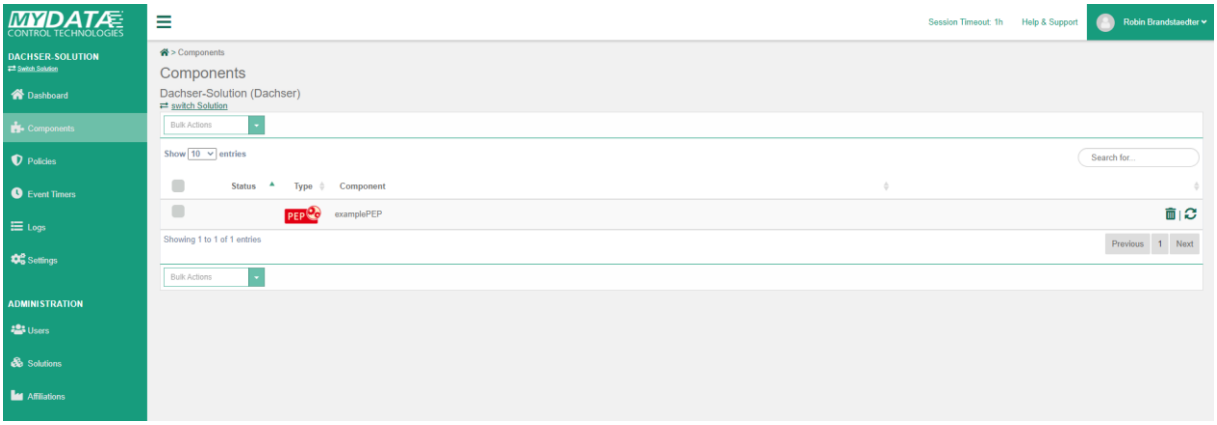
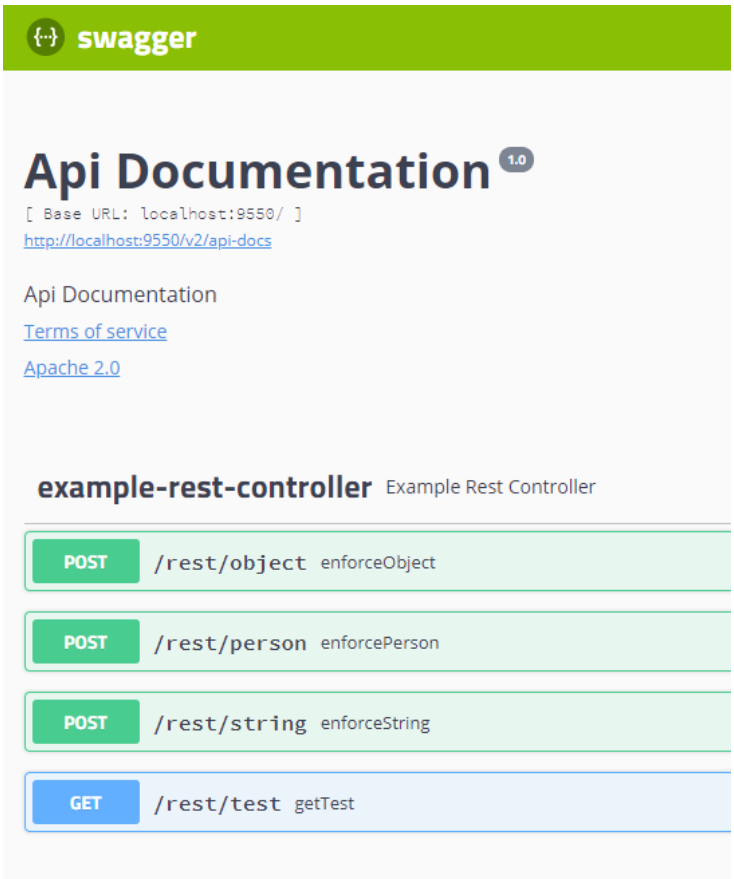
We have implemented a small spring-boot application for you as an example how our technology works, it is attached as example-mydata.zip. You can unzip it at import it as a maven project in your Java IDE, like eclipse or IDEA.
Your Java IDE should support Spring-Boot, Maven and Lombok (https://projectlombok.org/).

Just run the application and it will register a PEP at MYDATA https://administration.mydata-control.de/components. If you see this, it worked.



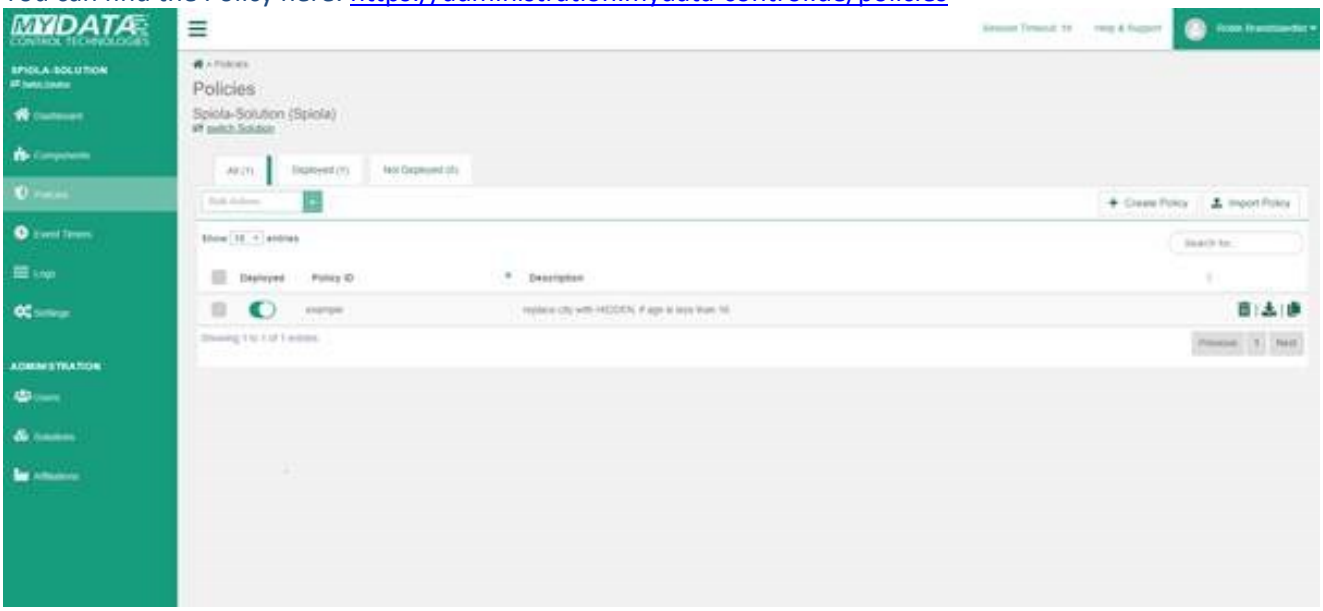
At http://localhost:9550/swagger-ui/ you can easily access the API of this example application. (localhost could be also your external IP or DNS, if reachable)



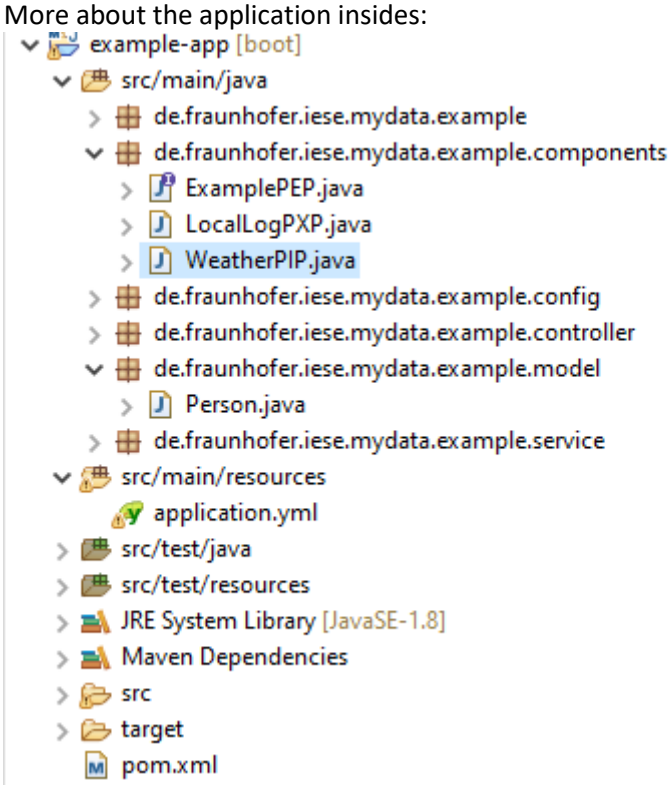
Let us have look at the getPerson Endpoint. To this Endpoint you can send person data (age, city, name, surname) to the application. Inside the application, the data is sent to MYDATA and it checks, if there is a policy for this event, the policy is enforced and the enforced data is returned. See the following screenshots

We also added a policy for you, where the city value is replaced with “HIDDEN”, if age is less than 16 of the person object.

You can find the Policy here: https://administration.mydata-control.de/policies



More about the application insides:



The important Files:

ExamplePEP this is the interface with the event definitions to enforce the data, it is recognized because we added `@EnablePolicyEnforcementPoint(basePackages = "de.fraunhofer.iese")` at the ExampleApplication

`ExampleRestController enables the endpoints and calls the ExampleService methods`

ExampleService is calling the ExamplePEP inside to enforce the policies at the data. E.g. ..

```
  /**
   * Method call with ExamplePep enforcement for a Person object
   *
   * @param person
   * @return returns the enforcedPerson
   */
  public Person enforceMydataPerson(Person person) {
    final Event event =
this.examplePep.enforcePerson(person).toBlocking().first();
    return event.getParameterValue("person", person.getClass());
  }
```

Person.java is just the data model for a person. If you add or remove attributes and restart the application, MYDATA automatically recognizes the changes and you can use it, e.g. inside the policy editors at jsonPathQueries. (Click on it ☺)

Application.yml is the configuration file, it defines the access to MYDATA and vice versa to your application (instead of localhost you can add a public reachable domain).
If you want to have PIPs or PXPs, MYDATA needs access to these endpoints.

To have access to MYDATA, we created a client token (and secret) for your solution, which is placed into the application.yml.

Now you can start playing around with the code and the application, to get familiar with mydata. If you have any further questions, please contact us.

PIP and PXP

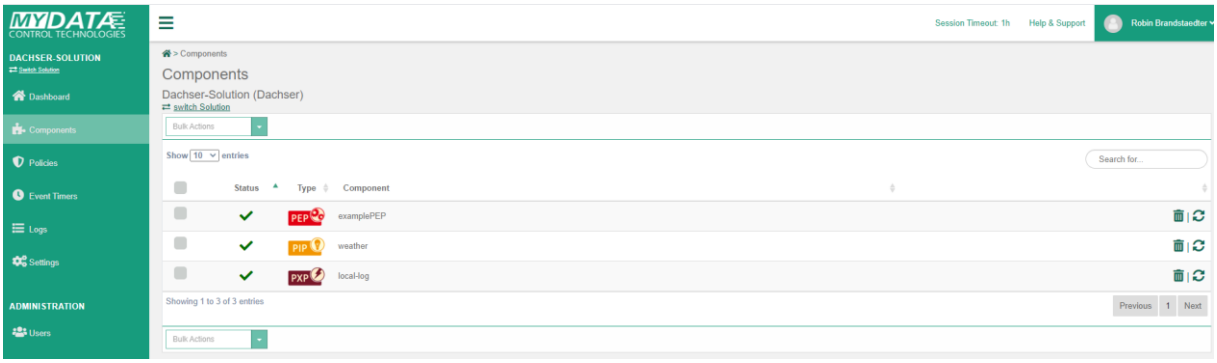
To have a working PIP and PXP the application, where the PIP and PXP are running, has to be reachable from outside – other wise it will not work.

The Application has to tell MYDATA via the application.yml where the components can be reached – marked bold:

```
server:
  port: [your-application-port-number]

mydata:
  pmp:
    cloud:
      url: https://management.mydata-control.de
      client-id: urn:client:[your-solution-id]:[your-token-name]
client-secret: [your-secret]
  solution: [your-solution-id]
  external-server-url: http(s)://[your-IP-or-DNS]:[your-public-port-number]
  operational-mode: cloud
```

At the management console at components you can see the Status of your components, only if PIP and PXP are available the mechanism works.



In this example we have implemented a WeatherPIP.java which returns “nice” for input strings which have an even number of characters. Also, an PXP which is logging the given message as string locally, see LocalLogPXP.java.

Policy with PXP and PIP

The Policy named example-weatherPIP-localLogPXP is for using the PIP and PXP, if the components are not reachable or producing errors, the default values set inside the Policy are used.

In general, this policy acts like the example policy, but add some additional conditions.

If the persons age is older or equals to 16, it checks if the weather is nice at the persons city. If it is nice the usage is allowed, if not the usage is inhibited and the message is logged via the PXP local log action.

Attention: if it is inhibited, an Inhibit Exceptions is thrown, you can handle it inside the app like you want. The current Implementation returns an HTTP-Status forbidden - 403.