Service Integration: How to build a Python AI Service

IIP-Ecosphere Platform
Table of Contents

- Prerequisites
- Building an Application
Prerequisites

• Required:
  • Installed the platform and its dependencies or the development container
  • Installed the IDE for IIP-Ecosphere Platform (provided Eclipse Version)
  • How to configure datatypes
  • How to configure services
  • How to configure an application
  • How to build an application
  • How to test the application

• Optional:
  • Introduction to code generation
Table of Contents

• Prerequisites
• Building an Application
The configuration of datatypes is the same as for every other service.
Add your needed datatypes to the "AllTypesPart....ivml".
Configure the Services (1)

- Define the services in the `AllServicesPart....ivml` file:
- Only highlight the changes to a java service

```
Service pyth = PythonService {
    id = "PyService",
    name = "PyService",
    description = "",
    ver = "0.1.0",
    deployable = true,
    traceRcv = TraceKind::SYSOUT,
    traceSent = TraceKind::SYSOUT,
    input = {{type=refBy(InData)}},
    output = {{type=refBy(OutData)}},
    artifact = "de.iip-ecosphere.platform.apps:TestTestAppServices:" + iipVer,
    kind = ServiceKind::TRANSFORMATION SERVICE,
    dependencies = {refBy(PYTHON39)}
};
```

- **Servicetype**: PythonService
- **class**: left out as it is not needed for a python service
- **kind**: The type of service: Python can only be used as a TRANSFORMATION_SERVICE which is why this value can be left out
- **dependencies**: the python version needed for this service
- **asynchronous**: as this is no sink it will be true if not set otherwise
Configure the Services (2)

- Define the services in the `AllServicesPart....ivml` file:
  - We also add further dependencies of the python service

```python
PythonDependency sklearn0232 = {
    name = "scikit-learn",
    version = "0.23.2"
};

PythonDependency numpy1201 = {
    name = "numpy",
    version = "1.20.1"
};

PythonDependency pickle40 = {
    name = "pickle",
    version = "4.0"
};

PythonDependency pyflakes250 = {
    name = "pyflakes",
    version = "2.5.0"
};
```

**PythonDependency**: The defined type

- **name**: the name of the package as it is installed i.e. “pip install numpy==1.20.1”
- **version**: The specific version that is needed for the service
To add functionality to your python service you need to first generate the templates
Run "mvn -U generate-sources"

- Rename the "impl.model" as needed/desired
- Use cmd to run "mvn -U generate-sources" in the "impl.model" directory
Adding Resources

- If you want to use a AI solution with a pre trained model you need to add the model in the same directory as the python service.
- If you are working with a dev. container you first need to copy the model into the container using "docker cp"
Adding Functionality

- Here we have an example for an AI python service with a pre-trained model.
- When opening the resource add "services/" in front of the file name, as the files will be rearranged when building the application from it.
- The method to edit is "process<InputData Name>".
- To pass on data utilise "self.ingest()".

```python
def processNewInput(self, data: NewInput):
    """Asynchronous data processing method. Use self.ingest(data) to pass the result back to the data store.""
    result = [1, 2]
    if (numplyExists):  
        if (self.clf == None):
            with open("services/trained_forest.pkl", "rb") as p:
                self.clf = pickle.load(p)
    result = [1, 2]
    if (numplyExists):
        if (self.clf != None):
            result = self.clf.predict(data)
    result = None
    result = self.ingest(result)
```
Finishing the Python Service

• After adding functionality to your other services and testing everything, finish building the application as mentioned in “How to Build an Application”
Summary

• What we learned
  • How to build python services
  • How to add resources like models to the application

• How to go on
  • None