Nada Bounajma 77807

Homework#2

**Technologies**

***Build Automation Tool :*** Gradle - automates software build and supports Java language which is convenient to code and run our server scripts.

***IDE* :** IntelliJ IDEA - recommended for small projects, supports Gradle, and can easily create and manage a Gradle project.

***Service Definition Language*** : WSDL - this web service API is language agnostic and will allow our client that is coded in Python to communicate with our cloud server that is coded in Java through remote procedure calls.

***Protocol*** : SOAP over HTTP - used by WSDL and specifies how the client and server interact

***Provider Programming Language*** : Java (Java API for XML web services)

***Consumer Programming Language*** : Python (Zeep Module - instead of Jax.WS it's called ZEEP and will support the interaction of both programs by generating from the client side the corresponding code to use the remote procedures)

**Development Approach : Code First Approach**

We opted in this case with the code first approach because we still are not familiar with the language WSDL, and it is faster to generate the WSDL file from the script that we will be coding than to hardcode it. Hence, it is the WSDL file that will adapt to how our programs will interact and not the other way around.

*Our project Development Process followed the same steps as the Development Process of the Calculator Case Study :*

**1. Server Side Development**

The business implementation of the service is in server.java that was marked with @WebService annotation. This service will then be published by Provider.java through the following URL :

**http://localhost:5050/server**

**2. WSDL file and Java server skeleton generation**

To generate our WSDL file, we run first the command ./gradlew build that will compile our server file. After it gets compiled, we will use it to generate our WSDL file through the following command :

**wsgen -wsdl -cp build/classes/java/main/ -d build/classes/java/main/ -r src/main/resources/ homework2.Calculator**

Graphical user interface, text, application

Description automatically generated

ServerService is the WSDL file and ServerService \_schema1 is the skeleton.

The URL of the service location inside the WSDL file was changed to **http://localhost:5050/server** and then we ran the project using the command ./gradlew run

**Client Side Development**

The consumer of the service is in client.py. We installed ZEEP through PIP using Terminal so that it works with our WSDL file and produces the client stub dynamically after running client.py.

Now that we have both our skeleton and stub generated, we change the mainClassName under build.gradle to homework1.Provider, then execute the command **./gradlew run that will run the Provider class and eventually publish the cloud service for it to be accessible by clients through the URL.**

**Design Document**

- The client can back up his or her files and keep them synchronized by connecting to the cloud.

- By backing up, we mean all files will get uploaded and stored in the server side.

- By synchronizing, we mean that any modifications on the client side will reflect on the server side as well (e.g., Change a file's content or Add a new file) Though, whatever is modified directly from the cloud will not have any effects on the client side.

- The Delete function was not implemented as a service because I assumed that it is not part of the functional requirements. Based on my understanding, a cloud service should only store and update files, not remove them whenever they are deleted from the client side. Hence the client will have to manually delete files from the cloud.

**Functions Used :**

- Create\_backup\_folder (line 13) : takes 1 argument which is the name of the backup folder provided by the client, and creates the folder in the server side.

- Sync function (line 22) : takes 3 arguments - file name, file size, and file content. In charge of keeping the files updated.

- Upload function (87) : takes the same arguments. In charge of uploading files that do not already exist in the cloud. Is called within the Sync function.

**How to Run the Application - *execution of programs were done through the command line provided in IntelliJ IDEA***

From my side :

- The server side is running on JDK 1.8 through the command : ./gradlew run

- And the client side is running on Python 3.8, through the following command :

python3 src/main/python/client.py [folder\_name]

[folder\_name] : an argument the client gives to specify which folder to backup

- The client will then be asked to prompt the name of the folder it wants to create on the server side.