Session 11

File Handling and Exception Handling in PHP

Session Overview

In this session, students will learn how to:

- Describe file uploading and file handling in PHP
- Explain upload script creation and upload form creation
- Explain files in PHP through the `readfile()` function
- Define file open/read/close in PHP
- Identify and explain various file functions
- Elaborate the process of file upload using PHP
- Explain error and exception handling in PHP
- Illustrate `die()` function, try…catch statement, exception object, and try…catch…finally statement

File Handling [1-2]

PHP Write File - `fwrite()`

`fwrite()` function helps a user to write some content of string(s) into a file

Syntax:

```
int fwrite (reso
urce $handle, st
ring $string [,
 int $length ])
```

File Handling [2-2]

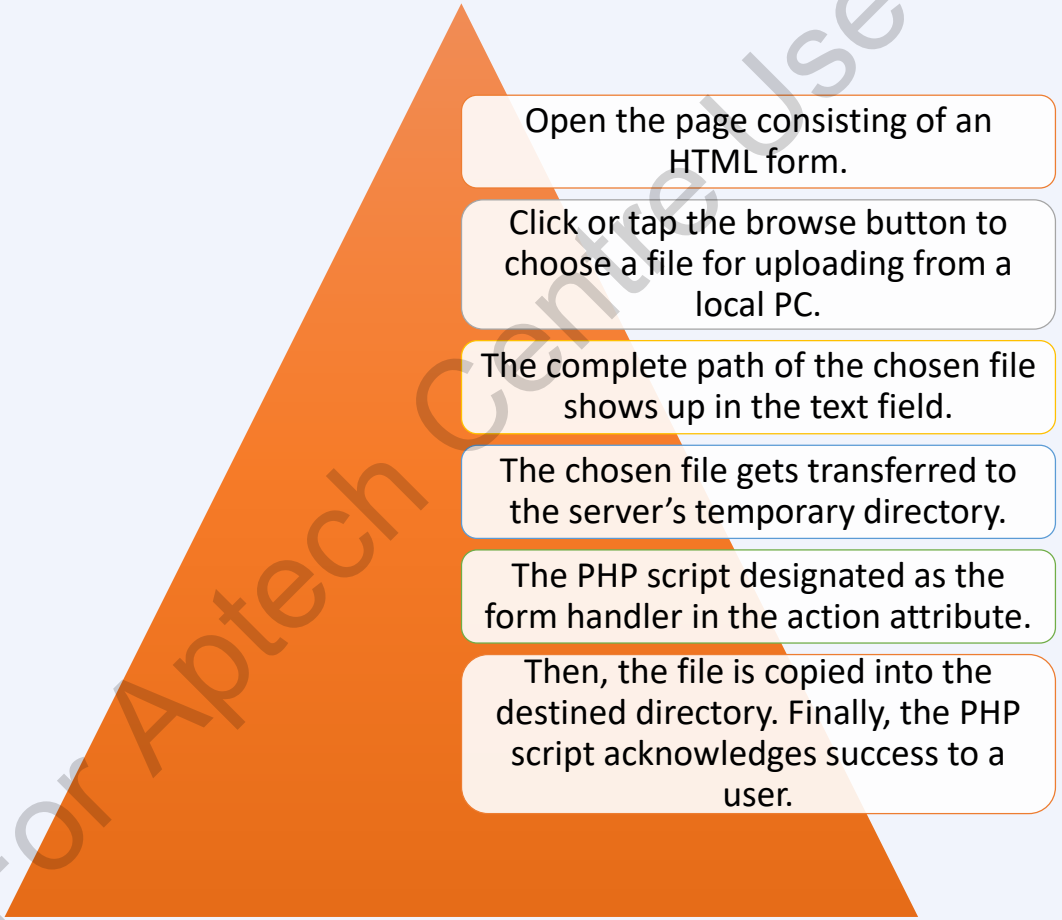
PHP Delete File - `unlink()`

Users can delete a file using the PHP `unlink()` function

Syntax:

```
bool unlink (string $filename [, resource $context ])
```

File Uploading



Upload Form Creation

Using HTML form, PHP code allows users to upload files to the server. In the beginning, files are placed in a temporary directory. Then, a PHP script is used to move the files to a target destination.

The HTML code given in Code creates an uploaded form. The form has the ectype attribute set to multipart/form-data and the method attribute assigned as a post.

Upload Script Creation

- `$_FILES['file']['error']` - Error code related to file upload.
- `$_FILES['file']['type']` - Multipurpose Internet Mail Extensions (MIME) type of the uploaded file.
- `$_FILES['file']['name']` - Real name of the uploaded file.
- `$_FILES['file']['size']` - Size of the uploaded file in bytes.
- `$_FILES['file']['tmp_name']` - Uploaded file located in the temporary directory on a Web server.

Manipulating Files [1-3]

- PHP has multiple functions for reading, editing, creating, and uploading files. However, it is important to be cautious while manipulating files. A wrong move can prove to be detrimental to the computer system.

- Following are common errors that occur during file manipulation:
  - Deleting a file's content
  - Filling a hard disk drive with garbage data
  - Editing the wrong file

Manipulating Files [2-3]

Open File - fopen() PHP's fopen() function helps us to open files.

Syntax:
```
resource fopen (string $filename, string $mode [
, bool $use_include_path = false [, resource $context
]]
```

Read File - `fread()` The `fread()` function in PHP enables users to read the content of an open file.

Syntax:
```
string fread(resource $handle, int $length)
```

Close File - `fclose()` The `fclose()` function helps a user to close an opened.

Syntax:
```
bool fclose (resource $handle)
```

Manipulating Files [3-3]

Read Single Line - fgets() PHP's fgets() function allows a user to read a single line from an open file.

Check End-Of-File - feof() PHP function feof() helps a user to loop through data of undisclosed length. It examines a file and verifies whether the user has reached the ‘End-of-File' (EOF).

Read Single Character - fgetc() PHP's fgetc() function serves as a means to read a single character from the files.

PHP Files

Reading File - `fopen()` and `fread()` : Users can read a file's content using a function termed as `fread()` after it has been opened through the `fopen()` function. The `fread()` function requires two arguments.

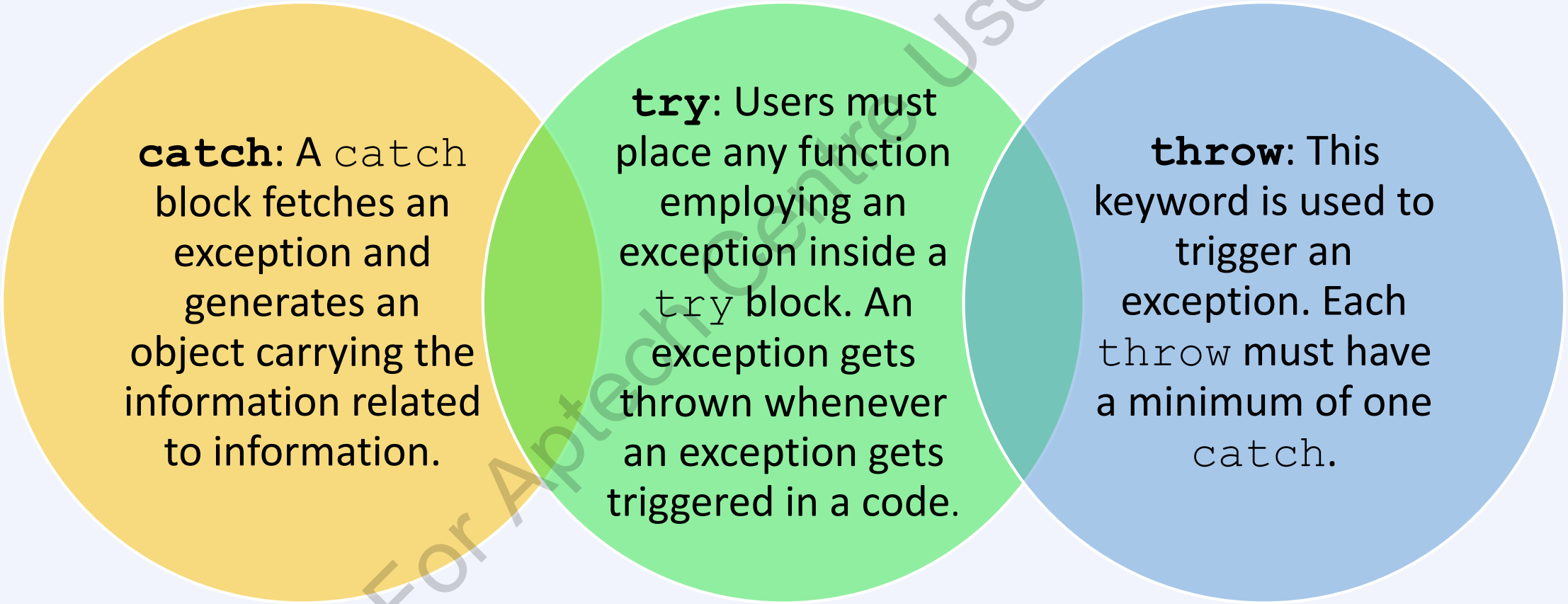
Writing to File - `fwrite()` : Users can use the `fwrite()` function to append text to an existing file or write a new file. This PHP function requires two arguments. One specifies the string of data to be written and the other specifies a file pointer.

Error and Exception Handling in PHP

Error handling can be defined as the process of detecting errors in a program and undertaking appropriate measures to get rid of them.

`die()` Function: Users must evaluate any error conditions while writing PHP code. The purpose of the `die()` function is similar to that of an exit function.

Exception Handling [1-5]



Exception Handling [2-5]

In a `catch` block, exceptions can be thrown or re-thrown.

Each `try` block must have a minimum of one matching `catch` block. Users can utilize multiple `catch` blocks for catching multiple classes of exceptions.

An exception can be caught and thrown within PHP. The code that can successfully catch exceptions might be included in a `try` block.

Exception Handling [3-5]

- **`try...catch` Statement**

Users can implement the `try...catch` statement for catching exceptions and keep the process going.

- **Syntax**:

```
try {
code that may cause exceptions
  }
catch(Exception $e)
  {
code that gets executed after catching an exception
  }
```

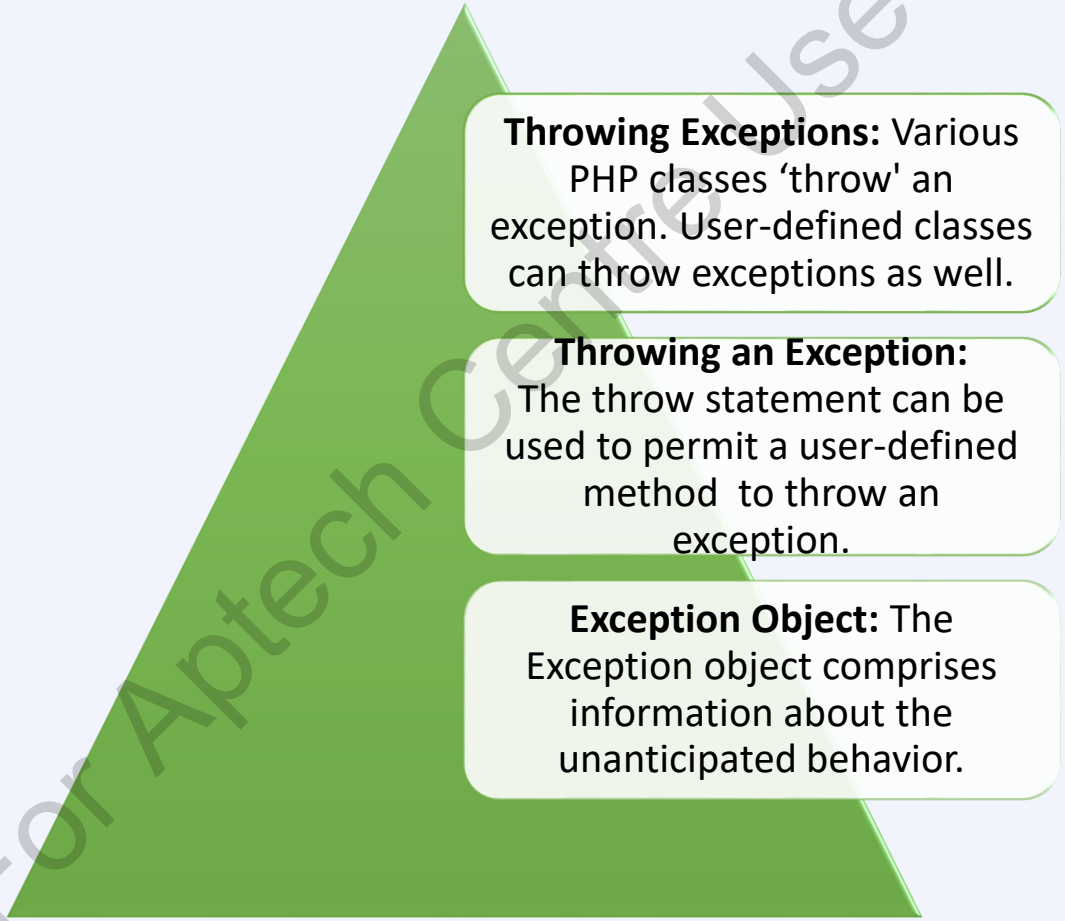
Exception Handling [4-5]

- **`try...catch...finally` Statement** Users can also use the `try...catch...finally` statement to catch exceptions.
- The code written in a `finally` block will always get executed irrespective of whether an exception is caught or not. The `catch` block is optional if the `finally` block already exists and catching the exception is not a priority.

Syntax:

```
try {
code that may cause exceptions
}
catch(Exception $e) {
code that gets executed after catching an exception
  }
finally
{
 code that always gets executed regardless of whether an exception was caught or not
  }
```

Exception Handling [5-5]



Parameter Values of Exception Object

Parameters	Description
`Previous`	Optional. This is a string explaining why an exception was thrown.
`Code`	Optional. This is an integer that can be employed to differentiate this exception from other exceptions of the identical type.
`message`	Optional. It is advisable to pass an exception into the message parameter if that exception gets thrown in a catch block of some other exception.

Methods in Exception Handling

Method	Description
`getPrevious()`	This method returns a previous exception if the exception gets activated by another one.
`getFile()`	This method returns the complete path of a file in which the exception gets thrown.
`getMessage()`	This method returns a string explaining the reason for throwing the exception.
`getCode()`	This method returns the exception code.
`getLine()`	This method returns the line number of the code where the exception is thrown.

Summary

- PHP file handling system enables users to perform tasks including writing, appending, creating, closing, and deleting files.
- PHP has various functions such as `unlink()`, `fwrite()`, `fopen()`, `fclose()`, `fread()`, `fgets()`, `fgetc()`, and so on for file handling.
- It is essential to stay cautious while manipulating files. Any mistakes can lead to errors such as editing a wrong file, deleting a file's content, and filling a hard disk with garbage data.
- Error handling in PHP is a process of identifying errors in a code and tackling those errors attentively.
- Exception handling model in PHP offers better control over errors. The main keywords for exception handling are try, throw, catch, and finally.