Method Overloading কি?

ধরুন আপনার একটা class আছে DB নামে কিন্তু এর whereUsername() নামে কোনো মেথড নেই কিন্তু আপনি বাহির থেকে এই নামের মেথডটিকে কল করলেন এমতাবস্থায় যে পরিস্থিতির সৃষ্টি হবে তাকেই Method Overloading বলে। অর্থাৎ class এ absent যেকোনো মেথডকে কল করাই হলো Method Overloading. যেমনঃ

**class** **DB**

{

}

$db = **new** DB;

**echo** $db->whereUsername('alien');

**Output:**

enter image description here

তো এই রকমের পরিস্থিতি আমরা কিভাবে হ্যান্ডল করতে পারি, হ্যা ঠিক এর জন্যই আমরা ব্যবহার করবো

\_\_call()

\_\_call() হলো PHP এর আরেকটি জনপ্রিয় Magic Method যার দুটি প্যারামিটার \_\_call($method, $agrs). এই মেথডটি তখনই ট্রিগার হবে যখন কেউ class এর কোনো absent অথবা অস্তিত্বহীন কোনো মেথড কে কল করবে। তাহলে একটি উদাহরন দেখা যাকঃ আমরা আগের উদাহরনের উপরেই এটি দেখবো

**class** **DB**

{

**public** **function** **\_\_call**($method, $args)

{

**echo** 'You call the method: '.$method;

}

}

$db = **new** DB;

**echo** $db->whereUsername('alien');

**Output:**

You call the method: whereUsername

অর্থাৎ যখনই আমি whereUsername() মেথডটি কল করেছি তখনই PHP’র ইন্টারপ্রিটার দেখলো যে এই নামে কোনো মেথড আমাদের DB ক্লাসে নেই তাই ও সাথে সাথে \_\_call() মেথডটিকে ট্রিগার করেছে। এর মানে দাড়ালো PHP যখন কোনো absent মেথড কলের রিকুয়েষ্ট পায় তখন একটি ইভেন্ট ফায়ার হয় যেটি \_\_call()মেথডটি কে এক্সিকিউট করে। কি, ব্যপারটা মজার না? আরেকটু মজা দেয়ার চেষ্টা করি, যারা Laravel এ কাজ করছেন তারা হয়তোবা Magic Wheres এ কাজ করেছেন। অর্থাৎ আপনি যদি লারাভেলে User::where('first\_name', 'Nahid')->get() রান না করে যদি User::whereFirstName('Nahid')প্রোগ্রামটাও রান করেন তাও একই কথা কারন লারাভেল এটি \_\_call() Magic Method দিয়ে হ্যান্ডল করে। আসোলে লারাভেলে whereEmail() নামে কোনো মেথডই নেই। তাহলে চলুন আমরাও \_\_call() method টি ব্যবহার করে একটি এরকমের প্রোগ্রাম তৈরী করিঃ

**class** **DB**

{

**protected** $table;

*//Handle Absent Method*

**public** **function** **\_\_call**($method, $args)

{

**if**(substr($method, 0, 5) == 'where') {

$column = substr($method, 4);

$column = $this->camelCaseToUnderscore($column);

**echo** 'SELECT \* FROM ' .$this->table. ' where '.$column.'='.$this->makeSqlString($args[0]);

}

}

*// set table name*

**public** **function** **table**($name)

{

$this->table = $name;

**return** $this;

}

*// make camelCase to snake\_case*

**protected** **function** **camelCaseToUnderscore**($input) {

preg\_match\_all('!([A-Z][A-Z0-9]\*(?=$|[A-Z][a-z0-9])|[A-Za-z][a-z0-9]+)!', $input, $matches);

$ret = $matches[0];

**foreach** ($ret **as** &$match) {

$match = $match == strtoupper($match) ? strtolower($match) : lcfirst($match);

}

**return** implode('\_', $ret);

}

*// make sql string based on datatype*

**protected** **function** **makeSqlString**($val)

{

$types = ['integer', 'double', 'float'];

**if**(in\_array(gettype($val), $types)) {

**return** $val;

}

**return** "'{$val}'";

}

}

$db = **new** DB;

$db->table('tbl\_users')->whereFirstName('Nahid');

Output:

SELECT \* FROM tbl\_users where first\_name='Nahid'

খেয়াল করুন আমাদের ক্লাসে কোনো whereFirstName() নামে মেথড নেই তারপরেও এটি কল করতে পেরেছি এবং তা এক্সিকিউটও করেছে। আর এটি সম্ভব হয়েছে শুধু \_\_call() মেথডের জন্য।

বিঃদ্রঃ এখানে camelCaseToUnderscore() ও makeSqlString() মেথড দুটো দেখে বিভ্রান্ত হবার কারন নেই এটি শুধু প্রোগ্রামটি কে বাস্তবিক করার জন্য ব্যবহার করেছি এর সাথে Magic Method এর কোনো সম্পর্ক নেই

Method Overloading হ্যান্ডলের জন্য আরেকটি ক্লাস হচ্ছে

\_\_callStatic()

এ মেথডটি মূলত static মেথড হ্যান্ডলের জন্য ব্যবহার করা হয়। যেমনঃ আপনি যদি একটি ক্লাস থেকে কোনো মেথডকে statically কল করেন এবং সেই ক্লাসে ঐ মেথডটির অস্তিত্ব না থাকে তবে যে মেথডটি automatic এক্সিকিউট হয় সেটিই হলো \_\_callStatic()।

**class** **DB**

{

**public** **static** **function** **\_\_callStatic**($method, $args)

{

**echo** 'You call the static method: '.$method;

}

}

DB::hello();

**Output:**

You call the static method: hello

তাহলে কি ঘটলো এখানে? হুম এই ক্লাসে এরকমের কোনো static মেথড নেই তাই PHP অটোমেটিক \_\_callStatic() মেথডটিকে এক্সিকিউট করেছে।

এখানে মনে রাখা খুব জরুরী যে \_\_callStatic() মেথডটি কে অবশ্যই statically কল করতে হবে এবং অবশ্যই স্টেটিক মেথড কল করার ক্ষেত্রে উক্ত ক্লাসের কোনো নন-স্টাটিক মেথড কে কল করা যাবে না

এই মেথডটি ব্যবহৃত হয় Laravel Facades এ। আমরা এখন চেষ্টা করবো এরকমের কিছু একটা করার।

আমরা প্রথমে একটি ফাইল নিব calculator.php নামে এবং এখানে একটি ক্লাস Calculator তৈরী করবো এবং এতে কিছু public method লিখবো নিচের মতো করে

*//calculator.php*

**class** **Calculator**

{

**public** **function** **sum**($a, $b)

{

**return** $a+$b;

}

**public** **function** **sub**($a, $b)

{

**return** $a-$b;

}

**public** **function** **square**($a)

{

**return** $a\*$a;

}

}

এবার আমরা আরেকটি ফাইল নিবো static\_calculator.php এবং এখানে calculator.php file টিকে include করবো এবং একটি class লিখবো যাতে \_\_callStatic() মেথড দ্বারা Calculator ক্লাসকে স্টাটিক্যালি হ্যান্ডল করবো নিচের মতো করে

*// static\_calculator.php*

**require** 'calculator.php';

**class** **StaticCalculator**

{

**public** **static** **function** **\_\_callStatic**($method, $args)

{

$calculator = **new** Calculator;

**if**(method\_exists($calculator, $method)) {

**return** call\_user\_func\_array([$calculator, $method], $args);

}

}

}

**echo** StaticCalculator::square(5);

**Output:**

25

দেখুন যখনই আমরা StaticCalculator::square() কে কল করেছি তখনই ও Calculator ক্লাস থেকে এর square() নামের মেথডকে কল করেছে। এটাই হলো \_\_callStatic() এর মজা। আর এভাবেই Laravel এর Facade কাজ করে।