Subject: MemFn() only offers void return types
Posted by Tom1 on Tue, 29 May 2018 09:32:42 GMT
View Forum Message <> Reply to Message

Hi,

It seems to me that MemFn() (and therefore THISFN()) do not work correctly with return types. For some reason MemFn() uses Event instead of Function and therefore only offers void return types.

Switching to using Function instead of Event fixes this behavior, but I'm not sure if it is correct fix or causes trouble elsewhere...:

```
template <class Ptr, class Class, class Res, class... ArgTypes>
Function<Res (ArgTypes...)> MemFn(Ptr object, Res (Class::*method)(ArgTypes...))
{
 return [=](ArgTypes... args) { return (object->*method)(args...); };
}
```

Update: The code for MemFn() can be found in Core/Function.h ...

Update 2: Testcase:

```
#include <Core/Core.h>

using namespace Upp;

// Temporary fix:

template <class Ptr, class Class, class Res, class... ArgTypes>
Function<Res (ArgTypes...)> MemFn2(Ptr object, Res (Class::*method)(ArgTypes...))
{
 return [=](ArgTypes... args) { return (object->*method)(args...); };
}

#define THISFN2(x)   MemFn2(this, &CLASSNAME::x)


// Testcase for MemFn

class MemFnTest{
 typedef MemFnTest CLASSNAME;

 String fn1(double a, double b, String &c){
  String s=Format("%f %f %s\r\n",a,b,~c);
  return s;
 }
```

```
public:
 Function<String (double,double,String&)> fn;

 MemFnTest(){
  fn=THISFN2(fn1);
  fn=THISFN(fn1);
 }
};

CONSOLE_APP_MAIN{
 MemFnTest x;
 String s = x.fn(1,2,String("Hi there!"));
 printf(s);
}
```

Best regards,

Tom

Subject: Re: MemFn() only offers void return types
Posted by mirek on Mon, 11 Jun 2018 13:53:08 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 29 May 2018 11:32Hi,

It seems to me that MemFn() (and therefore THISFN()) do not work correctly with return types. For some reason MemFn() uses Event instead of Function and therefore only offers void return types.

Switching to using Function instead of Event fixes this behavior, but I'm not sure if it is correct fix or causes trouble elsewhere...:

```
template <class Ptr, class Class, class Res, class... ArgTypes>
Function<Res (ArgTypes...)> MemFn(Ptr object, Res (Class::*method)(ArgTypes...))
{
 return [=](ArgTypes... args) { return (object->*method)(args...); };
}
```

Update: The code for MemFn() can be found in Core/Function.h ...

Update 2: Testcase:

```
#include <Core/Core.h>

using namespace Upp;

// Temporary fix:

template <class Ptr, class Class, class Res, class... ArgTypes>
Function<Res (ArgTypes...)> MemFn2(Ptr object, Res (Class::*method)(ArgTypes...))
{
 return [=](ArgTypes... args) { return (object->*method)(args...); };
}

#define THISFN2(x)   MemFn2(this, &CLASSNAME::x)


// Testcase for MemFn

class MemFnTest{
 typedef MemFnTest CLASSNAME;

 String fn1(double a, double b, String &c){
  String s=Format("%f %f %s\r\n",a,b,~c);
  return s;
 }

public:
 Function<String (double,double,String&)> fn;

 MemFnTest(){
  fn=THISFN2(fn1);
  fn=THISFN(fn1);
 }
};

CONSOLE_APP_MAIN{
 MemFnTest x;
 String s = x.fn(1,2,String("Hi there!"));
 printf(s);
}
```

Best regards,

Tom

You are absolutely right. Applied. Thank you.

Mirek
