的对象，将参数值的类型设置成 ParameterType.PARAMETER_STRING（即字符串类型），然后对字符串进行赋值，最后将参数值对象赋值给参数对象的 value 属性。第三步则调用 call_set_parameters 更新参数，使用 [param] 将 param 转换成一个数组，遍历响应结果。如果结果中 successful 属性为 True，则表示设置成功；否则表示设置失败，然后输出失败的原因。

最后修改 main 函数，修改不同参数并发送请求，如代码清单 4-46 所示。

代码清单 4-46　更新检测模型

```
def main(args=None):
    rclpy.init(args=args)
    face_detect_client = FaceDetectorClient()
    face_detect_client.update_detect_model('hog')
    face_detect_client.send_request()
    face_detect_client.update_detect_model('cnn')
    face_detect_client.send_request()
    rclpy.spin(face_detect_client)
    rclpy.shutdown()
```

在上面的 main 函数中，首先调用 update_detect_model 方法更新服务端检测模型为 hog，接着发送检测请求，然后修改检测模型为 cnn，再次发送检测请求。

再次构建功能包，运行人脸检测服务端，然后运行客户端，客户端运行命令及结果如代码清单 4-47 所示。

代码清单 4-47　运行人脸检测客户端

```
$ ros2 run demo_python_service face_detect_client_node
---
[INFO] [1682617409.497532963] [face_detect_client]: 参数 face_locations_model 设置为 hog
[INFO] [1682617409.763817490] [face_detect_client]: 接收到响应：图像中共有：3 张脸，耗时 0.23881101608276367
[INFO] [1682617409.848686984] [face_detect_client]: 参数 face_locations_model 设置为 cnn
[INFO] [1682617442.451341096] [face_detect_client]: 接收到响应：图像中共有：3 张脸，耗时 29.150156021118164
```

从上面的检测结果可以看出，当检测模型参数设置为 hog 时，只需 0.2s 左右就可以检测完成；当检测模型参数设置成 cnn 时，则需要 29s。

除了可以通过服务设置其他节点的参数，还可以通过服务获取其他节点的参数列表和值，原理和方法相同，这里就不再赘述。好了，通过本节的学习，相信你已经掌握在 Python 节点中声明和设置参数的方法，下一节我们来学习如何在 C++ 节点中使用参数。

4.5　在 C++ 节点中使用参数

在 C++ 节点的代码中使用参数的方式和 Python 节点基本一致。所以本节将结合 4.3 节的巡逻海龟项目，将服务端中控制器的比例系数 k_ 和最大速度 max_speed_ 参数化。