.

- **Parameters**
  - `req` – Is the HttpServletRequest.
  - `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.6.4, page 70)

- `public void` **doGet**`(HttpServletRequest` **req**`, HttpServletResponse` **res**`)`

3.10.2 Class ExportServlet

HttpServlet to get a Dataexport request from the WebGUI.

Declaration

```
public class ExportServlet
 extends ExportDownloadCommunication.HttpServlet
```

Field summary

properties Contains the Properties of an Export Request.

Constructor summary

ExportServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by starting the export of the desired Data.

Fields

- `public ExportProperties` **properties**
  - Contains the Properties of an Export Request.

Constructors

- **ExportServlet**

  ```
  public ExportServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    Handles a GET request by starting the export of the desired Data. At the same time a DownloadID is sent back to the WebGUI, so that it can check for the File.

  - **Parameters**

    * `req` – Is the HttpServletRequest.
    * `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.6.4, page 70)

- `public void` **doGet**`(HttpServletRequest` **req,** `HttpServletResponse` **res)**

3.10.3 Class FileExtensionServlet

Servlet, to let the WebGUI ask for the available FileExtensions for the Export.

Declaration

```
public class FileExtensionServlet
 extends ExportDownloadCommunication.HttpServlet
```

Constructor summary

FileExtensionServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by sending Information about the available FileExtensions.

Constructors

- **FileExtensionServlet**

  ```
  public FileExtensionServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    Handles a GET request by sending Information about the available FileExtensions.
  - **Parameters**
    - `req` – Is the HttpServletRequest.
    - `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.6.4, page 70)

- `public void` **doGet**`(HttpServletRequest req, HttpServletResponse res)`

3.10.4 Class HttpServlet

Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. (javax.servlet.http.HttpServlet)

Declaration

```
public class HttpServlet
 extends java.lang.Object
```

All known subclasses

StatusServlet (in 3.6.5, page 71), FileExtensionServlet (in 3.6.3, page 69), ExportServlet (in 3.6.2, page 68), DownloadServlet (in 3.6.1, page 66)

Constructor summary

HttpServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Called by the server (via the service method) to allow a servlet to handle a GET request.

Constructors

- **HttpServlet**

  ```
  public HttpServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    Called by the server (via the service method) to allow a servlet to handle a GET request.
  - **Parameters**
    - `req` – Is the HttpServletRequest.
    - `res` – Is the HttpServletResponse.

3.10.5 Class StatusServlet

Servlet to let the WebGUI check if a Download is ready.

Declaration

```
public class StatusServlet
 extends ExportDownloadCommunication.HttpServlet
```

Field summary

downloadID Is an Identifier for a specific Download.

Constructor summary

StatusServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by checking the availability of the desired download.

Fields

- `public DownloadID` **downloadID**
  - Is an Identifier for a specific Download.

Constructors

- **StatusServlet**

  ```
  public StatusServlet()
  ```

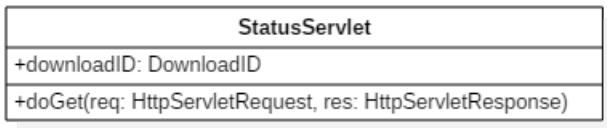
  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```



- **Description**

  Handles a GET request by checking the availability of the desired download.

- **Parameters**

  * `req` – Is the HttpServletRequest.
  * `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.6.4, page 70)

- `public void` **doGet**`(HttpServletRequest` **req**`, HttpServletResponse` **res**`)`