Networking

Introduction

- The stream layer enables the access both to the local files system and to resources on the web.

  The same functionality used to access the local files system can be used to access various resources on the web.

Introduction

- The easies way to access a resource on the web would be treating it as if it is a local file on the files system.

```
$resource = fopen ('http://www.zindell.com','r');
$content = '';
if ($resource)
{
   while ($str = fread ($resource, 1000))
   {
      $content .= $str;
   }
}
else
{
   throw new Exception ("Unable to open connection");
}
```

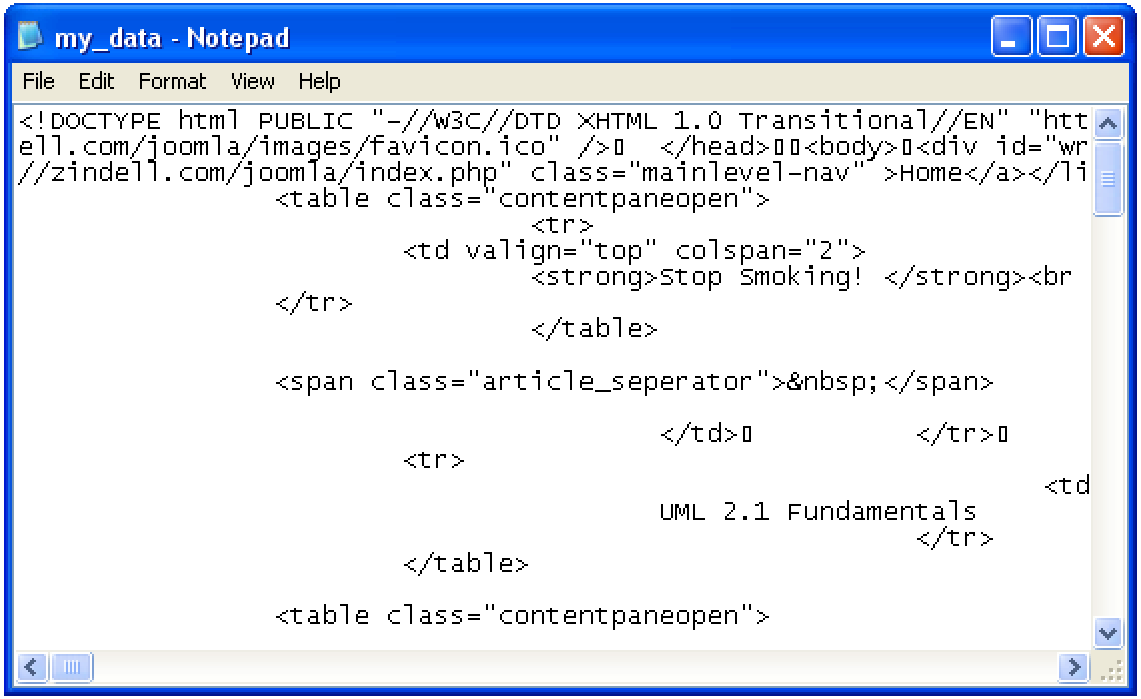
Sample

```
<?php

    $resource = fopen ('http://zindell.com/joomla/','r');
    $content = '';
    if ($resource)
    {
        while ($str = fread ($resource, 1000))
        {
            $content .= $str;
        }
    }
    else
    {
        throw new Exception ('Unable to open connection');
    }
    $fp = fopen('my_data.txt', 'w+');
    fwrite($fp, $content);
    fclose($fp);

?>
```

Sample



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
ell.com/joomla/images/favicon.ico" />▯  </head>▯▯<body>▯<div id="wr
//zindell.com/joomla/index.php" class="mainlevel-nav" >Home</a></li
                <table class="contentpaneopen">
                                <tr>
                        <td valign="top" colspan="2">
                                <strong>Stop Smoking! </strong><br
                </tr>
                                </table>

                <span class="article_seperator">&nbsp;</span>

                                        </td>▯              </tr>▯
                        <tr>
                                                                <td
                                        UML 2.1 Fundamentals
                                                        </tr>
                        </table>

                <table class="contentpaneopen">
```

The 'require' & 'include' Commands

- It is possible to use networking to access files on other servers using the 'require' and the 'include' commands.

```
include 'http://www.abelski.com/funcs.php';
include 'http://www.zindell.com/lib.php';
```

- Using this capability might cause a security hole.

Resource Context

- Using the `stream_context_create` function we can get a resource we can later use to customize the way we use stream resources.

  The type of the returned value is resource. We can pass it over to the `file_get_contents` function. Doing so the file handle we get will be tweaked in accordance with the object the `stream_context_create` function returned.

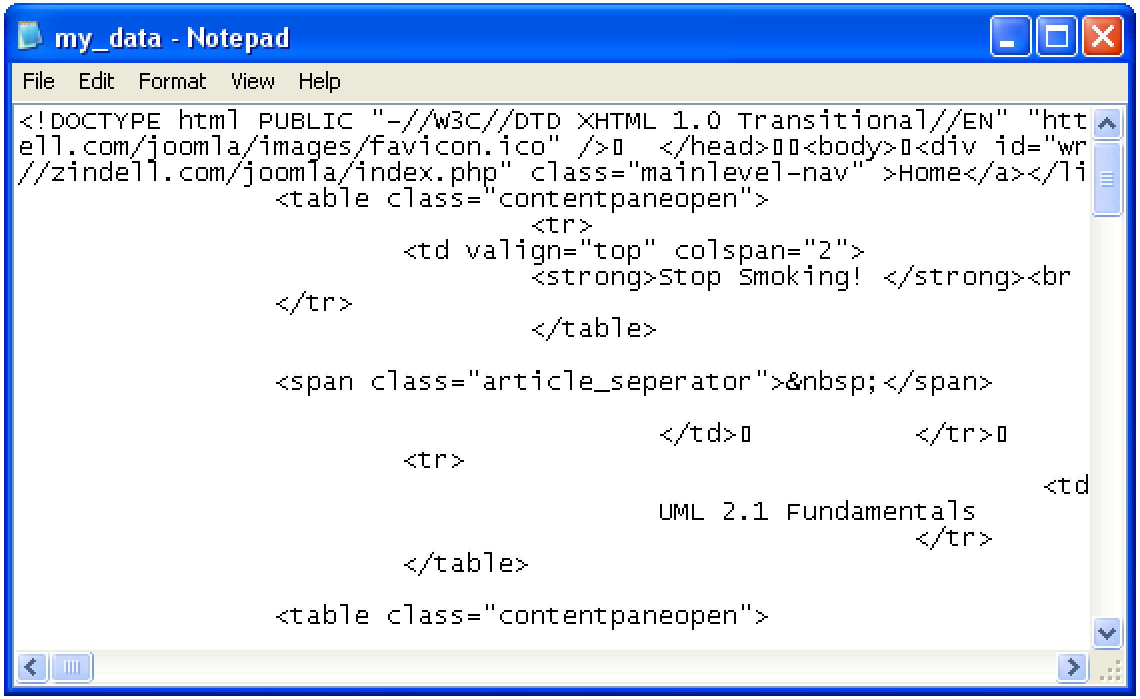
Resource Context Sample

```
<?php
    $http_options = stream_context_create(
        array(’http’ =>
            array(
                   "user_agent" => "Sabra Browser",
                   "max_redirects" => 3
                                          )
                                              )
                                                  );
    $content = "";
    $file = fopen(
        "http://zindell.com/joomla",
        'r',
        false,
        $http_options);
```

Resource Context Sample

```
    if ($file)
    {
        while ($str = fread ($file, 1000))
        {
            $content .= $str;
        }
    }
    else
    {
        throw new Exception ("Unable to open connection");
    }
    $fp = fopen('my_data.txt', 'w+');
    fwrite($fp, $content);
    fclose($fp);
?>
```

Resource Context Sample



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
ell.com/joomla/images/favicon.ico" />▯  </head>▯▯<body>▯<div id="wr
//zindell.com/joomla/index.php" class="mainlevel-nav" >Home</a></li
                <table class="contentpaneopen">
                                <tr>
                        <td valign="top" colspan="2">
                                <strong>Stop Smoking! </strong><br
                </tr>
                                </table>

                <span class="article_seperator">&nbsp;</span>

                                        </td>▯          </tr>▯
                        <tr>
                                                                <td
                                        UML 2.1 Fundamentals
                                                        </tr>
                        </table>

                <table class="contentpaneopen">
```

Client Server Applications

- Calling the `stream_socket_server()` function returns a resource that represents a server socket.

  This server socket will be waiting to be connected by another socket as soon as we call the stream_socket_accept() function.. If this function fails it returns '`false`'.

- We can pass over the server socket resource to the `stream_socket_accept()` function, that returns a socket resource when the connection succeeds.

Client Server Applications

```
<?php

$server_socket = stream_socket_server("tcp://127.0.0.1:1300");
while ($socket = stream_socket_accept($server_socket))
{
    fwrite($socket, "<br>Hello World<br>");
    fclose($socket);
}
fclose($server_socket);

?>
```

Client Server Applications

- Calling the `stream_socket_client()` function returns a resource that represents the connection.

  If this function fails it returns '`false`'. The returned resource is kind of a file handle.

- We can use the resource we get as any other file handle.

  We can use the `fread()` function to read the received data, we can call the `feof()` function in order to know whether there is still data to read and we can call the `fclose()` function in order to close it.

Client Server Applications

```
<?php

$socket = stream_socket_client('tcp://127.0.0.1:1300');
if(!$socket)
{
    echo "<br>error<br>";
}
else
{
    while (!feof($socket))
    {
        echo fread($socket, 100);
    }
    fclose($socket);
}

?>
```

Client Server Applications

- We can use the resources that represent connection we get, both the one on the server side and the one on the client side, as any other file handle.

  We can use the `fread()` function to read the received data, we can use `fwrite()` function to write data, we can call the `feof()` function in order to know whether there is still data to read and we can call the `fclose()` function in order to close the resources.

Stream Filters

- We can connect stream filters with each other in order to pass data through a series of stream filters. Each stream filter can alter the stream (the data being passed).

  We can, for example, have a stream filter that changes all texts that go through it into upper case texts. Another example can be a filter that encodes the passed data.

Stream Filters

- We can add a filter to an available stream by calling one of the following two functions:

```
stream_filter_prepend()

stream_filter_append()
```

- The first adds a filter to the beginning of the stream. The second adds a filter to the end of the stream.

Stream Filters

```
<?php

$server_socket = stream_socket_server("tcp://127.0.0.1:1300");
while ($socket = stream_socket_accept($server_socket))
{
    stream_filter_append($socket, 'string.toupper');
    stream_filter_append($socket, 'zlib.deflate');
    fwrite($socket, "<br>Hello World<br>");
    fclose($socket);
}
fclose($server_socket);

?>
```

Stream Filters

```
<?php

$socket = stream_socket_client('tcp://127.0.0.1:1300');
if(!$socket)
{
    echo "<br>error<br>";
}
else
{
    stream_filter_append($socket, 'zlib.inflate');
    while (!feof($socket))
    {
        echo fread($socket, 100);
    }
    fclose($socket);
}

?>
```