Web 编程

什么是 CGI ？

* 通用网关接口（Common Gateway Interface），简称 CGI，是一组定义如何将信息在 Web 服务器和一个自定义脚本之间交换的标准。
* 目前 CGI 的规范是由 NCSA 维护的；NCSA 定义 CGI 如下：
* 通用网关接口（CGI），是外部网关程序与信息服务器（例如，HTTP 服务器）之间的接口标准。
* 目前应用的版本是 CGI /1.1；CGI/1.2 版本正在开发中。

网页浏览

要了解 CGI 的概念，那我们先来看看，当我们点击超链接来浏览特定的网页或 URL 时会发生什么。

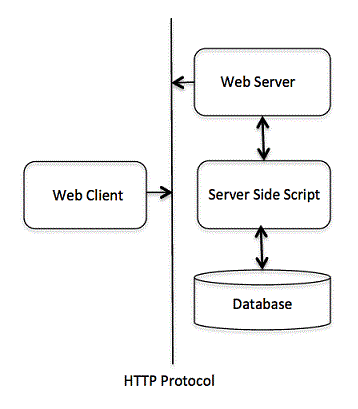
* 您的浏览器连接到 HTTP Web 服务器，并且请求 URL ，文件名。
* Web 服务器将会解析 URL，并且会查找文件名。如果找到所请求的文件，那么 Web 服务器会将该文件发送给浏览器，否则它会发送一个错误信息表明您请求了一个错误的文件。
* Web 浏览器的响应来自于 Web 服务器，并且浏览器要显示所接收到的文件或者错误消息的响应。

然而，也可以以这样的方式来设置 HTTP 服务器：每当在某个目录中的文件被请求时，该文件并不被发送回；代替的是，这个请求作为一个程序来被执行，由该程序产生输出，并被发送回给浏览器来显示。

公共网关接口（CGI）是一个能使应用程序（称为 CGI 程序或 CGI 脚本）与 Web 服务器和客户端进行交互的标准协议。这些 CGI 程序可以使用 Python，Perl，Shell，C 或 C++ 等语言来编写。

CGI 架构图

下图简单的展示了CGI程序的一个简单架构:



Web 服务器配置

在你进行 CGI 编程之前确保您的 Web 服务器支持 CGI，并且它被配置为能够处理 CGI 程序。 HTTP 服务器执行的所有 CGI 程序被保存在一个预先配置的目录里。这个目录称为 CGI 目录，一般其命名为 /var/www/cgi-bin。尽管 CGI 文件是 C++ 可执行的，但是按照惯例他们会有扩展名 **.cgi**.

默认情况下，Apache Web 服务器配置为在/var/www/cgi-bin运行 CGI 程序。如果你想指定其他目录运行 CGI 脚本，你可以在 httpd.conf 配置文件中修改下面部分的内容:

<Directory "/var/www/cgi-bin">

**AllowOverride** None

**Options** ExecCGI

**Order** allow,deny

**Allow** from all

</Directory>

<Directory "/var/www/cgi-bin">

**Options** All

</Directory>

在这里，我们假定你有 Web 服务器，启动并运行成功, 并且你可以运行任何其他语言（比如 Perl 或 Shell 等）所写的 CGI 程序。

第一个 CGI 程序

思考下面的 C++ 程序内容:

**#include <iostream>**

**using** **namespace** std;

**int** **main** ()

{

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Hello World - First CGI Program</title>\n";

cout << "</head>\n";

cout << "<body>\n";

cout << "<h2>Hello World! This is my first CGI program</h2>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

编译上面的代码，并命名可执行文件为 cplusplus.cgi. 该文件被保存在/var/WWW/cgi-bin目录里，并且它有上面的内容。在运行 CGI 程序之前，请确保你已使用 **chmod755 cplusplus.cgi** UNIX 命令来改变文件的权限，使文件能够可执行。现在，如果你点击 [cplusplus.cgi](http://www.tutorialspoint.com/cgi-bin/cplusplus.cgi) ，那么这将产生以下的输出：

**Hello World! This is my first CGI program**

以上的 C++ 程序是一个简单的将输出写入 STDOUT 文件（即屏幕）的程序。它有一个可用的重要和额外的功能是，第一行要输出 **Content-type:text/html\r\n\r\n**. 这一行被发送回浏览器，并指定在浏览器屏幕上显示出来的内容类型。 现在，你理解了 CGI 的基本概念，然后你就可以使用 Python 写许多复杂的 CGI 程序。一个 C++ 的 CGI 程序可以与任何其他外部系统(例如 RDBMS )交互，从而来进行信息的交换。

HTTP 报头

**Content-type:text/html\r\n\r\n** 是发送到浏览器的 HTTP 报头的一部分，它用来帮助浏览器解析文本内容。下面的表格展示了 HTTP 报头

**HTTP** Field Name: Field Content

For Example

Content-type: text/html\r\n\r\n

还有其他一些你可能经常会在 CGI 编程中使用的重要HTTP报头。

|  |  |
| --- | --- |
| **报头** | **描述** |
| Content-type: | 一个MIME字符串格式的文件会被返回。例如：Content-type:text/htm |
| Expires: Date | Date：页面信息失效的日期。这运行在浏览器端，由浏览器决定何时需要刷新页面。一个有效的日期字符串格式应该是这样的： 01 Jan 1998 12:00:00 GMT. |
| Location: URL | 这里的 URL 是应当返回的 URL，而不是请求的 URL.你可以使用此来重定向一个请求到任何文件。 |
| Last-modified: Date | 资源最近修改的日期。 |
| Content-length: N | 返回数据的长度，以字节为单位存储。浏览器使用这个值来预估计文件的下载时间。 |
| Set-Cookie: String | 通过字符串 string 设置 cookie |

CGI 环境变量

所有 CGI 程序可使用下列环境变量。这些变量在编写 CGI 程序时，发挥着重要的作用。

|  |  |
| --- | --- |
| **变量名** | **描述** |
| CONTENT\_TYPE | 内容的数据类型。当客户端发送连接内容到服务器时使用。例如文件上传等。 |
| CONTENT\_LENGTH | 查询信息的长度。它仅适用于 POST 请求。 |
| HTTP\_COOKIE | 以键值对的形式返回设置的 cookies 。 |
| HTTP\_USER\_AGENT | 用户代理请求头字段包含发起请求的用户代理的有关信息，包括 Web 浏览器的名称。 |
| PATH\_INFO | CGI 脚本的路径信息。 |
| QUERY\_STRING | 与发送 GET 请求一同发送的 URL 编码信息。 |
| REMOTE\_ADDR | 发出请求的远程主机的 IP 地址。这可用于记录日志或认证。 |
| REMOTE\_HOST | 发出请求的主机的完全资格名称。如果该信息不可用，则REMOTE\_ADDR 可用于获取 IR 地址。 |
| REQUEST\_METHOD | 用于请求的方法。最常用的方法是 GET 和 POST. |
| SCRIPT\_FILENAME | CGI 脚本的完全路径。 |
| SCRIPT\_NAME | CGI 脚本的名称。 |
| SERVER\_NAME | 服务器的主机名或是 IP 地址。 |
| SERVER\_SOFTWARE | 服务器正在运行软件的名称和版本。 |

这里提供一个小 CGI 程序来列出所有的 CGI 变量。点击链接 [Get Environment](http://www.tutorialspoint.com/cgi-bin/cpp_env.cgi) 来查看结果。

**#include <iostream>**

**#include <stdlib.h>**

**using** **namespace** std;

**const** string ENV[ 24 ] = {

"COMSPEC", "DOCUMENT\_ROOT", "GATEWAY\_INTERFACE",

"HTTP\_ACCEPT", "HTTP\_ACCEPT\_ENCODING",

"HTTP\_ACCEPT\_LANGUAGE", "HTTP\_CONNECTION",

"HTTP\_HOST", "HTTP\_USER\_AGENT", "PATH",

"QUERY\_STRING", "REMOTE\_ADDR", "REMOTE\_PORT",

"REQUEST\_METHOD", "REQUEST\_URI", "SCRIPT\_FILENAME",

"SCRIPT\_NAME", "SERVER\_ADDR", "SERVER\_ADMIN",

"SERVER\_NAME","SERVER\_PORT","SERVER\_PROTOCOL",

"SERVER\_SIGNATURE","SERVER\_SOFTWARE" };

**int** **main** ()

{

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>CGI Envrionment Variables</title>\n";

cout << "</head>\n";

cout << "<body>\n";

cout << "<table border = \"0\" cellspacing = \"2\">";

**for** ( **int** i = 0; i < 24; i++ )

{

cout << "<tr><td>" << ENV[ i ] << "</td><td>";

*// attempt to retrieve value of environment variable*

**char** \*value = getenv( ENV[ i ].c\_str() );

**if** ( value != 0 ){

cout << value;

}**else**{

cout << "Environment variable does not exist.";

}

cout << "</td></tr>\n";

}

cout << "</table><\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

C++ CGI 库

对于真正的应用来说，你需要你的CGI程序来完成许多操作。现在有一个用 C++ 编写的 CGI 程序库，你可以从 <ftp://ftp.gnu.org/gnu/cgicc/> 下载并按照以下步骤来安装此库：

$tar xzf cgicc-X.X.X.tar.gz

$cd cgicc-X.X.X/

$./configure --prefix=/usr

$make

$make install

你可以在 [**C++ CGI Lib Documentation**](http://www.gnu.org/software/cgicc/doc/index.html) 上查看相关文档。

GET 和 POST 方法

你一定遇到过很多将信息数据从浏览器发送到 Web 服务器，并最终发送到 CGI 程序的情况。浏览器最常使用 GET 和 POST 两种方法将信息传送给 Web 服务器。

利用 GET 方法传递数据

GET 方法将编码的用户信息附加到页面请求后面，页面和编码信息用字符？分割开。如下所示：

http:*//www.test.com/cgi-bin/cpp.cgi?key1=value1&key2=value2*

GET 方法是将信息数据从浏览器传递到 Web 服务器的默认方法。在浏览器的地址栏里面，此方法会产生一个长长的字符串。如果你有密码或其他敏感信息要传递给服务器，千万不要使用 GET 方法。 GET 方法有大小限制，在请求字符串里面，最多可以有 1024 个字符。

当使用 GET 方法时，信息数据使用 HTTP 报头 QUERY\_STRING 传递，同时你的 CGI 程序通过访问 QUERY\_STRING 环境变量来获取信息数据。

你可以通过简单地串联键值对和任何 URL 来传递信息数据，或者可以使用HTML<FORM>标签，通过利用 GET 方法来传递信息。

简单的 URL 例子： GET 方法

下面是一个使用 GET 方法传递两个值到 hello\_get.py 程序的简单 URL。

**/cgi-bin/cpp\_get.cgi?first\_name=ZARA&last\_name=ALI**

下面是一个生成 **cpp\_get.cgi** CGI 程序的程序，它会处理来自 Web 浏览器的输入。我们使用了 C++ CGI 库，这使得它非常容易访问传递的信息：

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc formData;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Using GET and POST Methods</title>\n";

cout << "</head>\n";

cout << "<body>\n";

form\_iterator fi = formData.getElement("first\_name");

**if**( !fi->isEmpty() && fi != (\*formData).end()) {

cout << "First name: " << \*\*fi << endl;

}**else**{

cout << "No text entered for first name" << endl;

}

cout << "<br/>\n";

fi = formData.getElement("last\_name");

**if**( !fi->isEmpty() &&fi != (\*formData).end()) {

cout << "Last name: " << \*\*fi << endl;

}**else**{

cout << "No text entered for last name" << endl;

}

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

现在，编译上面的程序，如下所示：

$g++ -o cpp\_get.cgi cpp\_get.cpp -lcgicc

它将会产生 cpp\_get.cgi，并把它放在你的 CGI 目录，并尝试使用下面的链接访问：

**/cgi-bin/cpp\_get.cgi?first\_name=ZARA&last\_name=ALI**

这将会产生如下的结果：

**First** name: ZARA

Last name: ALI

简单的 FORM 例子： GET 方法

下面是一个使用 HTML FORM 和提交按钮来传递两个值的简单例子。我们将使用同一个 CGI 脚本 cpp\_get.cgi 来处理此输入。

<form action="/cgi-bin/cpp\_get.cgi" method="get">

First Name: <input type="text" name="first\_name"> <br />

Last Name: <input type="text" name="last\_name" />

<input type="submit" value="Submit" />

</form>

利用 POST 方法传递数据

将信息数据传递给 CGI 程序，一般比较可靠的方法是 POST 方法。它会以与 GET 方法完全相同的方式将数据信息打包，但不是将其作为在 URL 中？后面的一个文本字符串来发送，而是将它作为一个单独的消息发送。此消息以标准输入的形式的发送到 CGI 脚本。

同样，利用同一个 cpp\_get.cgi 程序来处理 POST 方法的输入。我们用与上面一样的例子，利用 HTML FORM 和提交按钮来传递两个值，但是这一次用 POST 方法，如下所示：

<form action="/cgi-bin/cpp\_get.cgi" method="post">

First Name: <input type="text" name="first\_name"><br />

Last Name: <input type="text" name="last\_name" />

<input type="submit" value="Submit" />

</form>

传递复选框数据到 CGI 程序

当有不止一个选项需要被选择的时候，要使用复选框。

下面是一个有两个复选框的 HTML 表单样例代码。

<form action="/cgi-bin/cpp\_checkbox.cgi"

method="POST"

target="\_blank">

<input type="checkbox" name="maths" value="on" /> Maths

<input type="checkbox" name="physics" value="on" /> Physics

<input type="submit" value="Select Subject" />

</form>

下面 C++ 程序会产生 cpp\_checkbox.cgi 脚本来处理由 Web 浏览器提供的复选框按钮的输入。

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc formData;

**bool** maths\_flag, physics\_flag;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Checkbox Data to CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

maths\_flag = formData.queryCheckbox("maths");

**if**( maths\_flag ) {

cout << "Maths Flag: ON " << endl;

}**else**{

cout << "Maths Flag: OFF " << endl;

}

cout << "<br/>\n";

physics\_flag = formData.queryCheckbox("physics");

**if**( physics\_flag ) {

cout << "Physics Flag: ON " << endl;

}**else**{

cout << "Physics Flag: OFF " << endl;

}

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

传递单选按钮数据到 CGI 程序

当仅需要选择一个选项时，使用单选按钮。

下面是一个有两个单选按钮的 HTML 表单样例代码。

<form action="/cgi-bin/cpp\_radiobutton.cgi"

method="post"

target="\_blank">

<input type="radio" name="subject" value="maths"

checked="checked"/> Maths

<input type="radio" name="subject" value="physics" /> Physics

<input type="submit" value="Select Subject" />

</form>

下面的 C++ 程序会产生 cpp\_checkbox.cgi 脚本来处理由 Web 浏览器提供的单选按钮的输入。

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc formData;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Radio Button Data to CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

form\_iterator fi = formData.getElement("subject");

**if**( !fi->isEmpty() && fi != (\*formData).end()) {

cout << "Radio box selected: " << \*\*fi << endl;

}

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

传递文本域数据到 CGI 程序

当有多行文本需要传递到 CGI 程序时，要使用文本域元素。

下面是一个有文本域框的 HTML 表单样例代码。

<form action="/cgi-bin/cpp\_textarea.cgi"

method="post"

target="\_blank">

<textarea name="textcontent" cols="40" rows="4">

Type your text here...

</textarea>

<input type="submit" value="Submit" />

</form>

下面的 C++ 程序会产生 cpp\_checkbox.cgi 脚本来处理由 Web 浏览器提供的文本域的输入。

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc formData;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Text Area Data to CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

form\_iterator fi = formData.getElement("textcontent");

**if**( !fi->isEmpty() && fi != (\*formData).end()) {

cout << "Text Content: " << \*\*fi << endl;

}**else**{

cout << "No text entered" << endl;

}

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

传递下拉框数据到 CGI 程序

当有多个可选项，但是只能选择一个或者两个选项时，需要使用下拉框。

下面是一个有一个下拉框的 HTML 表单样例代码。

<form action="/cgi-bin/cpp\_dropdown.cgi"

method="post" target="\_blank">

<select name="dropdown">

<option value="Maths" selected>Maths</option>

<option value="Physics">Physics</option>

</select>

<input type="submit" value="Submit"/>

</form>

下面的 C++ 程序会产生 cpp\_checkbox.cgi 脚本来处理由 Web 浏览器提供的下拉框的输入。

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc formData;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Drop Down Box Data to CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

form\_iterator fi = formData.getElement("dropdown");

**if**( !fi->isEmpty() && fi != (\*formData).end()) {

cout << "Value Selected: " << \*\*fi << endl;

}

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

在 CGI 中使用 Cookies

HTTP 协议是一个无状态协议。但是对于一个商业网站，它需要保持不同的页面间的会话信息。例如，一个用户的注册需要完成许多的页面，那么将如何保持所有网页之间的用户的会话信息将是一个问题。

在许多情况下，使用 cookies 是记录和跟踪，喜好，所购物品，佣金或其他能提供更好访问体验的信息或是网站的统计数据的最有效的方法。

它是如何工作的

你的服务器发送一些 cookie 格式的数据给访问者的浏览器。该浏览器可能接受了该 cookie. 如果是这样，它以纯文本的方式记录在访问者的硬盘驱动器上。现在，当访问者访问你的网站上的其他页面时，该 cookie 可用于检索。一旦检索到，你的服务器就会知道/记起已存储的信息。

Cookies 是一个记录了5可变长度字段的纯文本数据：

* **Expires**：cookie 将失效的日期。 如果此字段为空，那么当访问者退出浏览器后 cookie 将会失效。
* **Domain** ：网站的域名。
* **Path** ：设置 cookie 的目录或网页的路径。如果你想要从任何目录或页面检索 cookie ，此字段将设置为空。
* **Secure** ：如果该字段包含单词 "secure"，那么该 cookie 仅可由一个安全的服务器检索到。如果该字段为空，将不存在这样的限制。
* **Name=Value**：cookie 将以键值对的形式设置和检索。

设置 Cookies

将 cookies 发送到浏览器是非常容易的。这些 cookie 会设置在 HTTP报头的 Content-type 字段之前，并与其一起发送出去。假设你要将 UserID 和 Password 设置为 cookie. cookie 的设置如下所示

**#include <iostream>**

**using** **namespace** std;

**int** **main** ()

{

cout << "Set-Cookie:UserID=XYZ;\r\n";

cout << "Set-Cookie:Password=XYZ123;\r\n";

cout << "Set-Cookie:Domain=www.tutorialspoint.com;\r\n";

cout << "Set-Cookie:Path=/perl;\n";

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Cookies in CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

cout << "Setting cookies" << endl;

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

从这个例子中,你一定要了解如何设置 cookie。 我们使用 HTTP 报头的 **Set-Cookie** 字段设置 cookie。

cookies 的属性，如 Expires， Domain 和 Path 是可选设置项。值得注意的是，cookies 设置在魔力代码行 **Content-type:text/html\r\n\r\n** 之前。

编译上面的程序将产生 setcookies.cgi，并尝试使用下面的链接来设置 cookie。它会在你的电脑上设置四个 cookie：

**/cgi-bin/setcookies.cgi**

检索 Cookie

检索所有设置的 cookie 是非常容易的。 cookie 存储在 CGI 环境变量的 HTTP\_COOKIE 字段，它们的形式如下所示：

key1=value1;key2=value2;key3=value3....

下面是一个如何检索 cookie 的例子。

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc cgi;

const\_cookie\_iterator cci;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>Cookies in CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

cout << "<table border = \"0\" cellspacing = \"2\">";

*// get environment variables*

**const** CgiEnvironment& env = cgi.getEnvironment();

**for**( cci = env.getCookieList().begin();

cci != env.getCookieList().end();

++cci )

{

cout << "<tr><td>" << cci->getName() << "</td><td>";

cout << cci->getValue();

cout << "</td></tr>\n";

}

cout << "</table><\n";

cout << "<br/>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

现在，编译上面的程序将产生 getcookies.cgi，并且试图列出你电脑上所有可用的 cookies：

**/cgi-bin/getcookies.cgi**

这将会列出在上一节设置的四个 cookies 和其它你电脑上设置的 cookies。

**UserID** XYZ

Password XYZ123

Domain www.tutorialspoint.com

Path /perl

文件上传

要上传一个文件，HTML 表单必须将 enctype 属性设置为 **multipart/form-data**。 type 为 file 的 input 标签将会产生一个“选择文件”按钮。

<html>

<body>

<form enctype="multipart/form-data"

action="/cgi-bin/cpp\_uploadfile.cgi"

method="post">

<p>File: <input type="file" name="userfile" /></p>

<p><input type="submit" value="Upload" /></p>

</form>

</body>

</html>

**注意：**对上面的例子，我们已经禁止了向我们的服务器上传文件。但是你可以在你自己的服务上实验以上的代码。

下面是一个用来处理文件上传的脚本 **cpp\_uploadfile.cpp**：

**#include <iostream>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <cgicc/CgiDefs.h>**

**#include <cgicc/Cgicc.h>**

**#include <cgicc/HTTPHTMLHeader.h>**

**#include <cgicc/HTMLClasses.h>**

**using** **namespace** std;

**using** **namespace** cgicc;

**int** **main** ()

{

Cgicc cgi;

cout << "Content-type:text/html\r\n\r\n";

cout << "<html>\n";

cout << "<head>\n";

cout << "<title>File Upload in CGI</title>\n";

cout << "</head>\n";

cout << "<body>\n";

*// get list of files to be uploaded*

const\_file\_iterator file = cgi.getFile("userfile");

**if**(file != cgi.getFiles().end()) {

*// send data type at cout.*

cout << HTTPContentHeader(file->getDataType());

*// write content at cout.*

file->writeToStream(cout);

}

cout << "<File uploaded successfully>\n";

cout << "</body>\n";

cout << "</html>\n";

**return** 0;

}

上面的例子是在 **cout** 流中写内容，但是你可以打开你的文件流，并将上传文件的内容保存到一个预定位置的文件中。