

Connecting Second Life to Web Applications

Created for CAP 4800/CAP 5805 on 09/09/09

It is recommend you use space on the CISE server to house your web programs, since you can run PHP scripts, and compile C++ code in this environment. Also, you can run Java Servlets on the server after Apache Tomcat is installed. When working with files on the server, you'll need to work from the CISE lab, or connect remotely using programs like Putty and/or WinSCP (Windows), or the Terminal and/or Cyber Duck (OS X). If working in this environment is new to you, please see the TA and he will be glad to help you get up and running.

The following instructions will be step-by-step, as if working on the CISE Servers. In the first three examples, you are creating a simple object in Second Life that, when given a number through chat on channel 5, outputs the factorial of that number through local chat. In the fourth example, you are connecting to Second Life from a different location and using an LSL script to calculate the factorial.

Wherever you are running a PHP script from, you need to place a file called 'htaccess' with permissions 644 (read all/write owner). The contents of this file should be:

```
AddHandler cgi-script .php
DirectoryIndex index.html index.php
```

I. Connecting to PHP

1. Create a new file somewhere under your 'public_html' folder called 'SLExample.php.'
2. Set the permission of this file to 755 (read all/execute all).
3. Add the following contents to the file:

```
#!/usr/local/bin/php

<?php

$num = $_GET["var"];
$val = factorial($num);
echo($val);

function factorial($n)
{
    if($n == 0)
    {
        return 1;
    }
    else if($n < 0)
    {
        return -1;
    }
    else
    {
        $value = $n;
        for($i=$n-1;$i>1;$i--)
        {
            $value *= $i;
        }
        return $value;
    }
}

?>
```

Note: #!/usr/local/bin/php is required to run a PHP script on the CISE server.

4. Create a new object in Second Life.
5. Attach a script to this object.
6. Remove the existing code, and add the following contents to the script:

```

key requestid;
string NUM;

default
{
    state_entry()
    {
        llListen(5, "", "", "");
        llSetText("Factorial: php. Give me an integer on channel 5.",
<1,1,0>,1);
    }

    listen(integer chan, string name, key id, string message)
    {
        requestid = llHTTPRequest("your_url?var="+message,
[HTTP_METHOD, "GET"], "");
        NUM = message;
    }

    http_response(key request_id, integer status, list metadata, string body)
    {
        if (request_id == requestid)
        {
            llSay(0, "The factorial of "+NUM+" is "+body);
        }
    }
}

```

Change "your_url" in the llHTTPRequestFunction to the URL of your PHP script you created in step 1.

7. In Second Life, set the script to 'Running.'
8. Leave Edit mode, typically by pressing ESC.

Now when you chat on channel 5 near your object, the LSL script should run and send the chat message as input to the PHP program. Try it by typing "/5 10" in the chat. Your object should say "The factorial of 10 is 3628800."

II. Connecting to PHP, then running a C++ program

1. Create a new file somewhere under your 'public_html' folder called 'SLExample.cpp.'
2. Add the following contents to the file:

```
#include <iostream>

/* this program calculates the factorial of argv[1] */

int main ( int argc, const char* argv[])
{
    if(argc <2)
    {
        printf("Please provide a number.");
        return 0;
    }

    int f = atoi(argv[1]);

    if(f == 0)
    {
        printf("1");
    }
    else if(f < 0)
    {
        printf("No negative numbers, please.");
    }
    else
    {
        int fact = 1;

        for(int i=f;i>1;i--)
        {
            fact *= i;
        }

        printf("%i", fact);
    }
    return 0;
}
```

3. Compile the file from the command line and write the output to SLExample.program using the following command:

```
g++ SLExample.cpp -o SLExample.program
```

4. Set the permission of SLExample.program to 755 (read all/execute all).
5. Create a new file in the same directory called 'SLExample.php.'
6. Add the following contents to the SLExample.php:

```
#!/usr/local/bin/php

<?php

$num = $_GET["var"];
$val = exec("./SLExample.program ".$num, $result);
echo ($result[0]);

?>
```

Note: #!/usr/local/bin/php is required to run a PHP script on the CISE server.

7. Set the permission of SLExample.php to 755 (read all/execute all).
8. Create a new object in Second Life.
9. Attach a script to this object.
10. Remove the existing code, and add the following contents to the script:

```
key requestid;
string NUM;

default
{
    state_entry()
    {
        llListen(5, "", "", "");
        llSetText("Factorial: c++ via php. Give me an integer on channel 5.",
<1,1,0>,1);
    }

    listen(integer chan, string name, key id, string message)
    {
        requestid = llHTTPRequest("your_url?var="+message,
[HTTP_METHOD, "GET"], "");
        NUM = message;
    }

    http_response(key request_id, integer status, list metadata, string body)
    {
        if (request_id == requestid)
        {
            llSay(0, "The factorial of "+NUM+" is "+body);
        }
    }
}
```

Change "your_url" in the llHTTPRequestFunction to the URL of your PHP script you created in step 7.

11. In Second Life, set the script to 'Running.'
12. Leave Edit mode, typically by pressing ESC.

Now when you chat on channel 5 near your object, the LSL script should run and send the chat message as input to the PHP program. Try it by typing "/5 10" in the chat field. Your object should say "The factorial of 10 is 3628800."

III. Connecting to PHP, then running a Java Servlet program

1. Download and unzip the most recent Apache Tomcat from: <http://tomcat.apache.org/download-60.cgi#6.0.20>
2. Change the root folder name from 'apache-tomcat-XXX to just 'tomcat.'
3. Upload the 'tomcat' folder to your space on the CISE server. It should be around 9 MB.
4. Within the 'tomcat' folder, navigate to 'webapps/examples/WEB-INF/classes.' This is where we will create our servlet. There are other programs in this directory to use as examples, as well.
5. Create a file called SLExample.java.
6. Add the following contents to SLExample.java:

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SLExample extends HttpServlet
{
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        ResourceBundle rb =
            ResourceBundle.getBundle("LocalStrings", request.getLocale());
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String str = request.getParameter("number");

        int number = Integer.valueOf(str).intValue();

        int value = 1;
        if (number == 0)
        {
            value = 1;
        }
        else if (number < 0)
        {
            value = -1;
        }
        else
        {
            value = number;
            for (int i = (number-1); i > 1; i--)
            {
                value *= i;
            }
        }

        out.println(""+value);
    }
}
```

```
}  
}
```

7. From the command line, navigate to the root 'tomcat' directory.
8. Compile SLEExample.java using the following command:

```
javac -classpath lib/servlet-api.jar webapps/examples/WEB-INF/classes/  
SLEExample.java
```

9. Set the permission of SLEExample.class to 755 (read all/execute all).
10. Add the servlet to the web.xml file located in 'tomcat/webapps/examples/WEB-INF' by adding:

```
<servlet>  
<servlet-name>SLEExample</servlet-name>  
<servlet-class>SLEExample</servlet-class>  
</servlet>
```

near the other servlet definitions, and by adding:

```
<servlet-mapping>  
<servlet-name>SLEExample</servletName>  
<url-pattern>/servlets/servlets/SLEExample</url-pattern>  
</servlet-mapping>
```

by the other servlet-mapping definitions.

11. Choose a machine and port to run it on. For the machine, you can use any public machine (one of the lab machines for example). The more important decision is what port you will run tomcat on. For the port, choose a number between 10000 and 65000 (choose it semi-randomly so that it is unlikely that you will conflict with someone else running a servlet).

12. Modify the server config file. This file is found in: 'tomcat/conf/server.xml.' Look for the two lines containing:

```
port="8005"  
and  
port="8080"
```

Both of these lines must be changed. Change the 8005 to one number, and change the 8080 to a second (different) number. Both should be "random" and between 10000 and 65000. Make a note of the port number that you replace the 8080 with.

13. Startup Tomcat by navigating to the root 'tomcat' directory from the command line, and using the following command:

```
bin/startup.sh
```

14. Create a new object in Second Life.
15. Attach a script to this object.
16. Remove the existing code, and add the following contents to the script:

```

key requestid;
string NUM;

default
{
    state_entry()
    {
        llListen(5,"","","");
        llSetText("Factorial: java servlet. Give me an integer on channel
5.", <1,1,0>,1);
    }

    listen(integer chan, string name, key id, string message)
    {
        requestid = llHTTPRequest("your_url?number="+message,
[HTTP_METHOD,"GET"], "");
        NUM = message;
    }

    http_response(key request_id, integer status, list metadata, string body)
    {
        if (request_id == requestid)
        {
            llSay(0,"The factorial of "+NUM+" is "+body);
        }
    }
}

```

Note: replace "your_url" with "http://MACHINE.cise.ufl.edu:PORT/examples/servlets/SLEExample", where MACHINE is a CISE machine and PORT is the number you replaced 8080 with in step 11. For example, a valid url could be:

```
http://rain.cise.ufl.edu:13000/examples/servlets/SLEExample
```

17. In Second Life, set the script to 'Running.'
18. Leave Edit mode, typically by pressing ESC.

Now when you chat on channel 5 near your object, the LSL script should run and send the chat message as input to the Java Servlet program. Try it by typing "/5 10" in the chat field. Your object should say "The factorial of 10 is 3628800."

19. Shutdown Tomcat by navigating to the root 'tomcat' directory from the command line, and using the following command:

```
bin/shutdown.sh
```

IV. Connecting to LSL from PHP (XML-RPC)

1. Create a new object in Second Life.
2. Attach a script to this object.
3. Remove the existing code, and add the following contents to the script:

```
key remoteChannel;

init() {
    llOpenRemoteDataChannel(); // create an XML-RPC channel
    llOwnerSay("My key is " + (string)llGetKey());
}

integer factorial(integer n)
{
    if (n == 0 || n == 1)
        return 1;

    else
    {
        integer k = n - 1;
        return n * factorial (k);
    }
}

default {
    state_entry() {
        init();
    }

    state_exit() {
        return;
    }

    on_rez(integer param) {
        llResetScript();
    }

    remote_data(integer type, key channel, key message_id, string sender,
integer ival, string sval) {
        if (type == REMOTE_DATA_CHANNEL) { // channel created
            llSay(DEBUG_CHANNEL, "Channel opened for REMOTE_DATA_CHANNEL" +
                (string)channel + " " + (string)message_id + " " +
(string)sender + " " +
                (string)ival + " " + (string)sval);
            remoteChannel = channel;
            llOwnerSay("Ready to receive requests on channel \"" +
(string)channel + "\"");
        }
    }
}
```

```

        state receiving; // start handling requests
    } else {
        llSay(DEBUG_CHANNEL,"Unexpected event type");
    }
}
}

state receiving {

    state_entry() {
        llOwnerSay("Ready to receive information from outside SL");
    }

    state_exit() {
        llOwnerSay("No longer receiving information from outside SL.");
        llCloseRemoteDataChannel(remoteChannel);
    }

    on_rez(integer param) {
        llResetScript();
    }

    remote_data(integer type, key channel, key message_id, string sender,
integer ival, string sval) {
        if (type == REMOTE_DATA_REQUEST) { // handle requests sent to us
            llSay(DEBUG_CHANNEL,"Request received for REMOTE_DATA_REQUEST "
+ (string)channel + " " +
                (string)message_id + " " + (string)sender + " " +
(string)ival + " " + (string)sval);
            llRemoteDataReply(channel,NULL_KEY,"I got it",2008);
            llOwnerSay("I just received data in "+ llGetRegionName() +
                " at position " + (string)llGetPos() + "\n" +
                "The string was " + sval + "\nThe number was " +
(string)ival + ".");

                llOwnerSay("The factorial of"+(string)ival+" is "+
(string)factorial(ival));

            &nbsp; }
        }
}
}

```

Note: This script was taken from http://wiki.secondlife.com/wiki/Category:LSL_XML-RPC and modified slightly.

4. Set the script to 'Running.'
5. Note the string for channel, you'll need it soon.
6. Create a new file somewhere under your 'public_html' folder called 'SLExample_XMLRPC.php.'

7. Set the permission of this file to 755 (read all/execute all).
8. Add the following contents to the file:

```
#!/usr/local/bin/php

<?php
    echo '<pre>';
    $channel = ""; //Fill in the channel you are using (key)
    $intvalue = ""; //Fill in the intvalue you are using (integer)
    $strvalue = ""; //Fill in the strvalue you are using (string)
    $xmldata = "<?xml
version=\"1.0\"?><methodCall><methodName>llRemoteData</methodName>
<params><param><value><struct>
<member><name>Channel</name><value><string>\".$channel.\"</string></value></member>
<member><name>IntValue</name><value><int>\".$intvalue.\"</int></value></member>
<member><name>StringValue</name><value><string>\".$strvalue.\"</string></value></member>
</struct></value></param></params></methodCall>";
    echo sendToHost("xmlrpc.seconlife.com", "POST", "/cgi-bin/xmlrpc.cgi", $xmldata);
    echo '</pre>';

function sendToHost($host,$method,$path,$data,$useragent=0)
{
    $buf="";
    // Supply a default method of GET if the one passed was empty
    if (empty($method))
        $method = 'GET';
    $method = strtoupper($method);

    $fp = fsockopen($host, 80, $errno, $errstr, 30);

    if( !$fp )
    {
        $buf = "$errstr ($errno)<br />\n";
    }else
    {
        if ($method == 'GET')
            $path .= '?' . $data;
        fputs($fp, "$method $path HTTP/1.1\r\n");
        fputs($fp, "Host: $host\r\n");
        fputs($fp, "Content-type: text/xml\r\n");
        fputs($fp, "Content-length: " . strlen($data) . "\r\n");
        if ($useragent)
            fputs($fp, "User-Agent: MSIE\r\n");
        fputs($fp, "Connection: close\r\n\r\n");
        if ($method == 'POST')
            fputs($fp, $data);
        while (!feof($fp))
            $buf .= fgets($fp,128);
        fclose($fp);
    }
    return $buf;
}
?>
```

Note: This script was taken from http://wiki.secondlife.com/wiki/Category:LSL_XML-RPC and modified slightly.
Note: `#!/usr/local/bin/php` is required to run a PHP script on the CISE server.

9. Fill in the 'channel' string with the string from step 5.
10. Fill in the intvalue string for which you want to compute the factorial in Second Life.
11. Fill in the strvalue string if you want.
12. Verify your script is still running in Second Life.
13. In a web browser, navigate to the PHP script in your CISE webspace. If the script is running properly, you should see something like the following:

```
HTTP/1.1 200 OK
Date: Wed, 09 Sep 2009 15:41:31 GMT
Server: Apache/2.2.3 (Debian) DAV/2 SVN/1.4.2 mod_ssl/2.2.3 OpenSSL/0.9.8c
Content-Length: 372
Connection: close
Content-Type: text/xml

Channel138a9ba6d-6583-db3e-ac56-21b476a3a7e4StringValueI got itIntValue2008
```

14. Return to Second Life and see the output from the LSL script in the local chat, after it received a message from your PHP script.