an enclosing-function local variable cannot be referenced in a lambda body unless if it is in capture list

I have json::value object and I try to get values in a struct but i get this error about capture list. I understand that in then phrase this bracet [] holds capture list but i cant figure out how. How can i return a value in lambda function?

```
   void JsonDeneme::setValues(json::value obj)
{
    weather.coord.lon = obj.at(L"coord").at(L"lon").as_double();
    weather.coord.lat= obj.at(L"coord").at(L"lat").as_double();

}

void JsonDeneme::getHttp()
{
    //json::value val;
    http_client client(U("http://api.openweathermap.org/data/2.5/weather?q=Ankara,TR"));

    client.request(methods::GET)

    .then([](http_response response) -> pplx::task<json::value>
    {

        if (response.status_code() == status_codes::OK)
        {
            printf("Received response status code:%u\n", response.status_code());
            return response.extract_json();
        }
        return pplx::task_from_result(json::value());
    })

    .then([ ](pplx::task<json::value> previousTask)
    {
        try
        {
            json::value   v = previousTask.get();
            setValues(v);//-----------------------------------------
        }
        catch (http_exception const & e)
        {
            wcout << e.what() << endl;
        }
    })
    .wait();

}
```

c++ function lambda

asked Nov 13 '14 at 7:36
user2957741
20 5

1 What is the error you're getting? – SingerOfTheFall Nov 13 '14 at 7:37

an enclosing-function local variable cannot be referenced in a lambda body unless if it is in capture list – user2957741 Nov 13 '14 at 7:48

this is fixed when i added [this] but i try to understand why – user2957741 Nov 13 '14 at 7:49

1 Answer

The capture-list is what you put inbetween the square brackets. Look at this example:

```
void foo()
{
    int i = 0;
    []()
    {
        i += 2;
    }
}
```

Here the labda does not capture anything, thus it will not have access to the enclosing scope, and will not know what `i` is. Now, let's capture *everything* by reference:

```
void foo()
{
    int i = 0;
    [&]()//note the &. It means we are capturing all of the enclosing scope 
    variables by reference
    {
        i += 2;
    }
    cout << 2;
}
```

In this example, the `i` inside the lambda is a reference to the `i` in the enclosing scope.

In your example, you have a lambda inside a member-function of an object. You are trying to call the object's function: `setValues(v)` , but your capture list is empty, so your lambda does not know what `setValues` is. Now, if you capture `this` in the lambda, the lambda will have access to all of the object's methods, because `setValues(v)` is the same as `this->setValues(v)` in your case, and the error will be gone.

answered Nov 13 '14 at 7:54

SingerOfTheFall
16.4k 5 31 62

thank you very much i understand the concept. in order to advertise all i should use [this]. – user2957741 Nov 13 '14 at 7:59