|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | | |
|  |  | | | |  | |  | |
| Пермский государственный национальный  исследовательский университет | | | | | | | | | |
|  | |  | | | |  | |
| ОТЧЕТ  по лабораторной работе №3  «Разработка распределенного приложения, использующего технологию веб-служб»  по дисциплине  «Технологии разработки распределенных приложений» | | | | | | | |
|  | | |  |  | | | |
| Работу выполнил студент гр. КМБ-16  Дровосеков Даниил Алексеевич  03.12.2020 | | |  | Проверил  доцент кафедры прикладной математики и информатики,  к. ф.-м.н., доц.  Деменев Алексей Геннадьевич | | | |
|  | | |  |  | | | |

|  |
| --- |
| Пермь, 2020 г. |

**Введение**

**Цель**: Изучение возможностей технологии веб-служб для создания распределенных приложений.

**Задача**: реализовать веб-сервис и приложение, его использующее, при этом

1. Веб-сервис и приложение, его использующее, должны быть разработаны на разных объектно-ориентированных языках программирования.
2. Веб-сервис должен принимать параметры и передавать приложению результат.
3. В качестве параметров должны передаваться объекты классов, написанных самостоятельно, т.е. не должны передаваться строки, числа или другие простейшие типы

**Постановка задачи**

Обеспечить проектирование, реализацию, проверку созданного приложения требованиям в таблице:

***Распределение баллов за выполнение работы (***Максимальное количество баллов — 6)***:***

Таблица 1. Требования к выполнению работы

|  |  |
| --- | --- |
| **Требование к заданию** | **Максимальное количество баллов** |
| Приложение, написанное студентом, работает в сети Интернет без сбоев. | 2 |
| Веб-сервис и приложение, его использующее, написаны на разных языках программирования. | 1 |
| Веб-сервис принимает от приложения исходные данные и возвращает результат. | 1 |
| В качестве параметров веб-сервис принимает объекты классов. | 2 |

**Проектирование**

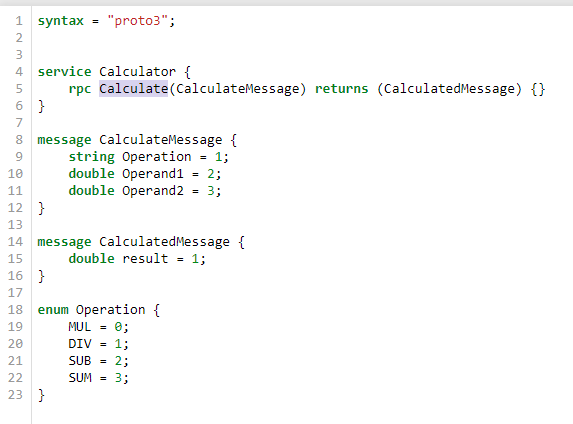
Веб-сервис осуществляет вычисления для выполнения ряда арифметических операций над двумя целыми числами: сложение, вычитание, умножение, целочисленное деление.

В качестве фреймворка для вызова удалённых процедур был выбран gRPC[1]. gRPC – фреймворк с открытым исходным кодом, разработанный компанией Google, доступный для бесплатного использования. gRPC также является веб сервисом, так как использует язык описания интерфейсов IDL[9], по определению веб сервисов из Web Services Glossary[8].

В качестве протокола сериализации был выбран Protocol Buffers[3]. Протокол с открытым исходным кодом также размещён в публичном репозиторие и доступен для скачивания.

В качестве утилиты для генерации кода использовалась утилита *protoc*[2]. Поставляется вместе с Protocol Buffers.

Приложение при передаче данных оперирует объектами классов, описанных в файле serv\_proto.proto. Конкретные объекты (CalculateMessage, CalculatedMessage)



**Рис. 1.** Содержимое файла serv\_proto.proto

Таким образом, были спроектированы два объекта:

* «CalculateMessage» с двумя числовыми полями (double Operand1 и double Operand2) – для первого и второго операндов – и одним текстовым (string Operation) – для наименования операции. Будет формироваться клиентским приложением и посылаться серверу.
* «CalculatedMessage» с одним числовым полем (double result) – для результата выполнения арифметической операции. Будет формироваться сервером и отправляться клиенту для передачи результата выполнения запроса.

Также в файле описан сервис «Calculator», содержащий единственный метод «Calculate», принимающий на вход объект типа «CalculateMessage» и возвращающий в качестве результата объект типа «CalculatedMessage».

Для разработки серверного и клиентского приложений был выбран объектно-ориентированный язык программирования Python 3.7 [4,5]. В качестве среды разработки была выбрана среда Jupyter Notebook 7.2 [6,7].

**Сценарии использования программы-клиента**

Сценарий использования №1 – сложение

Предусловия: Сервер запущен

1. Выбор операции «Сложение»
2. Ввод первого целого числа
3. Ввод второго целого числа
4. Ожидание получения ответа

Ожидаемый результат: от сервера получен ответ, соответствующий результату сложения ведённых чисел

Сценарий использования №2 – вычитание

Предусловия: Сервер запущен

1. Выбор операции «Вычитание»
2. Ввод первого целого числа
3. Ввод второго целого числа
4. Ожидание получения ответа

Ожидаемый результат: от сервера получен ответ, соответствующий результату вычитания ведённых чисел

Сценарий использования №3 – умножение

Предусловия: Сервер запущен

1. Выбор операции «Умножение»
2. Ввод первого целого числа
3. Ввод второго целого числа
4. Ожидание получения ответа

Ожидаемый результат: от сервера получен ответ, соответствующий результату умножения ведённых чисел

Сценарий использования №4 – целочисленное деление

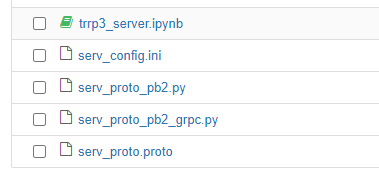
Предусловия: Сервер запущен

1. Выбор операции «Целочисленное деление»
2. Ввод первого целого числа
3. Ввод второго целого числа
4. Ожидание получения ответа

Ожидаемый результат: от сервера получен ответ, соответствующий результату целочисленного деления ведённых чисел

**Тестирование**

При разработке с помощью инструмента gRPC на основе файла serv\_proto.proto было получено два автогенерируемых класса – serv\_proto\_pb2.py и serv\_proto\_pb2\_grpc.py – для работы с спроектированным сервисом.



**Рис. 2.** Автогенерируемые классы serv\_proto\_pb2.py и serv\_proto\_pb2\_grpc.py

Для настройки сетевых параметров применяются конфигурационные файлы «serv\_config.ini» и «cl\_config.ini» для сервера и клиента соответственно.



**Рис. 3.** Содержимое файла «serv\_config.ini»



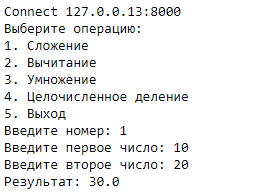
**Рис. 4.** Содержимое файла «cl\_config.ini»

Запустим сервер:



**Рис. 5.** Запуск сервера

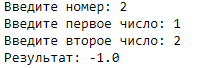
Сценарий №1



**Рис. 6.** Работа клиента

Результат: От сервера получен ответ, соответствующий результату сложения ведённых чисел. Тест пройден успешно.

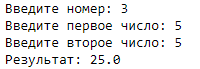
Сценарий №2



**Рис. 7.** Работа клиента

Результат: От сервера получен ответ, соответствующий результату вычитания ведённых чисел. Тест пройден успешно.

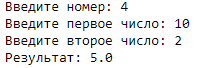
Сценарий №3



**Рис. 8.** Работа клиента

Результат: От сервера получен ответ, соответствующий результату умножения ведённых чисел. Тест пройден успешно.

Сценарий №4



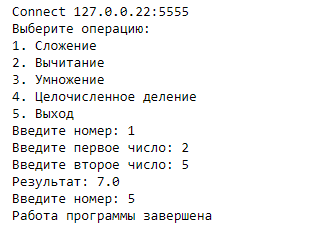
**Рис. 9.** Работа клиента

Результат: От сервера получен ответ, соответствующий результату деления ведённых чисел. Тест пройден успешно.

Запустим приложения с измененными параметрами сети, работоспособность сохраняется.



**Рис. 10.** Работа сервера с измененными параметрами сети



**Рис. 11.** Работа клиента с измененными параметрами сети

**Заключение**

В результате работы была достигнута цель и выполнены следующие требования:

* Требование «Приложение, написанное студентом, работает в сети Интернет без сбоев» выполнено полностью. Ожидаемое количество баллов: 2.
* Требование «Веб-сервис и приложение, его использующее, написаны на разных языках программирования» не выполнено. Ожидаемое количество баллов: 0.
* Требование «Веб-сервис принимает от приложения исходные данные и возвращает результат» выполнено полностью. Ожидаемое количество баллов: 1.
* Требование «В качестве параметров веб-сервис принимает объекты классов» выполнено полностью. Ожидаемое количество баллов: 2.

**Перечень использованных источников**

1. grpc/LICENSE. url: <https://github.com/grpc/grpc/blob/master/LICENSE> (дата обращения: 03.12.2020)
2. protocolbuffers/protobuf. url: <https://github.com/protocolbuffers/protobuf> (дата обращения: 03.12.2020)
3. Лицензия: url: <https://github.com/protocolbuffers/protobuf/blob/master/LICENSE> (дата обращения: 03.12.2020)
4. Python 3.7.0. url: <https://www.python.org/downloads/release/python-370/> (дата обращения: 03.12.2020)
5. History and License. url: <https://docs.python.org/3.7/license.html> (дата обращения: 03.12.2020)
6. Installation. url: <https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html> (дата обращения: 03.12.2020)
7. Documentation. url: <https://jupyter.org/documentation> (дата обращения: 12.11.2020)
8. Web Services Glossary. url: <https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice> (дата обращения: 03.12.2020)
9. Interface Definition Language. url: <https://www.omg.org/spec/IDL/4.2/> (дата обращения: 03.12.2020)

**Приложение**

trrp3\_server:

|  |
| --- |
| import serv\_proto\_pb2  import serv\_proto\_pb2\_grpc  import grpc  from concurrent import futures  from configparser import ConfigParser  class CalculatorServicer(serv\_proto\_pb2\_grpc.CalculatorServicer):  def Calculate(self,request\_msg, context):  first\_num = request\_msg.Operand1  second\_num = request\_msg.Operand2  operation = request\_msg.Operation  if(operation=='MUL'):  result = first\_num \* second\_num  if(operation=='DIV'):  result = first\_num // second\_num  if (operation == 'SUB'):  result = first\_num - second\_num  if (operation == 'SUM'):  result = first\_num + second\_num  response = serv\_proto\_pb2.CalculatedMessage(result = result)  return response  configur = ConfigParser()  configur.read('serv\_config.ini')  ip = configur.get('network', 'ip')  port = configur.get('network', 'port')  server = grpc.server(futures.ThreadPoolExecutor(max\_workers=4))  serv\_proto\_pb2\_grpc.add\_CalculatorServicer\_to\_server(CalculatorServicer(), server)  print('Starting server on ' + str(ip) + ':' + str(port))  server.add\_insecure\_port( str(ip) + ':' + str(port) )  server.start()  server.wait\_for\_termination() |

trrp3\_client:

|  |
| --- |
| import serv\_proto\_pb2  import serv\_proto\_pb2\_grpc  import grpc  from concurrent import futures  from configparser import ConfigParser  configur = ConfigParser()  configur.read('cl\_config.ini')  ip = configur.get('network', 'ip')  port = configur.get('network', 'port')  print('Connect ' + str(ip) + ':' + str(port))  channel = grpc.insecure\_channel(str(ip) + ':' + str(port))  stub = serv\_proto\_pb2\_grpc.CalculatorStub(channel)  row\_num = 0  while(row\_num!=5):  print('Выберите операцию: ')  print('1. Сложение')  print('2. Вычитание')  print('3. Умножение')  print('4. Целочисленное деление')  print('5. Выход')  row\_num = int(input('Введите номер: '))  while row\_num!=5:  first\_num = float(input('Введите первое число: '))  second\_num = float(input('Введите второе число: '))  if(row\_num==1):  reaqest = serv\_proto\_pb2.CalculateMessage(Operation='SUM',Operand1=first\_num,Operand2=second\_num)  response = stub.Calculate(reaqest)  if(row\_num==2):  reaqest = serv\_proto\_pb2.CalculateMessage(Operation='SUB',Operand1=first\_num,Operand2=second\_num)  response = stub.Calculate(reaqest)  if(row\_num==3):  reaqest = serv\_proto\_pb2.CalculateMessage(Operation='MUL',Operand1=first\_num,Operand2=second\_num)  response = stub.Calculate(reaqest)  if(row\_num==4):  reaqest = serv\_proto\_pb2.CalculateMessage(Operation='DIV',Operand1=first\_num,Operand2=second\_num)  response = stub.Calculate(reaqest)  print('Результат: ' + str(response.result))  row\_num = int(input('Введите номер: '))    print('Работа программы завершена') |

serv\_proto.proto:

|  |
| --- |
| syntax = "proto3";  service Calculator {  rpc Calculate(CalculateMessage) returns (CalculatedMessage) {}  }  message CalculateMessage {  string Operation = 1;  double Operand1 = 2;  double Operand2 = 3;  }  message CalculatedMessage {  double result = 1;  }  enum Operation {  MUL = 0;  DIV = 1;  SUB = 2;  SUM = 3;  } |

serv\_proto\_pb2\_grpc:

|  |
| --- |
| # Generated by the gRPC Python protocol compiler plugin. DO NOT EDIT!  """Client and server classes corresponding to protobuf-defined services."""  import grpc  import serv\_proto\_pb2 as serv\_\_proto\_\_pb2  class CalculatorStub(object):  """Missing associated documentation comment in .proto file."""  def \_\_init\_\_(self, channel):  """Constructor.  Args:  channel: A grpc.Channel.  """  self.Calculate = channel.unary\_unary(  '/Calculator/Calculate',  request\_serializer=serv\_\_proto\_\_pb2.CalculateMessage.SerializeToString,  response\_deserializer=serv\_\_proto\_\_pb2.CalculatedMessage.FromString,  )  class CalculatorServicer(object):  """Missing associated documentation comment in .proto file."""  def Calculate(self, request, context):  """Missing associated documentation comment in .proto file."""  context.set\_code(grpc.StatusCode.UNIMPLEMENTED)  context.set\_details('Method not implemented!')  raise NotImplementedError('Method not implemented!')  def add\_CalculatorServicer\_to\_server(servicer, server):  rpc\_method\_handlers = {  'Calculate': grpc.unary\_unary\_rpc\_method\_handler(  servicer.Calculate,  request\_deserializer=serv\_\_proto\_\_pb2.CalculateMessage.FromString,  response\_serializer=serv\_\_proto\_\_pb2.CalculatedMessage.SerializeToString,  ),  }  generic\_handler = grpc.method\_handlers\_generic\_handler(  'Calculator', rpc\_method\_handlers)  server.add\_generic\_rpc\_handlers((generic\_handler,))  # This class is part of an EXPERIMENTAL API.  class Calculator(object):  """Missing associated documentation comment in .proto file."""  @staticmethod  def Calculate(request,  target,  options=(),  channel\_credentials=None,  call\_credentials=None,  insecure=False,  compression=None,  wait\_for\_ready=None,  timeout=None,  metadata=None):  return grpc.experimental.unary\_unary(request, target, '/Calculator/Calculate',  serv\_\_proto\_\_pb2.CalculateMessage.SerializeToString,  serv\_\_proto\_\_pb2.CalculatedMessage.FromString,  options, channel\_credentials,  insecure, call\_credentials, compression, wait\_for\_ready, timeout, metadata) |

serv\_proto\_pb2:

|  |
| --- |
| # -\*- coding: utf-8 -\*-  # Generated by the protocol buffer compiler. DO NOT EDIT!  # source: serv\_proto.proto  """Generated protocol buffer code."""  from google.protobuf.internal import enum\_type\_wrapper  from google.protobuf import descriptor as \_descriptor  from google.protobuf import message as \_message  from google.protobuf import reflection as \_reflection  from google.protobuf import symbol\_database as \_symbol\_database  # @@protoc\_insertion\_point(imports)  \_sym\_db = \_symbol\_database.Default()  DESCRIPTOR = \_descriptor.FileDescriptor(  name='serv\_proto.proto',  package='',  syntax='proto3',  serialized\_options=None,  create\_key=\_descriptor.\_internal\_create\_key,  serialized\_pb=b'\n\x10serv\_proto.proto\"I\n\x10\x43\x61lculateMessage\x12\x11\n\tOperation\x18\x01 \x01(\t\x12\x10\n\x08Operand1\x18\x02 \x01(\x01\x12\x10\n\x08Operand2\x18\x03 \x01(\x01\"#\n\x11\x43\x61lculatedMessage\x12\x0e\n\x06result\x18\x01 \x01(\x01\*/\n\tOperation\x12\x07\n\x03MUL\x10\x00\x12\x07\n\x03\x44IV\x10\x01\x12\x07\n\x03SUB\x10\x02\x12\x07\n\x03SUM\x10\x03\x32\x42\n\nCalculator\x12\x34\n\tCalculate\x12\x11.CalculateMessage\x1a\x12.CalculatedMessage\"\x00\x62\x06proto3'  )  \_OPERATION = \_descriptor.EnumDescriptor(  name='Operation',  full\_name='Operation',  filename=None,  file=DESCRIPTOR,  create\_key=\_descriptor.\_internal\_create\_key,  values=[  \_descriptor.EnumValueDescriptor(  name='MUL', index=0, number=0,  serialized\_options=None,  type=None,  create\_key=\_descriptor.\_internal\_create\_key),  \_descriptor.EnumValueDescriptor(  name='DIV', index=1, number=1,  serialized\_options=None,  type=None,  create\_key=\_descriptor.\_internal\_create\_key),  \_descriptor.EnumValueDescriptor(  name='SUB', index=2, number=2,  serialized\_options=None,  type=None,  create\_key=\_descriptor.\_internal\_create\_key),  \_descriptor.EnumValueDescriptor(  name='SUM', index=3, number=3,  serialized\_options=None,  type=None,  create\_key=\_descriptor.\_internal\_create\_key),  ],  containing\_type=None,  serialized\_options=None,  serialized\_start=132,  serialized\_end=179,  )  \_sym\_db.RegisterEnumDescriptor(\_OPERATION)  Operation = enum\_type\_wrapper.EnumTypeWrapper(\_OPERATION)  MUL = 0  DIV = 1  SUB = 2  SUM = 3  \_CALCULATEMESSAGE = \_descriptor.Descriptor(  name='CalculateMessage',  full\_name='CalculateMessage',  filename=None,  file=DESCRIPTOR,  containing\_type=None,  create\_key=\_descriptor.\_internal\_create\_key,  fields=[  \_descriptor.FieldDescriptor(  name='Operation', full\_name='CalculateMessage.Operation', index=0,  number=1, type=9, cpp\_type=9, label=1,  has\_default\_value=False, default\_value=b"".decode('utf-8'),  message\_type=None, enum\_type=None, containing\_type=None,  is\_extension=False, extension\_scope=None,  serialized\_options=None, file=DESCRIPTOR, create\_key=\_descriptor.\_internal\_create\_key),  \_descriptor.FieldDescriptor(  name='Operand1', full\_name='CalculateMessage.Operand1', index=1,  number=2, type=1, cpp\_type=5, label=1,  has\_default\_value=False, default\_value=float(0),  message\_type=None, enum\_type=None, containing\_type=None,  is\_extension=False, extension\_scope=None,  serialized\_options=None, file=DESCRIPTOR, create\_key=\_descriptor.\_internal\_create\_key),  \_descriptor.FieldDescriptor(  name='Operand2', full\_name='CalculateMessage.Operand2', index=2,  number=3, type=1, cpp\_type=5, label=1,  has\_default\_value=False, default\_value=float(0),  message\_type=None, enum\_type=None, containing\_type=None,  is\_extension=False, extension\_scope=None,  serialized\_options=None, file=DESCRIPTOR, create\_key=\_descriptor.\_internal\_create\_key),  ],  extensions=[  ],  nested\_types=[],  enum\_types=[  ],  serialized\_options=None,  is\_extendable=False,  syntax='proto3',  extension\_ranges=[],  oneofs=[  ],  serialized\_start=20,  serialized\_end=93,  )  \_CALCULATEDMESSAGE = \_descriptor.Descriptor(  name='CalculatedMessage',  full\_name='CalculatedMessage',  filename=None,  file=DESCRIPTOR,  containing\_type=None,  create\_key=\_descriptor.\_internal\_create\_key,  fields=[  \_descriptor.FieldDescriptor(  name='result', full\_name='CalculatedMessage.result', index=0,  number=1, type=1, cpp\_type=5, label=1,  has\_default\_value=False, default\_value=float(0),  message\_type=None, enum\_type=None, containing\_type=None,  is\_extension=False, extension\_scope=None,  serialized\_options=None, file=DESCRIPTOR, create\_key=\_descriptor.\_internal\_create\_key),  ],  extensions=[  ],  nested\_types=[],  enum\_types=[  ],  serialized\_options=None,  is\_extendable=False,  syntax='proto3',  extension\_ranges=[],  oneofs=[  ],  serialized\_start=95,  serialized\_end=130,  )  DESCRIPTOR.message\_types\_by\_name['CalculateMessage'] = \_CALCULATEMESSAGE  DESCRIPTOR.message\_types\_by\_name['CalculatedMessage'] = \_CALCULATEDMESSAGE  DESCRIPTOR.enum\_types\_by\_name['Operation'] = \_OPERATION  \_sym\_db.RegisterFileDescriptor(DESCRIPTOR)  CalculateMessage = \_reflection.GeneratedProtocolMessageType('CalculateMessage', (\_message.Message,), {  'DESCRIPTOR' : \_CALCULATEMESSAGE,  '\_\_module\_\_' : 'serv\_proto\_pb2'  # @@protoc\_insertion\_point(class\_scope:CalculateMessage)  })  \_sym\_db.RegisterMessage(CalculateMessage)  CalculatedMessage = \_reflection.GeneratedProtocolMessageType('CalculatedMessage', (\_message.Message,), {  'DESCRIPTOR' : \_CALCULATEDMESSAGE,  '\_\_module\_\_' : 'serv\_proto\_pb2'  # @@protoc\_insertion\_point(class\_scope:CalculatedMessage)  })  \_sym\_db.RegisterMessage(CalculatedMessage)  \_CALCULATOR = \_descriptor.ServiceDescriptor(  name='Calculator',  full\_name='Calculator',  file=DESCRIPTOR,  index=0,  serialized\_options=None,  create\_key=\_descriptor.\_internal\_create\_key,  serialized\_start=181,  serialized\_end=247,  methods=[  \_descriptor.MethodDescriptor(  name='Calculate',  full\_name='Calculator.Calculate',  index=0,  containing\_service=None,  input\_type=\_CALCULATEMESSAGE,  output\_type=\_CALCULATEDMESSAGE,  serialized\_options=None,  create\_key=\_descriptor.\_internal\_create\_key,  ),  ])  \_sym\_db.RegisterServiceDescriptor(\_CALCULATOR)  DESCRIPTOR.services\_by\_name['Calculator'] = \_CALCULATOR  # @@protoc\_insertion\_point(module\_scope) |