FaaS

Java code transformation

Dorodko S. 08.12.16

Why FaaS?

1. Microservices is a good approach
2. Developer can focus at code writing without environment maintenance issues
3. Flexible billing system (ms)
4. Auto Scalability
5. Event-driven invocation

Service Tooling initiative

In this scope the main aim of service tooling initiative is to speed up the moving of big companies into the cloud. Automatic translation and uploading allows to save time and not to rewrite whole project to get into the cloud.

Within the initiative were implemented supporting of 3 languages:

- Java (in the demo)
- Python
- Javascript

Goals

FaaS characteristics analysis (Some of the analysis were described in this post).

Automated translation of code into FaaS.

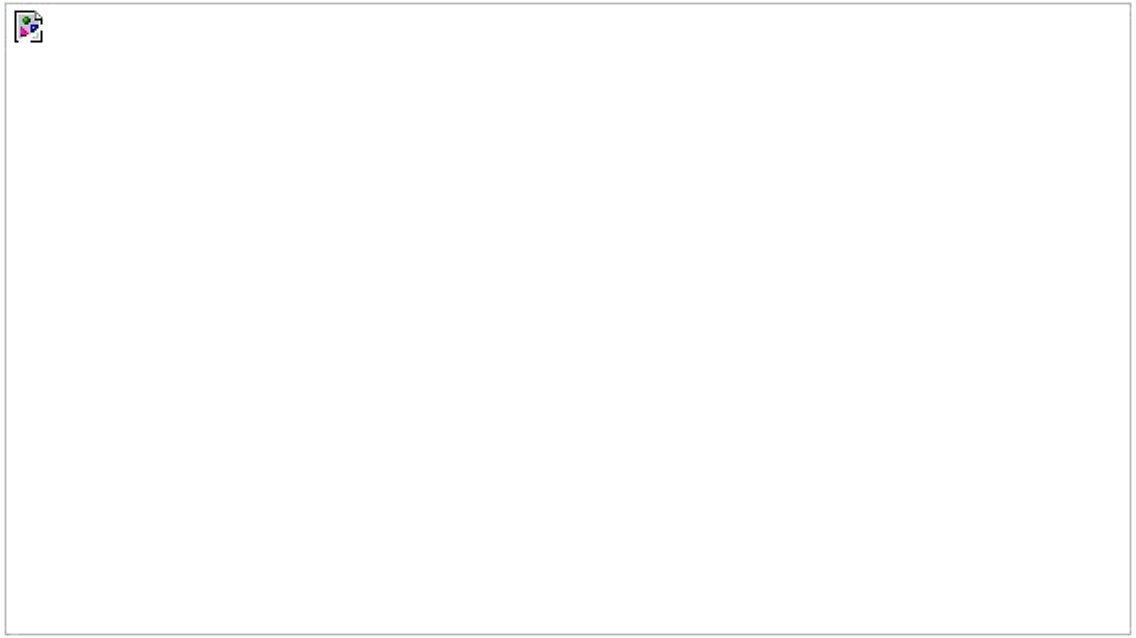
Implementation issues

Functions for deployment have certain programming model

FaaS principles dictate to be stateless

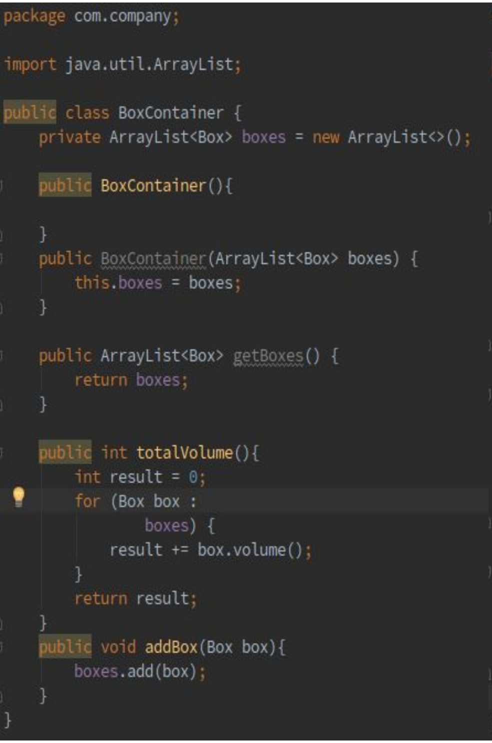
Translator architecture description

Transformed project architecture

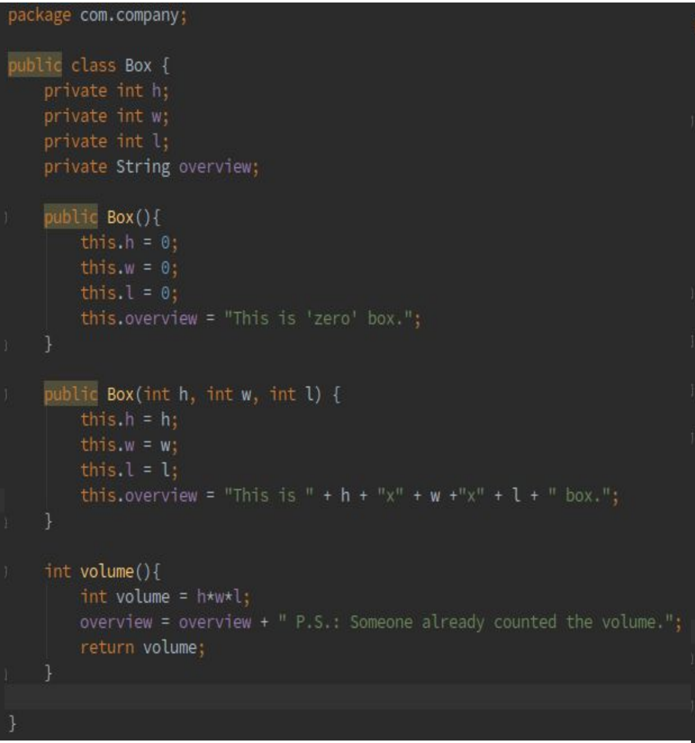


Result project structure

Demo

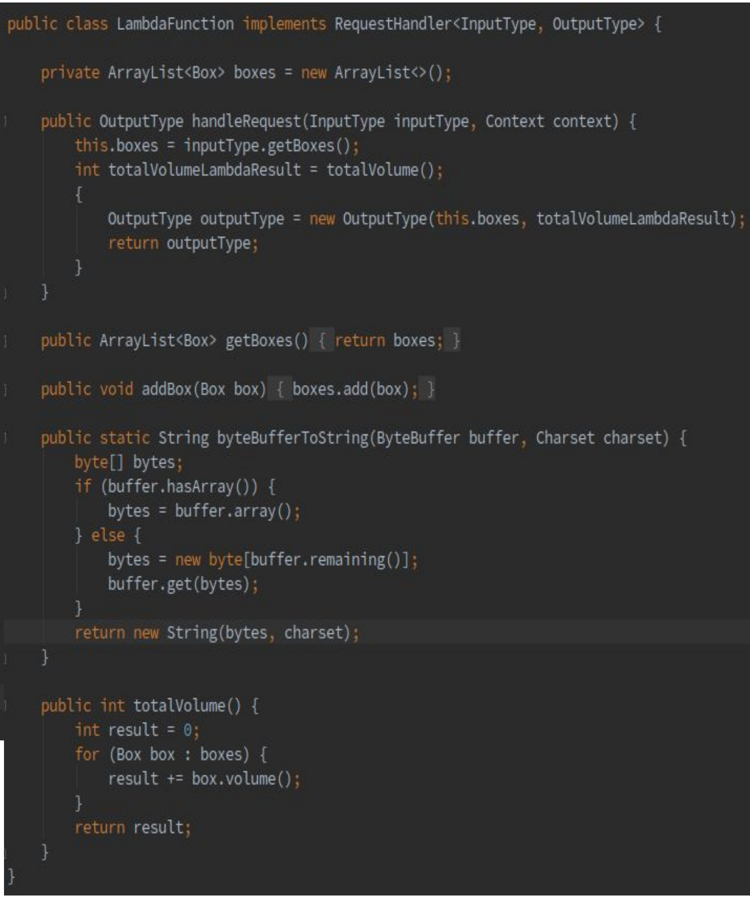


```java
package com.company;

import java.util.ArrayList;

public class BoxContainer {
    private ArrayList<Box> boxes = new ArrayList<>();

    public BoxContainer(){

    }
    public BoxContainer(ArrayList<Box> boxes) {
        this.boxes = boxes;
    }

    public ArrayList<Box> getBoxes() {
        return boxes;
    }

    public int totalVolume(){
        int result = 0;
        for (Box box :
                boxes) {
            result += box.volume();
        }
        return result;
    }
    public void addBox(Box box){
        boxes.add(box);
    }
}
```



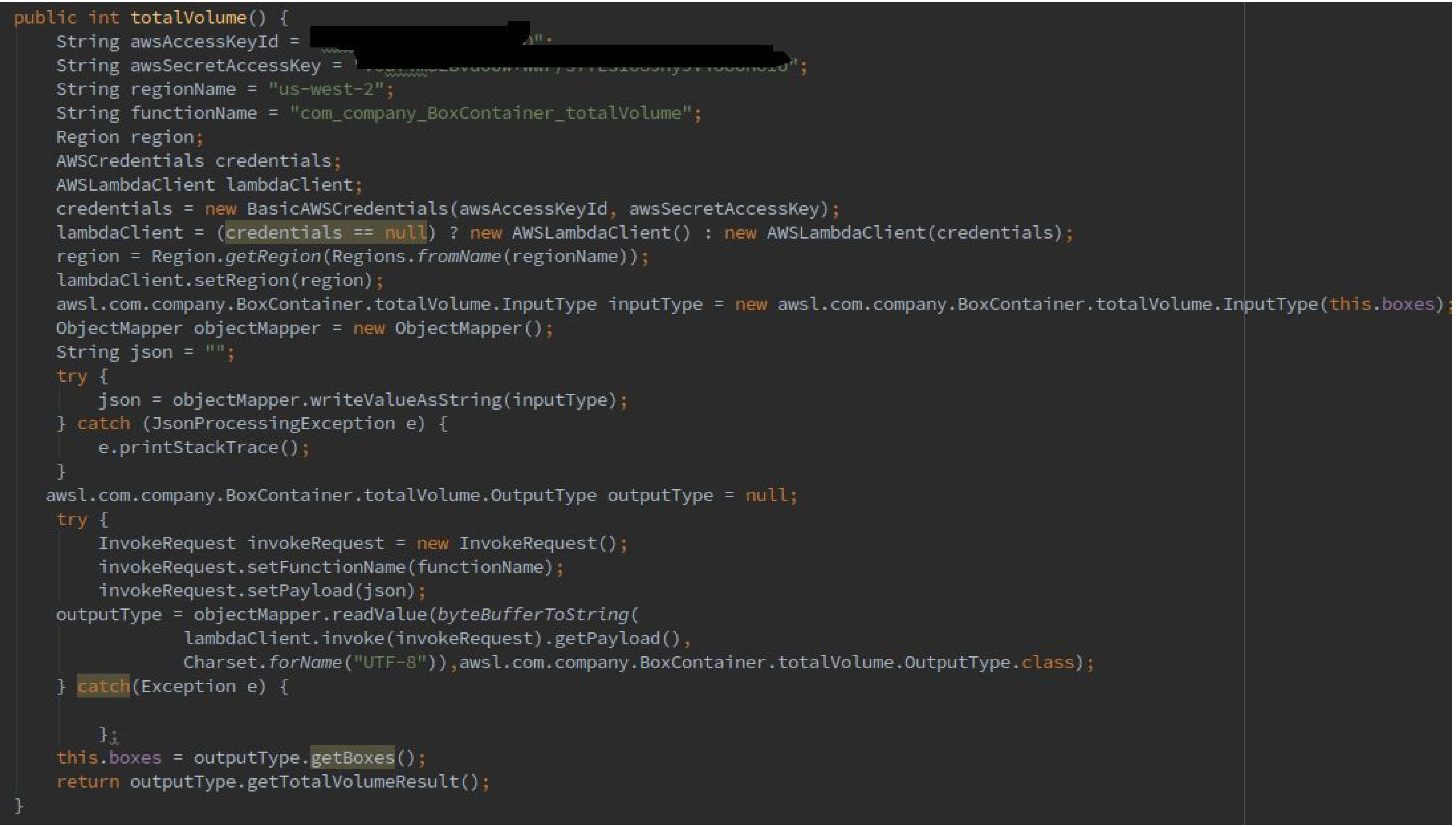
```java
package com.company;

public class Box {
    private int h;
    private int w;
    private int l;
    private String overview;

    public Box(){
        this.h = 0;
        this.w = 0;
        this.l = 0;
        this.overview = "This is 'zero' box.";
    }

    public Box(int h, int w, int l) {
        this.h = h;
        this.w = w;
        this.l = l;
        this.overview = "This is " + h + "x" + w +"x" + l + " box.";
    }

    int volume(){
        int volume = h*w*l;
        overview = overview + " P.S.: Someone already counted the volume.";
        return volume;
    }

}
```

Example of classes
from input project



```java
public class LambdaFunction implements RequestHandler<InputType, OutputType> {

    private ArrayList<Box> boxes = new ArrayList<>();

    public OutputType handleRequest(InputType inputType, Context context) {
        this.boxes = inputType.getBoxes();
        int totalVolumeLambdaResult = totalVolume();
        {
            OutputType outputType = new OutputType(this.boxes, totalVolumeLambdaResult);
            return outputType;
        }
    }

    public ArrayList<Box> getBoxes() { return boxes; }

    public void addBox(Box box) { boxes.add(box); }

    public static String byteBufferToString(ByteBuffer buffer, Charset charset) {
        byte[] bytes;
        if (buffer.hasArray()) {
            bytes = buffer.array();
        } else {
            bytes = new byte[buffer.remaining()];
            buffer.get(bytes);
        }
        return new String(bytes, charset);
    }

    public int totalVolume() {
        int result = 0;
        for (Box box : boxes) {
            result += box.volume();
        }
        return result;
    }
}
```

Example of Lambda Function for
‘totalVolume’ method



```java
public int totalVolume() {
    String awsAccessKeyId = [redacted]";
    String awsSecretAccessKey = [redacted]";
    String regionName = "us-west-2";
    String functionName = "com_company_BoxContainer_totalVolume";
    Region region;
    AWSCredentials credentials;
    AWSLambdaClient lambdaClient;
    credentials = new BasicAWSCredentials(awsAccessKeyId, awsSecretAccessKey);
    lambdaClient = (credentials == null) ? new AWSLambdaClient() : new AWSLambdaClient(credentials);
    region = Region.getRegion(Regions.fromName(regionName));
    lambdaClient.setRegion(region);
    awsl.com.company.BoxContainer.totalVolume.InputType inputType = new awsl.com.company.BoxContainer.totalVolume.InputType(this.boxes);
    ObjectMapper objectMapper = new ObjectMapper();
    String json = "";
    try {
        json = objectMapper.writeValueAsString(inputType);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
   awsl.com.company.BoxContainer.totalVolume.OutputType outputType = null;
    try {
        InvokeRequest invokeRequest = new InvokeRequest();
        invokeRequest.setFunctionName(functionName);
        invokeRequest.setPayload(json);
    outputType = objectMapper.readValue(byteBufferToString(
                lambdaClient.invoke(invokeRequest).getPayload(),
                Charset.forName("UTF-8")),awsl.com.company.BoxContainer.totalVolume.OutputType.class);
    } catch(Exception e) {

        };
    this.boxes = outputType.getBoxes();
    return outputType.getTotalVolumeResult();
}
```

Example of method from output project which invokes the function.

Limitations

So far it can translate code of console java applications without any references to file system.

More details about limits you can find at the project repository. (https://srv-lab-t-401.zhaw.ch/dord/service_tooling_initiative)

Next steps

We are planning to implement some features and optimise performance:

- Function project optimisation
- Java ‘this’ reference processing
- Make it more user friendly
- Transportation data optimisation
- Extend the circle of translatable code

Q?