

- Tutorial version 14 (September 23<sup>rd</sup>, 2007)
- for VBMcgi version 4.0,

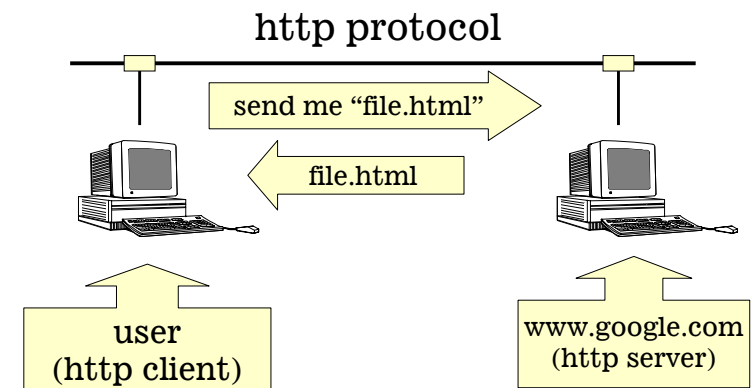
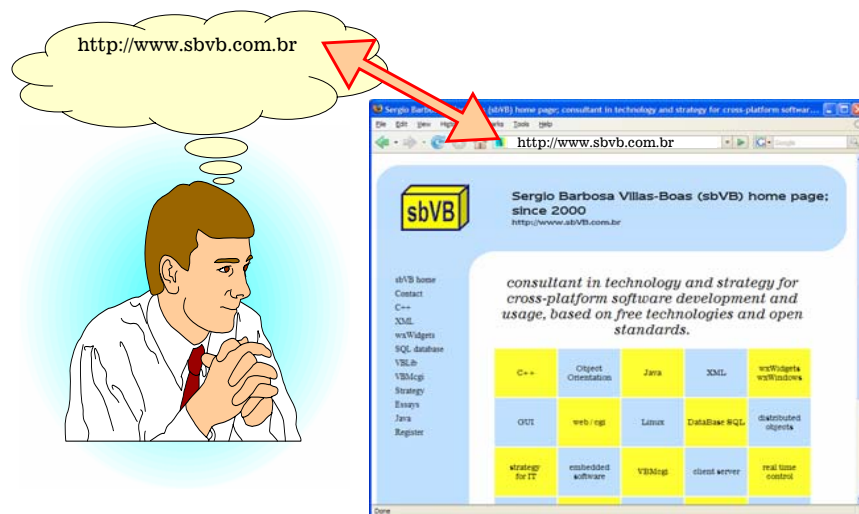
VBMcgi home  
<http://www.vbmcgi.org>



- By Sergio Barbosa Villas-Boas
  - [www.sbVB.com.br](http://www.sbVB.com.br)
  - [sbvb@sbvb.com.br](mailto:sbvb@sbvb.com.br)



- Web is the combined use of the following standards.
  - HTTP (Hyper Text Transfer Protocol)
  - URL (Universal Resource Locator)
  - HTML (Hyper Text Markup Language)
  - CGI (Common Gateway Interface)
- CGI is the interface that allows HTML pages to execute programs in the HTTP server
- VBMcgi is a free library in C++ for CGI development that allows 3-tier architecture. The isolation of tier 1 and tier 2 is achieved with 2 features (to be explained later).
  - string change feature
  - call function feature



## Hello html

```
<!-- hello.html -->
<html><body>
Some text in html
</body></html>
```

browser

http://.../hello.html

Some text in html

5

## Hyperlink html

```
<!-- a.html -->
<html><body>
Click <a href="b.html">here</a>
for more details.
</body></html>
```

```
<!-- b.html -->
<html><body>
More details. <p>
Click <a href="a.html">here</a>
to go back.
</body></html>
```

http://.../a.html  
Click [here](#)  
for more details.

http://.../b.html  
More details.  
Click [here](#)  
to go back.

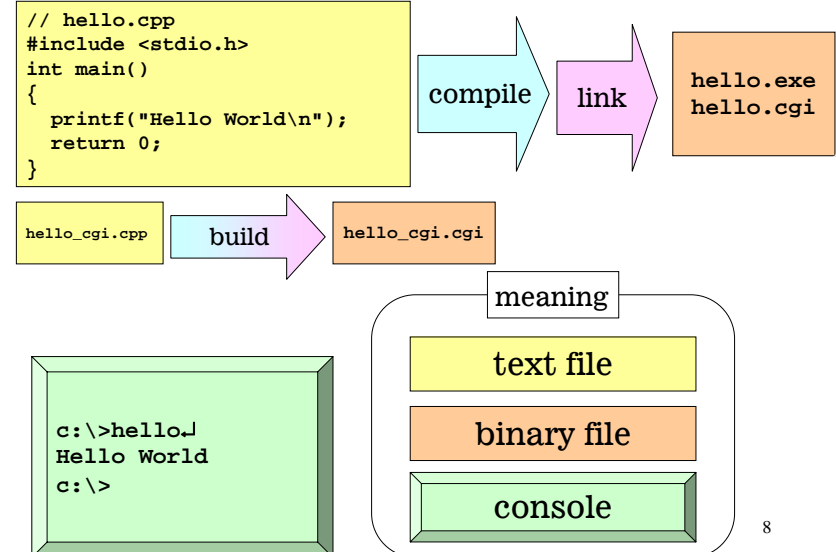
6

## The CGI interface

- CGI stands for “Common Gateway Interface”. It’s the interface that permits the user to start the execution of programs in the server from web pages (in html).
- The interface does not impose a computer language (any language can be used).
- Some computer languages for CGI are script, while others are compiled.
- Some of the most used languages for CGI are C/C++, Delphi, perl, ASP, PHP, JSP, Cold Fusion.
- We will be using the C++ language and the C++ library VBMcgi (Villas-Boas and Martins cgi - www.vbmcgi.org).
- In the VBMcgi home page, the extension \*.html has the same meaning as \*.cgi.

7

## Executable programs in command line interface



8

# Hello CGI

the http protocol requires this line

```
// hello_cgi.cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "Content-type:text/html" << endl << endl;
    cout << "<html><body>" << endl;
    cout << "Hello CGI" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```

```
c:\>hello_cgi.cgi
Content-type:text/html

<html><body>
Hello CGI
</body></html>
c:\>
```

http://.../hello\_cgi.cgi  
Hello CGI

# Example of calling a cgi program from web page in html

```
<!-- call_cgi.html -->
<html><body>
Click <a href="hello_cgi.cgi">here</a>
to call a cgi program directly
</body></html>
```

http://.../call\_cgi.html

Click [here](#) to call a cgi program directly

http://.../hello\_cgi.cgi

Some text in html

# Example of cgi calling from form

```
<!-- call_cgi_2.html -->
<html><body>
Example of cgi: <p>
<form action="hello_cgi.cgi" method="get">
<input type="submit" value="Send Data">
</form>
</body></html>
```

http://.../call\_cgi\_2.html

Example of cgi:

Send Data

http://.../hello\_cgi.cgi

Some text in html

# Multiply table

```
c:\>multiply_table.cgi
Content-type:text/html

<html><body>
5<br>
10<br>
15<br>
20<br>
25<br>
30<br>
35<br>
40<br>
45<br>
50<br>
</body></html>
c:\>
```

```
// multiply_table.cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "Content-type:text/html" << endl << endl;
    cout << "<html><body>" << endl;
    int multiply_table = 5;
    int i;
    for (int i=1 ; i <= 10 ; i++)
        cout << i*multiply_table << "<br>" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```

5  
10  
15  
20  
25  
30  
35  
40  
45  
50

# CGI programs and the environment

```

// environment.cpp
#include <iostream> // cout
using namespace std;
#include <stdlib.h> // getenv
void printEnv(const char *var)
{
    char *t = getenv(var);
    if (!t) t=""; // never reference a pointer to zero
    cout << "Environment[" << var
        << "] = \"" << t << "\"<br>" << endl;
}

int main()
{
    cout << "Content-type: text/html" << endl << endl;
    printEnv("SERVER_SOFTWARE");
    printEnv("SERVER_NAME");
    printEnv("SERVER_PORT");
    printEnv("REQUEST_METHOD");
    printEnv("SCRIPT_NAME");
    printEnv("REMOTE_ADDR");
    printEnv("HTTP_USER_AGENT");
    return 0;
}
    
```

# CGI programs and the environment (2)

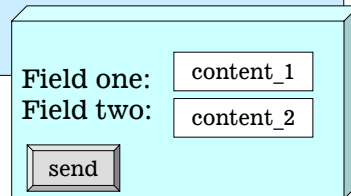
```

Environment[SERVER_SOFTWARE] = "Apache/1.3.12 Ben-SSL/1.41 (Unix) tomcat/1.0"
Environment[SERVER_NAME] = "vbmcgi.org"
Environment[SERVER_PORT] = "80"
Environment[REQUEST_METHOD] = "GET"
Environment[SCRIPT_NAME] = "/getenv.cgi"
Environment[REMOTE_ADDR] = "146.164.50.16"
Environment[HTTP_USER_AGENT] = "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
    
```

# QUERY\_STRING

```

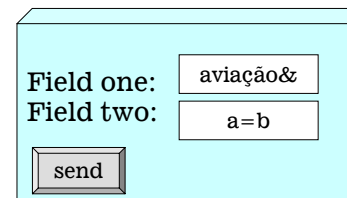
<!-- test_getenv.html -->
<HTML><body>
<form action="mycgi.cgi" method="get">
Field one: <input type="text" name="field_1" size="10"><br>
Field two: <input type="text" name="field_2" size="10"><br>
<input type="submit" value="send">
</form>
</body></HTML>
    
```



http://.../mycgi.cgi?field\_1=content\_1&field\_2=content\_2

QUERY\_STRING

# QUERY\_STRING and escape sequence



http://.../mycgi.cgi?field\_1=avia%26&field\_2=a%3Db





## Escape sequence and VBString

```
#include "VBLib.h"
using namespace br::com::sbVB::VBLib;
int main()
{
    VBString a;
    a = "avição&";
    a.escapeSequence();
    VBAssert(a=="avia%e7%e3o%26");
    a.escapeSequenceReverse();
    VBAssert(a=="avição&");
    return 0;
}
```

17



## History of VBMcgi

- Started as a DEL project, by A. Martins and supervised by sbVB.
  - VBMcgi is based on VBLib.
- Versions
  - November 2000 – Version 1.0
  - January 2002 – Version 2.0
  - September 2003 – Version 3.0
  - September 2007 – Version 4.0
- License LGPL (free for any use)
- Discussion group
  - [vbmcgi@yahoogroups.com](mailto:vbmcgi@yahoogroups.com)
  - <http://groups.yahoo.com/group/vbmcgi/>

18



## Installing VBMcgi in Visual C++ 6.0 SP5

- Get distribution file VBMcgi\_win\_30.zip
- Open the workspace VBMcgi.dsw
- command Build – Batch build
- Run hello\_vbmcgi.cpp as an ordinary console application.
- Have Apache for Windows installed to your computer.
  - Alter httpd.conf to allow binary cgi to run.
  - Place the CGI program in the directory Apache wants it to be (usually cgi-bin).
  - Check that the hello is running in [http://localhost/cgi-bin/hello\\_vbmcgi.cgi](http://localhost/cgi-bin/hello_vbmcgi.cgi)

19



## Installing VBMcgi in unix, and gnu g++ 3.2

- Get distribution file VBMcgi\_unix\_30.zip. Open it in the computer you want to run the HTTP server.
- Turn make.sh executable, and execute it. See the VBMcgi be built.
- Run hello\_vbmcgi.cpp as an ordinary console application.
- Have Apache installed to some computer (may be the one you're using)
  - Assure that httpd.conf is properly configured to allow binary CGI to run.
  - Place the CGI program in the directory Apache wants it to be (usually cgi-bin).
  - Check that the hello is running in [http://somewhere.com/cgi-bin/hello\\_vbmcgi.cgi](http://somewhere.com/cgi-bin/hello_vbmcgi.cgi)

20



## Main advantages of using VBMcgi

- Keep / cultivate culture of using C++.
  - Keep using C++ and don't miss the web/CGI development!
- 3-tier software architecture
  - No programming in the presentation tier !
  - Develop highly maintainable software !
- System in binary format
  - Customer can't easily reverse engineer
  - This protects the developer's intellectual property
- Executable cgi's don't require installation of script interpreter software to the server
  - Can be used in any computer, even commercial host computers that might not have all software installed.

21



## Hello VBMcgi

- Useful to see if VBMcgi can be compiled properly

```
// hello_VBMcgi.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBLib;
using namespace br::com::sbVB::VBMcgi;
using namespace std;
int main ()
{
    VBMcgi cgi;
    cgi.httpCompleteHeader();
    cout << "<html><body>" << endl;
    cout << "Hello VBMcgi" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```

```
c:\>hello_vbmcgi.cgi
Content-type:text/html

<html><body>
Some text in html
</body></html>
c:\>
```

Hello VBMcgi

22



## Decoding HTML form

- To decode the form, VBMcgi has the method "formDecode". It reads the QUERY\_STRING, separate separates the form names and its contents, reverses the escape sequence, and place the result to an cgi's inner attribute, which is a linked list (of a pair of strings).
- formDecode encapsulates difference of get and post method.
- To easily allow access to the form variables, there's the method "getVarContent" (that should be called after the method "formDecode").

23



## Multiply table with parameter

multiply table with parameter CGI example:

Multiply table of  ← type in "8" in this field

.../multiply\_table\_with\_parameter.cgi?mtable=8

```
8
16
24
32
40
48
56
64
72
80
```

24

## Multiply table with parameter (2)

```
<form action="multiply_table_with_parameter.cgi"
method="get">
Multiply table of <input type="text" name="mtable"
size="3" value="5"><br>
<input type="submit" value="send">
</form>
```

```
// multiply_table_with_parameter.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBLib;
using namespace br::com::sbVB::VBMcgi;
using namespace std;
int main() {
    VBMcgi cgi;
    cgi.httpCompleteHeader();
    cgi.formDecode(); // decode form vars
    cout << "<html><body>" << endl;
    VBString mt_str = cgi.getVarContent("mtable"); // get mtable as string
    int multiply_table = atoi(mt_str); // convert to int
    for (int i=1 ; i <= 10 ; i++)
        cout << i*multiply_table << "<br>" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```

## String parameters

```
<form action="string_parameters.cgi"
method="get">
<table border="0">
<tr>
<td align="right">Name:</td>
<td ><input type="text" name="s_name"
size="20"></td>
</tr>
<tr>
<td align="right">Telephone:</td>
<td ><input type="text"
name="s_telephone" size="20"></td>
</tr>
<tr>
<td align="right">Zip code:</td>
<td ><input type="text" name="s_zip" size="20"></td>
</tr>
<tr>
<td align="right">email:</td>
<td ><input type="text" name="s_email" size="20"></td>
</tr>
</table>
<input type="submit" value="send">
</form>
```

Name:	<input type="text" value="sbVB"/>
Telephone:	<input type="text" value="2233-4455"/>
Zip code:	<input type="text" value="20.234-140"/>
email:	<input type="text" value="sbVB@sbVB.com.br"/>
<input type="button" value="send"/>	

// string\_parameters.cgi

## String parameters (2)

[http://www.vbmcgi.org/string\\_parameter.cgi?s\\_name=sbVB&s\\_telephone=2222-3344&s\\_zip=20.234-140&s\\_email=sbVB@sbVB.com.br](http://www.vbmcgi.org/string_parameter.cgi?s_name=sbVB&s_telephone=2222-3344&s_zip=20.234-140&s_email=sbVB@sbVB.com.br)

URL  
Name is: sbVB  
Telephone is: 2233-4455  
ZIP is: 20.234-140  
email is: sbVB@sbVB.com.br

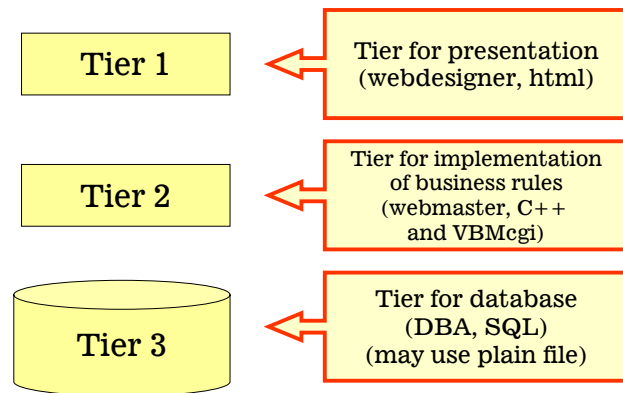
```
// string_parameters.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBLib;
using namespace br::com::sbVB::VBMcgi;
using namespace std;
int main() {
    VBMcgi cgi;
    cgi.httpCompleteHeader();
    cgi.formDecode();
    VBString str_name = cgi.getVarContent("s_name");
    VBString str_telephone = cgi.getVarContent("s_telephone");
    VBString str_zip = cgi.getVarContent("s_zip");
    VBString str_email = cgi.getVarContent("s_email");

    cout << "<html><body>" << endl;
    cout << "Your name is: <b>" << str_name << "</b><br>" << endl;
    cout << "Your telephone is: <b>" << str_telephone << "</b><br>" << endl;
    cout << "Your zip code is: <b>" << str_zip << "</b><br>" << endl;
    cout << "Your email is: <b>" << str_email << "</b><br>" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```

## 3-tier web software architecture: highly maintainable software

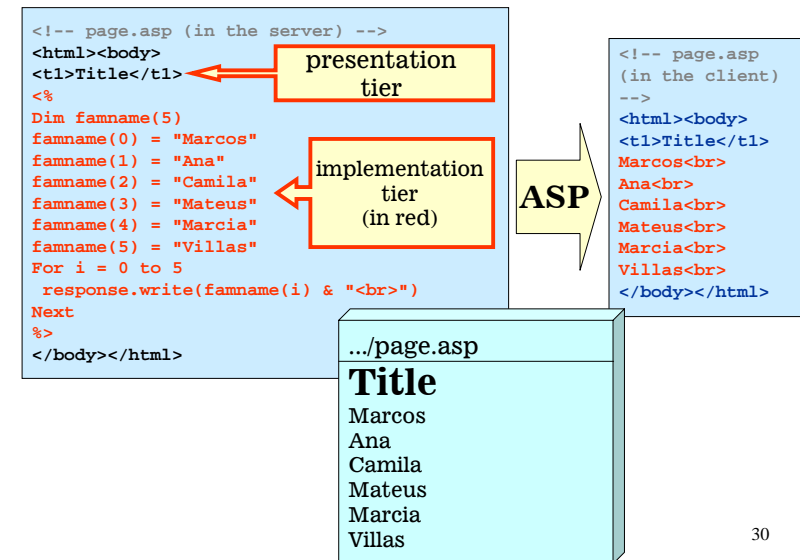
- **Webdesigner (tier for presentation)**
  - Cares about the site's design. Knows well HTML, DHTML, JavaScript. Uses tools like DreamWeaver and FrontPage.
- **Webmaster (tier for implementation of business rules)**
  - Cares about the implementation of the site's interactivity. Knows about programming and often about data base. Uses languages like C++ (and VBMcgi), PHP, ASP, JSP, Perl, etc.
- **DBA (tier for database)**
  - Cares about data modelling. Knows about SQL, relational databases, etc. Uses software like Oracle, SQL server, MySQL, DB2, PostgreSQL, etc.

## 3-tier web software architecture (2)



29

## Mixed tier web software: poorly maintainable software



30

## The VBMcgi golden rule

- To be productive using C++ for web software, you should

To isolate the work of webdesigner (tier 1, for presentation) and webmaster (tier 2, for implementation of business rules)

- This is implemented with 2 main features
  - string change feature
  - call function feature

31

## String change feature

- Register string pairs (“search”, “replace”) using method “addBySource”.
- Use the “out” method, that receives an HTML file as argument, to put HTML data to the console.
- This method will change all “search” occurrences to respective “replace” ones.
- The search string should be improbable char sequences to avoid unwanted changes (example of good char sequences for search string: s\_name, s\_telephone).
- The registered search strings should be left-unique (example of non-left-unique strings: s\_client, s\_clientTelephone).

32



www.sbVB.com.br

## String change feature (2)

```
// string_change_feature.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBLib;
using namespace br::com::sbVB::VBMcgi;
using namespace std;
int main () {
    VBMcgi cgi;

    // add some string exchanges
    cgi.addBySource("s_name","Sergio Barbosa Villas-Boas");
    cgi.addBySource("s_telephone","2233-4455");
    cgi.addBySource("s_zip","20.234-140");
    cgi.addBySource("s_email","sbvb@sbvb.com.br");

    // open html file, execute string changes and put to browser
    cgi.out("string_change_feature_out.html");
    return 0;
}
```

Name is: Sergio Barbosa Villas-Boas  
 Telephone is: 2233-4455  
 ZIP is: 20.234-140  
 email is: sbvb@sbvb.com.br

```
<!-- string_change_feature_out.html -->
Name is: <b>s_name</b><br>
Telephone is: <b>s_telephone</b><br>
ZIP is: <b>s_zip</b><br>
email is: <b>s_email</b><p>
```

33



www.sbVB.com.br

## String change feature (3)

string change feature

http://.../string\_change\_feature\_out.html

Here is a person from our data base  
 Name is: s\_name  
 Telephone is: s\_telephone  
 ZIP is: s\_zip  
 email is: s\_email  
 Thanks for using our system

Click [here](#) to see person from database

http://.../string\_change\_feature.cgi

Here is a person from our data base  
 Name is: Sergio Barbosa Villas-Boas  
 Telephone is: 2233-4455  
 ZIP is: 20.234-140  
 email is: sbvb@sbvb.com.br  
 Thanks for using our system



www.sbVB.com.br

## formDecode adds variables

- Other than decode form variables, the “formDecode” method can automatically add variables to the exchange string list.
  - To do that, simply use argument “true” in this method.
  - Thus feature might be an easy way to make a cgi program that uses string changes from the form.

35



www.sbVB.com.br

## formDecode adds variables (2)

multiply table with parameter CGI example:  
 <form action="formdecode.cgi" method="get">  
 ... (the same form as before)  
 <input type="submit" value="send">  
 </form>

Name:   
 Telephone:   
 Zip code:   
 email:

send

```
// formdecode.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBLib;
using namespace br::com::sbVB::VBMcgi;
using namespace std;
int main() {
    VBMcgi cgi;
    cgi.formDecode(true); // decode form variables

    // open html file, execute string changes and put to browser
    cgi.out("string_change_feature_out.html");
    return 0;
}
```

36

## formdecode adds variables (3)

http://.../person.html

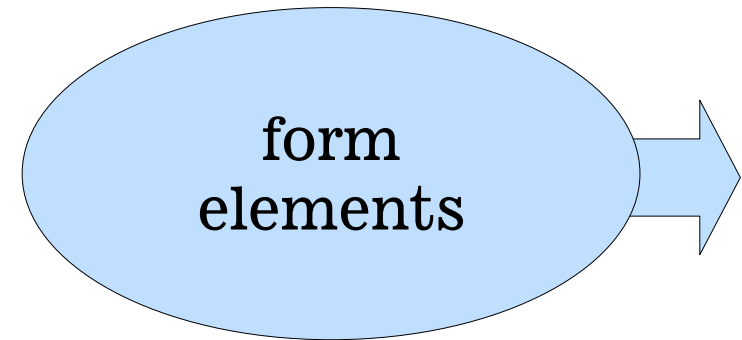
Name is: s\_name  
 Telephone is: s\_telephone  
 ZIP is: s\_zip  
 email is: s\_email

http://.../formdecode.cgi&(query\_string)

Name is: sbVB  
 Telephone is: 2233-4455  
 ZIP is: 20.234-140  
 email is: sbvb@sbvb.com.br

37

## form elements



38

## Form reference – textbox

**textbox:**  
 Name:

```
<html><body>
textbox:<br>
<form action="formdecode.cgi" method="get">
Name:<input type="text" name="fieldname" size="20"><br>
<input type="submit" value="send">
</form>
</body></html>
```

```
// formdecode.cpp
#include "VBMcgi.h"
int main() {
  VBMcgi cgi;
  cgi.formDecode(); // decode form variables
  VBString str = cgi.getVarContent("fieldname");
  // continue cgi code
}
```

39

## Form reference – textarea

**textarea (scrolling text box):**  
 Enter your opinion about something:

```
<html><body>
textarea (scrolling text box):<br>
<form action="formdecode.cgi" method="get">
Enter your opinion about something:<br>
<textarea name="opinion" rows="4" cols="55">Initial value</textarea>
<input type="submit" value="send">
</form>
</body></html>
```

```
// prologue
int main() {
  VBMcgi cgi;
  cgi.formDecode(); // decode form variables
  VBString str = cgi.getVarContent("opinion");
  // epilogue
}
```

40

## Form reference – checkbox

### checkbox:

You like:

- Sports
- Books
- News

```
<html><body>
checkbox:<br>
<form action="formdecode.cgi" method="get">
You like:<br>
<input type="checkbox" name="like_sports"
  checked value="ON">Sports<br>
<input type="checkbox" name="like_books"
  checked value="ON">Books<br>
<input type="checkbox" name="like_news"
  checked value="ON">News<br>
<input type="submit" value="send">
</form>
</body></html>
```

```
// prologue
int main() {
  VBMcgi cgi;
  cgi.formDecode(); // decode form variables
  bool b_sports = cgi.getCheckBox("like_sports", "ON");
  bool b_books = cgi.getCheckBox("like_books", "ON");
  bool b_news = cgi.getCheckBox("like_news", "ON");
  // epilogue
}
```

41

## Form reference – radio button

### radio button:

After dinner you:

- Brush your teeth
- Watch TV
- Study
- Read a book

```
// prologue
int main()
{
  VBMcgi cgi;
  cgi.formDecode(); // decode form variables
  VBString n_str = cgi.getVarContent
    ("after_dinner");
  int n = atoi(n_str);
  const char *stringsAfterDinner[4] = {
    "brush the teeth",
    "give TV a good watch",
    "go study for a while",
    "read the favorite book"
  };
  // continue cgi code
  cgi.httpCompleteHeader();
  cout << stringsAfterDinner[n-1]
    << endl;
  return 0;
}
```

```
<html><body>
radio button:<br>
<form action="formdecode.cgi" method="get">
After dinner you:<br>
<input type="radio" name="after_dinner"
  value="1">Brush your teeth<br>
<input type="radio" name="after_dinner"
  value="2">Watch TV<br>
<input type="radio" name="after_dinner"
  value="3" checked>Study<br>
<input type="radio" name="after_dinner"
  value="4">Read a book
</form>
</body></html>
```

## Form reference – drop down

```
<html><body>drow down:<br>
<form action="formdecode.cgi" method="get">
Your favorite color is: <br>
<select name="favorite_color" size="1">
<option value="1">Green</option>
<option value="2">Red</option>
<option value="3">Blue</option>
</select></form></body></html>
```

```
// prologue
int main() {
  VBMcgi cgi;
  cgi.formDecode(); // decode form variables
  VBString n_str = cgi.getVarContent("favorite_color");
  int n = atoi(n_str);
  const char *stringsForColor[3] = {
    "the color green",
    "red, a hot color",
    "blue, the color of the sky"
  };
  // continue cgi code
  cgi.httpCompleteHeader();
  cout << stringsForColor[n-1] << endl;
  return 0;
}
```

### drop down:

Your favorite color is:

Green  
Green  
Red  
Blue

## call function feature

call function  
feature

## Call function feature

- Allow calling functions inside a web page. It's like SSI (Server Side Includes), but gives complete control to the webmaster, and does not require changes to the web server configuration.
- The programmer creates his user callback function, having necessarily the prototype as shown below. This prototype must be used even if the function does not use any of the parameters.  

```
void userFunction(VBMcgi & cgi, void *p);
```
- Use the method "addFunction" of the vbmcgi class, to register the function in the VBMcgi object.
- Optionally, if using dynamic link (\*.dll in Windows or \*.so in Unix), the webmaster can use the same function in more than one CGI program. By updating the dll or so, all CGIs will be affected, without need to recompile.

45

## Call function feature, example

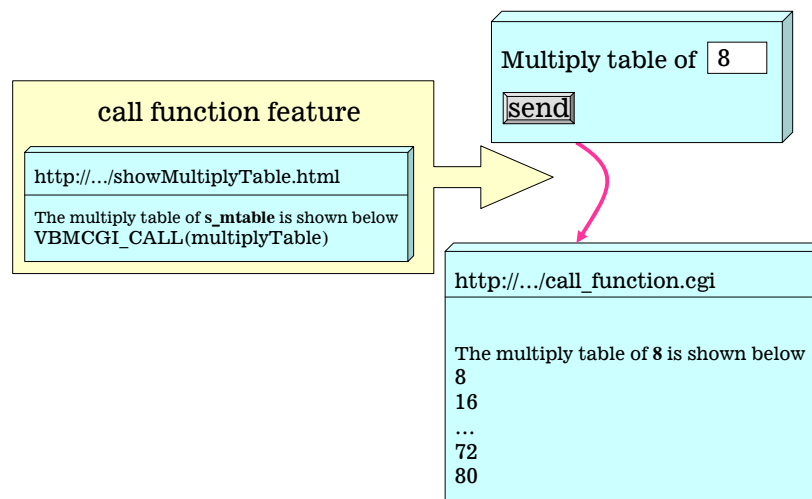
Multiply table of

```
<form action="call_function.cgi" method="get">
Multiply table of
<input type="text" name="s_mtable"><br>
<input type="submit" value="send">
</form>
```

```
// call_function.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VLib;
using namespace std;
void multiplyTable(VBMcgi & userCgi, void *p) {
    // get s_mtable as string
    VBString mt_str = userCgi.getVarContent("s_mtable");
    int multiply_table = atoi(mt_str); // convert to int
    for (int i=1 ; i <= 10 ; i++)
        cout << i*multiply_table << "<br>" << endl;
}
int main()
{
    VBMcgi cgi;
    // decode form vars and add to "string change feature"
    cgi.formDecode(true);
    cgi.addFunction(multiplyTable);
    cgi.out("showMultiplyTable.html");
    return 0;
}
```

46

## Call function feature (3)



47

## Call function feature passing parameters from C++

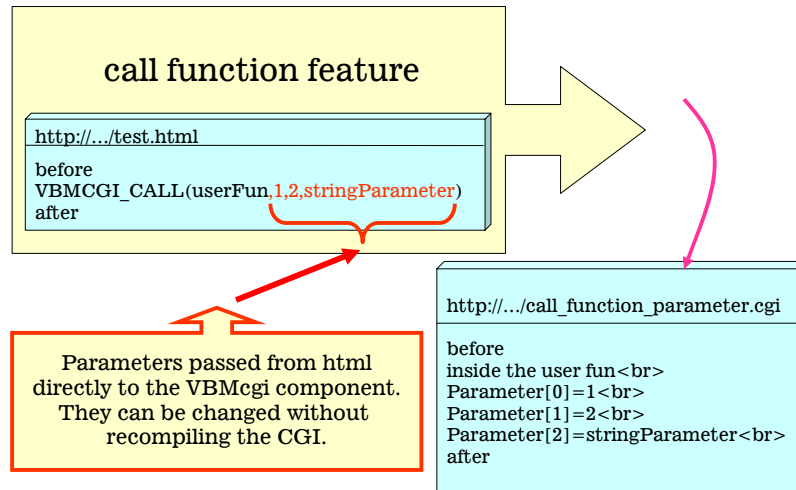
call function feature (passing parameters)

```
http://.../showMultiplyTable.html
The multiply table of s_mtable is shown below
VBMCgi_CALL(multiplyTable)
```

Multiply table of

```
http://.../call_function.cgi
The multiply table of 5 is shown below
-----
The value of i is 4
The value of d is 2.2
The value of str is abc
-----
5
10
15
...
45
50
```

## Call function feature passing parameters from HTML



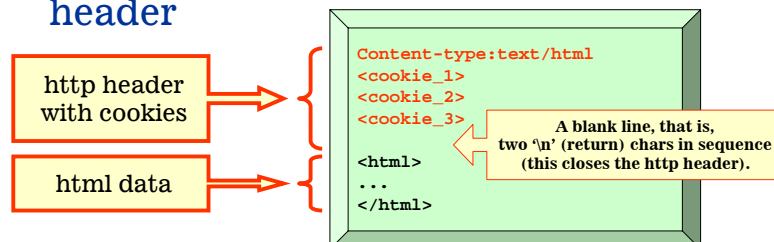
49

## Call function feature passing parameters from HTML (2)

```
// call_function_with_parameter_html.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
void userFun(VBMcgi & cgi, void *p)
{
    cout << "inside the user fun<br>" << endl;
    unsigned number_of_parameters = cgi.getHtmlParameterNumber();
    for (unsigned i = 0 ; i < number_of_parameters ; i++)
    {
        cout << "Parameter[" << i << "]=" <<
            cgi.getHtmlParameter(i) << "<br>";
        if (i < number_of_parameters-1) cout << endl;
    }
}
int main()
{
    VBMcgi cgi;
    cgi.addFunction(userFun);
    cgi.out("test.html");
    return 0;
}
```

## Cookies

- Types of cookie
  - end-of-session
  - persistent
- Cookies are sent inside the http header



51

## Example of cgi and 3 cookies

```
// cookies_3.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
int main()
{
    VBMcgi cgi;
    cgi.httpHeader(); // begin the http header
    // set parameters of cookie 1
    cgi.setCookieNameValue("cookie name 1", "cookie value 1");
    cgi.setCookieExpires("end-of-session");
    cgi.sendCookie(); // send cookie to browser
    // set parameters of cookie 2
    cgi.setCookieNameValue("cookie name 2", "cookie value 2");
    cgi.setCookieExpires("end-of-session");
    cgi.sendCookie(); // send cookie to browser
    // set parameters of cookie 3
    cgi.setCookieNameValue("cookie name 3", "cookie value 3");
    cgi.setCookieExpires("end-of-session");
    cgi.sendCookie(); // send cookie to browser
    cout << endl; // end the http header
    cgi.out("test.html"); // html code to the console
    return 0;
}
```

## Example of cgi and 3 cookies (2)

```
Content-type:text/html
Set-Cookie: cookie name 1=cookie value 1;expires=
Set-Cookie: cookie name 2=cookie value 2;expires=
Set-Cookie: cookie name 3=cookie value 3;expires=

<html>
test.html
</html>
```

53

## "Seeing the cookie" and browser configuration

- Most often, the browser configuration is set to accept cookies without any prompt or warning.
- If you want to see the cookies (to learn, to trace, or for curiosity), you must have your browser set to prompt for cookies.

54

## Persistent cookies

```
// persistent_cookie.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
int main()
{
    VBMcgi cgi;
    cgi.httpHeader(); // begin the http header
    // set parameters of cookie
    cgi.setCookieNameValue("cookie name", "cookie value");
    cgi.setCookieExpires("31-Mar-2002 23:59:59 GMT");
    cgi.sendCookie(); // send cookie to browser
    cout << endl; // end the http header
    cgi.out("test.html"); // html code to the console
    return 0;
}
```

```
Content-type:text/html
Set-Cookie: cookie name 1=cookie value 1;expires=31-Mar-2002 23:59:59 GMT

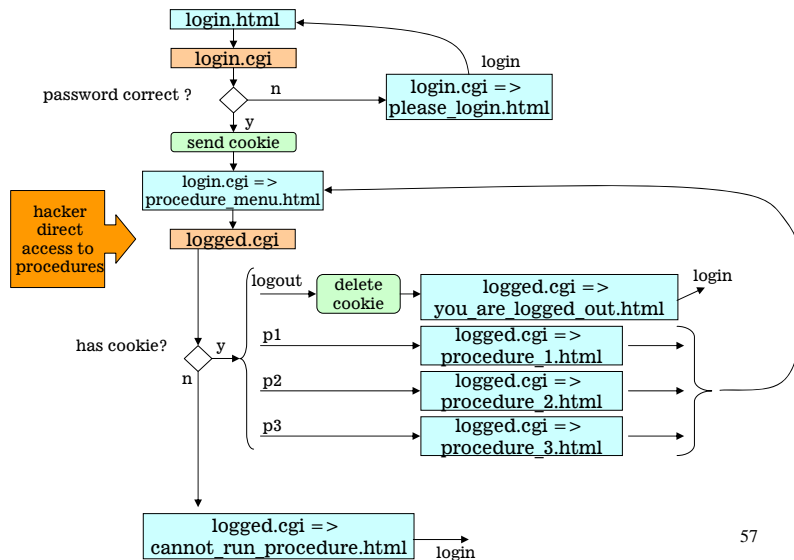
<html>
test.html
</html>
```

## Login to web using cookies

- Using cookies, one can define “state” for web software.
- By doing so, it’s possible to “login” to software.
- As a general rule, no understandable information should be saved in the client’s cookie.

56

# Login to web using cookies (2)



57

# gif or jpeg output Non HTML data in CGI programs

- The CGI program does not return HTML data, but a gif or jpg data.

```
#include <fstream>
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
int main()
{
    VBMcgi cgi;
    const char *file = "vbmcgi_logo.gif";
    ifstream gifFile(file);
    if (!gifFile) {
        cgi.httpCompleteHeader();
        cout << "Could not open gif file for reading" << endl;
        exit(1);
    }
    cgi.httpCompleteHeader("image/gif");
    unsigned char z;
    while (true) {
        gifFile.read((char*)&z, sizeof(char));
        if (gifFile.eof()) break;
        // copy binary to console (cout)
        cout.write((char*)&z, sizeof(char));
    }
    return 0;
}
```

58

# Redirection

```
// redirect.cpp
int main()
{
    VBMcgi cgi;
    cgi.redirect("http://www.sbVB.com.br");
    return 0;
}
```

```
status: 301
location: http://www.sbVB.com.br
```

http://www.sbVB.com.br

...

# Redirection (2)

```
Redirect to:
<form method="get" action="redirect.cgi">
<input type="text" name="redirect"
value="http://www.cplusplus.com/" size="80">
<input type="submit" value="Submit">
```

```
// redirect.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
int main()
{
    VBMcgi cgi;
    cgi.formDecode();
    VBString url = cgi.getVarContent("redirect");
    cgi.redirect(url);
    return 0;
}
```

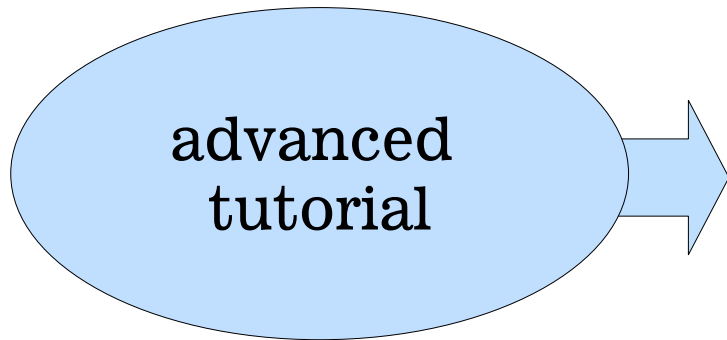
Redirect to:



http://www.sbVB.com.br

...

60



http://.../page\_count\_data.html

This page was accessed #pageCount times

```
// page_count.cpp
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
int main () {
    VBMcgi cgi;
    VBPageCount myPageCount("page_count");
    cgi.addBySource("#pageCount",myPageCount.getCount());
    cgi.out("page_count_data.html");
    return 0;
}
```

http://.../page\_count.cgi

This page was accessed 12 times

```
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
const char *comment="My friend's birthday";
void myCalendar(VBMcgi & cgi, void *p)
{
    VBTableMonth a;
    int day = 31;
    int year = 2002;
    int month = VBmar; // VBjan = 0, VBfeb = 1, VBmar = 2, ...
    a.setDate(year,month);
    a.setDayContent(day,comment);
    a.htmlOut();
}

int main()
{
    VBMcgi cgi;
    cgi.addFunction(myCalendar);
    cgi.out("table_month_data.html");
    return 0;
}
```

call function feature

http://.../table\_month\_data.html

before  
VBMCGI\_CALL(myCalendar)  
after

http://.../table\_month\_data.cgi

before

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						
comment						

after

## All CGI architecture

- The “all CGI” is an architecture recommended for your web software system.
- All accesses to the site should be CGI. Avoid using direct links to \*.html. By doing so, it is possible to use dynamic elements in the pages the user sees.
- Some dynamic elements are shown below
  - Access to data base or file system
  - Page count
  - Dynamic menu
  - Calendar
  - User made elements
- If your CGI program can only run in the cgi-bin directory, place a PHP redirect file index.php. See the example below.

```
<?php header("location:http://www.sbvb.com.br/cgi-bin/index.cgi?p=0"); ?>
```

65

## All cgi architecture (2) example of index.cpp

```
#include "VBMcgi.h"
using namespace br::com::sbVB::VBMcgi;
using namespace br::com::sbVB::VBLib;
using namespace std;
// i is passed as reference.
// This function can change its value
VBString getHtmlFileName(unsigned & i) {
    static const char*fileNames[] = {
        "index_original.html" // 0
        // more files
    };
    unsigned size = sizeof(fileNames)/sizeof(const char*);
    // don't let access outside the valid fileNames
    if (i >= size) i = 0;
    return fileNames[i];
}
int main() {
    VBMcgi cgi;
    cgi.formDecode();
    unsigned linkNumber = atoi(cgi.getVarContent("p"));
    VBString link = getHtmlFileName(linkNumber);
    switch (linkNumber)
    {
        case 0: // index_original.html
            VBPageCount myPageCount("index");
            cgi.addBySource("s_indexCount", myPageCount.getCount());
            break;
    }
    cgi.out(link);
    return 0;
}
```

A function to return the string of html file names

The number of the html file name is passed to the cgi by the query string

Special treatment to some pages

66

## Integrating database with web software

- If you are using VBMcgi and web software, it is natural that you use database through C++.
- This is easy to do, and the explanation is in a specific database tutorial.

67