YTU RACING SERVER REPORT

Name-Surname :Adem Berke Nargul

Department : Autonomous

Date : 06.10.2023

catkin_ws > src > calculate_factorial > src > factorial_request.cpp > ...

```cpp
#include "ros/node_handle.h"
#include "ros/ros.h"
#include "calculate_factorial/server.h"
#include "ros/service_client.h"

int main(int ac, char **av)
{
    ros::init(ac, av, "factorial_calculate_client");

    if (ac != 2)
    {
        ROS_INFO("Wrong argument!.\n");
        return (1);
    }

    ros::NodeHandle node_handle;

    ros::ServiceClient client;

    calculate_factorial::server serv;
    // programi calistirirken verilecek olan sayıyı string olarak tuttugu için longa ceviriyoruz ve
    // service'in icinde request'in sayisi olarak belirliyoruz. Bunu servera gonderecegiz.
    serv.request.number = atol(av[1]);

    client = node_handle.serviceClient<calculate_factorial::server>("factorial_calculate");

    if (client.call(serv))
    {
        ROS_INFO("Service called successfully and factorial is: %ld\n", serv.response.result);
    }
    else
    {
        ROS_ERROR("Failed to call service");
        return (1);
    }

    return (0);
}
```

Görsel 1

Client (istemci) kodu

ros::ServiceClient : Bu, ROS'ta bir hizmet istemcisini temsil eden C++ sınıfıdır.
node_handle.serviceClient<calculate_factorial::server>("factorial_calculate"): Bu komut, node_handle adlı bir ros::NodeHandle nesnesi üzerinden factorial_calculate adlı hizmet ile iletişim kuracak bir istemci oluşturur. Bu istemci, call() fonksiyonu kullanılarak hizmeti çağırır ve ardından hizmet sunucusundan gelen yanıtı geri alır. "factorial_calculate" ifadesi, hizmetin adını temsil eder. <calculate_factorial::server> ifadesi, bu hizmetin talep ve yanıt türünü belirtir. Burada calculate_factorial dosyasının server adlı request ve response mesajlarına sahip olduğunu belirtiyoruz.
client.call(server) : Eğer hizmet başarılı bir şekilde çağrılırsa, hesaplanan faktöriyel değeri ekrana yazdırılır. Hizmet çağrısı başarısız olursa hata mesajı gösterilir ve program sonlanır.

catkin_ws > src > calculate_factorial > src > factorial_calc.cpp > ...

```cpp
#include "ros/init.h"
#include "ros/node_handle.h"
#include "ros/service_server.h"
#include "ros/ros.h"
#include "calculate_factorial/server.h"

bool factorialCalc(calculate_factorial::server::Request& req, calculate_factorial::server::Response& resp)
{
    // basit bir factorial bulma fonksiyonu
    long int result = 1;
    for (long int i = 1; i <= req.number; i++)
        result *= i;
    // clientin de erişmesi için sonucu service'in icindeki resulta atıyoruz.
    resp.result = result;
    ROS_INFO("Input number: %ld, Result: %ld\n", (long int)req.number, resp.result);
    return true;
}


int main(int ac, char **av)
{
    ros::init(ac, av, "factorial_calculate_server", 0);

    // NodeHandle turunden bir degisken tanimliyoruz.
    ros::NodeHandle node_handle;

    // ServiceServer turunden degisken tanimliyoruz.
    ros::ServiceServer server;

    // server degiskenini baslatiyoruz.
    server = node_handle.advertiseService("factorial_calculate", factorialCalc);

    ros::spin();

    return (0);
}
```

Görsel 2

Server kodu

factorialCalc() fonksiyonu : Gelen bir sayının faktöriyelini hesaplar ve cevabı (resp.result) hizmet talep eden istemciye geri döner. Bu fonksiyon, calculate_factorial::server::Request türünden bir istek (req) ve calculate_factorial::server::Response türünden bir (resp) parametre alır. Request türünden gelen verinin sayısının faktöriyelini bulur ve response türünden gelen verinin sayısına eşitler. Böylelikle servis içinde faktöriyeli bulunmuş bir sayı Response.result değişkeninde tutulur.

node_handle.advertiseService("factorial_calculate", factorialCalc): Bir Ros hizmet sunucusunu başlatmak için kullanılır. Bu fonksiyon, iki ana parametre alır: Servis adı ve Servis fonksiyonu.

Servis adı: Diğer düğümler, bu isimle bu hizmete talep gönderebilir.

Servis fonksiyonu: Bu fonksiyon, gelen taleplere yanıt verir ve istemcilere hizmetin sonuçlarını döner.

Server nedir?

Ros'ta "server" (hizmet sunucusu) bir düğümün diğer düğümlere belirli görevleri yerine getirme kapasitesi sunan bir mekanizmayı temsil eder. Server mantığı, bir düğümün bir hizmeti yerine getirme yeteneğini ve dış dünyadan talep almasını sağlar. Bu mekanizma, kullanıcılardan gelen isteklere yanıt verebilir, veritabanları güncelleyebilir, hesaplamalar yapabilir veya sensör verilerini işleyebilir.

Serverlar, yaygın olarak hizmet çağrıları alır, bu istekleri işler ve ardından sonuçları geri gönderir. Bu sayede, sistemdeki farklı düğümler arasında işbirliği ve koordinasyon sağlanır. Serverlar ayrıca, uygulamaların gerçek zamanlı gereksinimlerini karşılayabilmek için dağıtılmış ve paralel hesaplamaları destekler.

Kullanım alanlarına göre, serverlar robot kontrolü, yapay zeka hesaplamaları, harita oluşturma, sensör verilerinin işlenmesi gibi bir dizi görevi yerine getirebilir. Bu sayede, Ros uygulamaları daha karmaşık ve özelleştirilmiş işlevselliklerle donatılabilir, genişletilebilir ve özelleştirilebilir. Server mantığı, Ros uygulamalarında dağıtılmış ve esnek bir yapı oluşturmanın anahtarıdır.

Publisher ve subscriber mantığından farkı ise: Serverlar, diğer düğümlerden gelen özel taleplere yanıt verirken, publisherlar verileri belirli bir konuya düzenli olarak yayınlarlar. Yani, serverlar cliemt tarafından gelen belirli bir talebi işlerken, publisherlar sürekli olarak veri akışını sürdürür

catkin_ws > src > calculate_factorial > srv > server.srv

```
1   int64 number
2   ---
3   int64 result
4
```

1. ".srv" uzantılı dosya : Bu .srv dosyası, "number" adında bir tamsayı parametresini alan bir hizmeti temsil eder. Bu parametre, hizmeti çağıran düğümden alınan bir tamsayı değerini temsil eder. Hizmet, "result" adında bir tamsayı değeriyle yanıt verir. Bu değer, hizmet tarafından hesaplanan sonucu veya işlem sonucunu temsil eder. Hizmet çağrıldığında, düğüm "number" parametresini hizmete gönderir, ardından hizmet bu sayıyı kullanarak bir hesaplama gerçekleştirir ve sonucu "result" parametresiyle geri döndürür.Bu .srv dosyası, ROS uygulamalarında farklı düğümler arasında veri alışverişi için kullanılır. Hizmet çağrıları genellikle hesaplama veya işlem gerektiren durumlar için kullanılır ve bir düğüm diğerinden bir hizmet çağırarak belirli bir görevi gerçekleştirmesini talep edebilir. Bu dosya sayesinde, ROS düğümleri arasında veri değişimi ve işbirliği gerçekleştirilebilir. Bunu bir template gibi düşünebiliriz. Aynı zamanda bu dosyayı Görsel 1 ve Görsel 2'de de görebileceğiniz gibi bir kütüphane olarak kullanıyoruz. Bu hizmet ile ilgili bütün verilere bu kütüphane aracılığıyla (linked) erişiyoruz.