Reflection Capabilities

Introduction

- The PHP Reflection API includes various methods and objects we can use to inspect PHP code during runtime.

  This capability can be very useful especially for reverse engineering, generating documentation and determining whether a specific functionality is available or not.

Introduction

- The `Reflector` interface is implemented by all reflection classes. The `Reflector` interface defines the shared functionality between all reflection classes.

  It includes the definition for two methods: _toString() and export().

- The `ReflectionException` extends `Exception` and is thrown by the various functions the reflection classes include.

The Reflection Classes

- The reflection classes include the following:

  ReflectionFunction

  ReflectionParameter

  ReflectionClass

  ReflectionObject

  ReflectionMethod

  ReflectionProperty

  ReflectionExtension

The `ReflectionFunction` Class

- This class represents a function. For every function in our PHP script we can get a ReflectionFunction object that represents it.

  Once we get a ReflectionFunction object we can call various methods on it and get detailed information about the function it represents.

The `ReflectionFunction` Class

```php
<?php

function doSomething()
{
    echo "<B>Do Something</B><BR>";
}

function sayHello($ob = "Students")
{
    echo "<B>Hello "+$ob+"</B>";
}

function sayGoodMorning($ob = "Students")
{
    echo "<B>Good Morning "+$ob+"</B>";
}

$funcs = get_defined_functions();
```

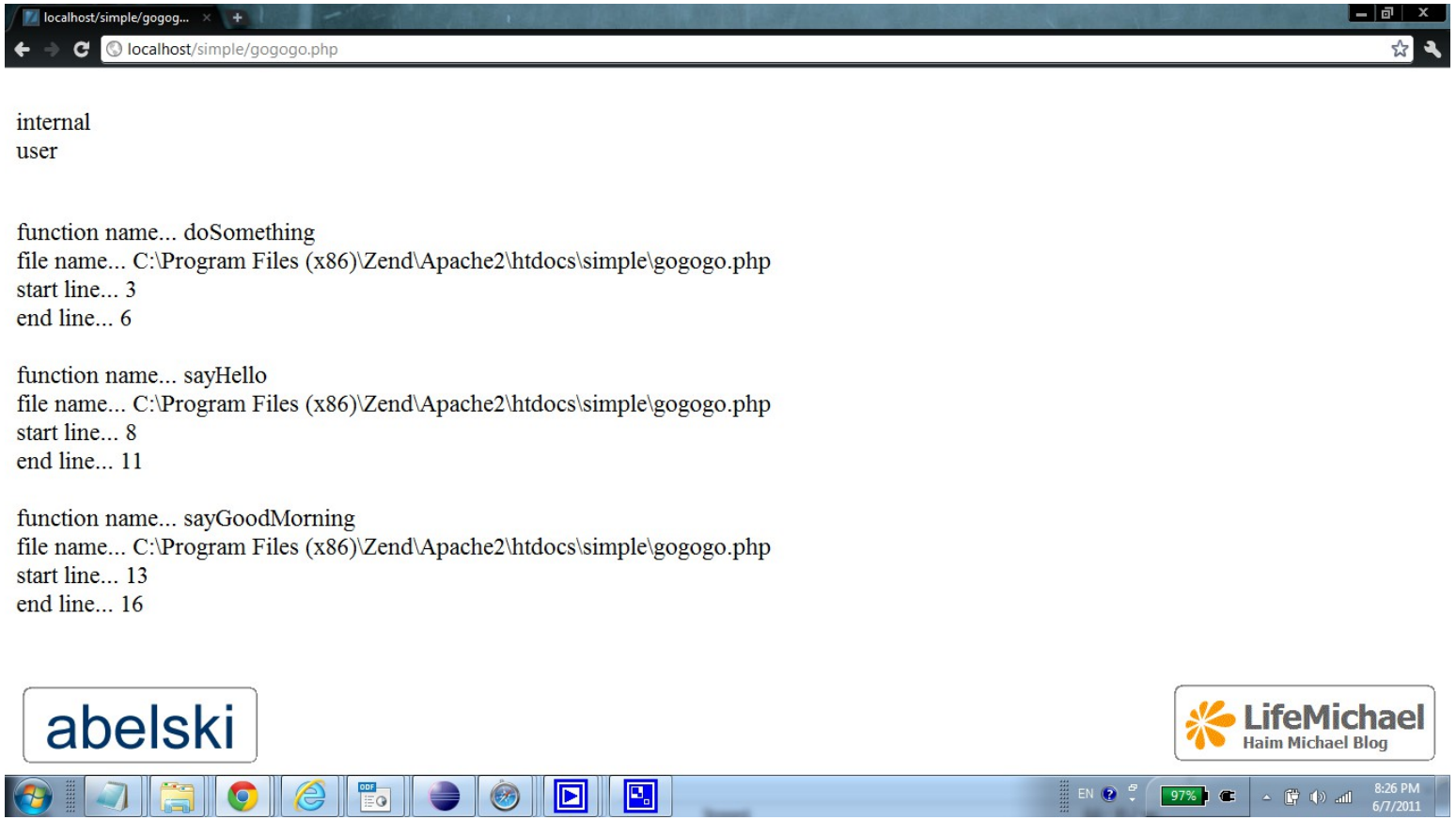
The `ReflectionFunction` Class

```
foreach($funcs as $k=>$v)
{
    echo "<br>$k";
}

echo "<p>";

foreach($funcs['user'] as $function)
{
    try
    {
        $ob = new ReflectionFunction($function);
        echo "<BR>function name... ".$ob->getName();
        echo "<BR>file name... ".$ob->getFileName();
        echo "<BR>start line... ".$ob->getStartLine();
        echo "<BR>end line... ".$ob->getEndLine();
        echo "<BR>";
    }
    catch(ReflectionException $exception)
    {
        echo "<BR><B>Exception Caught</B><BR>";
    }
}
?>
```

Sample



The `ReflectionParameter` Class

- Objects of this class represent parameters.

  Once we get a ReflectionParameter object we can call various methods on it and get detailed information about the parameter it represents.

The `ReflectionParameter` Class

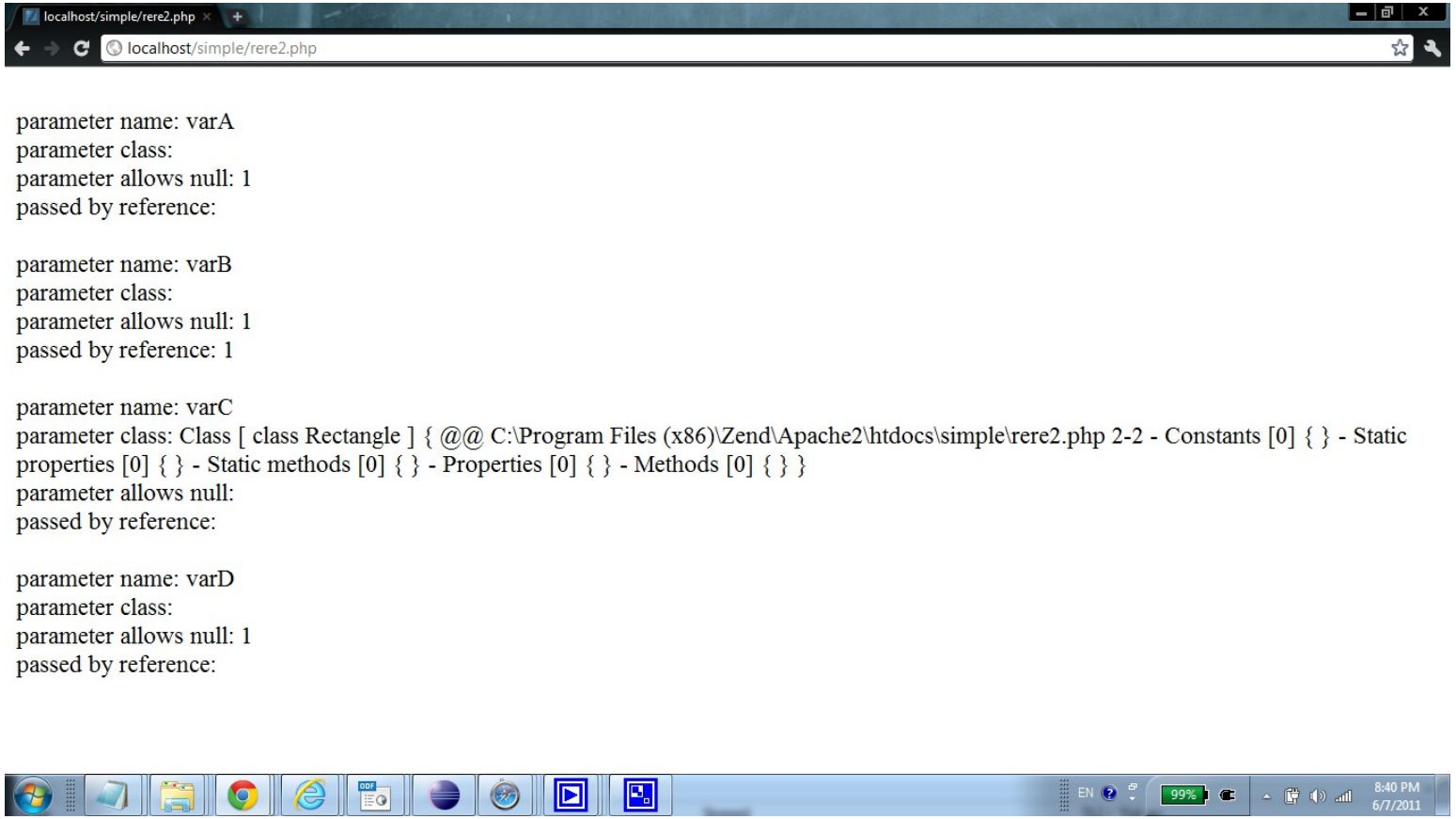
```php
<?php
class Rectangle {}

function doSomething($varA, &$varB, Rectangle $varC, $varD=12)
{
    echo "<br>something<br>";
}

$ob = new ReflectionFunction("doSomething");

foreach ($ob->getParameters() as $index => $param)
{
    echo "<br>parameter name: ".$param->getName();
    echo "<br>parameter class: ".$param->getClass();
    echo "<br>parameter allows null: ".$param->allowsNull();
    echo "<br>passed by reference: ".$param->isPassedByReference();
    echo "<br>";
}
?>
```

The `ReflectionParameter` Class



The `ReflectionClass` Class

- Objects of this class represent specific classes.

  Once we get a ReflectionClass object we can call various methods on it and get detailed information about the class it represents.

The `ReflectionClass` Class

```php
<?php

class Rectangle
{
    private $width;
    private $height;
    function Rectangle($valW,$valH)
    {
        $this->setHeight($valH);
        $this->setWidth($valW);
    }
    function area()
    {
        return $this->width * $this->height;
    }
    function setWidth($val)
    {
        if($val>0)
        {
            $this->width = $val;
        }
    }
```

The `ReflectionClass` Class

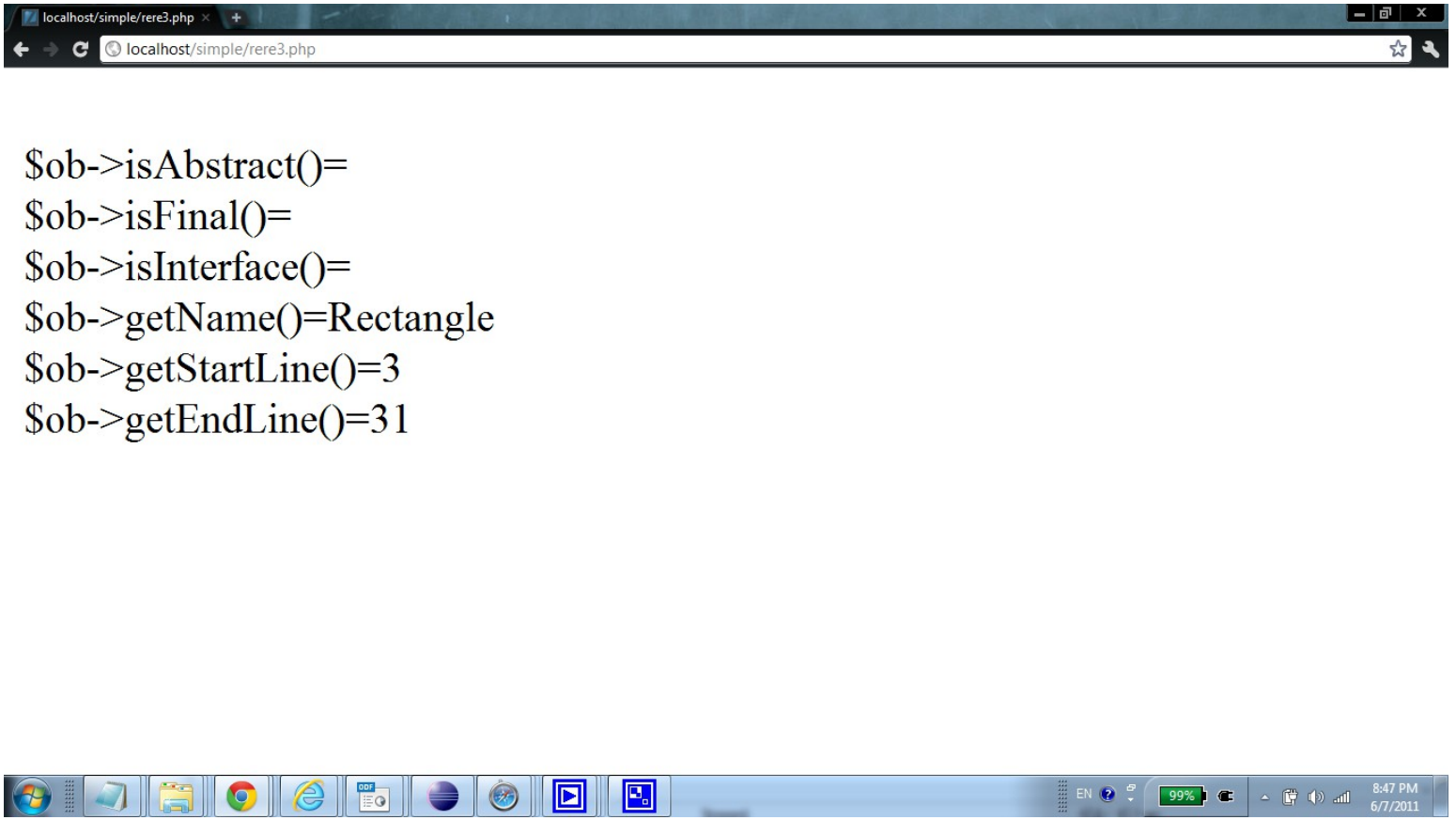
```
    function setHeight($val)
    {
        if($val>0)
        {
            $this->height = $val;
        }
    }
}

$ob = new ReflectionClass("Rectangle");

echo "<br>\$ob->isAbstract()=".$ob->isAbstract();
echo "<br>\$ob->isFinal()=".$ob->isFinal();
echo "<br>\$ob->isInterface()=".$ob->isInterface();
echo "<br>\$ob->getName()=".$ob->getName();
echo "<br>\$ob->getStartLine()=".$ob->getStartLine();
echo "<br>\$ob->getEndLine()=".$ob->getEndLine();

?>
```

The `ReflectionClass` Class



The `ReflectionMethod` Class

- Objects of this class represent specific methods.

  Once we get a ReflectionMethod object we can call various methods on it and get detailed information about the method it represents.

The `ReflectionMethod` Class

```php
<?php
class Rectangle
{
    private $width;
    private $height;
    function Rectangle($valW,$valH)
    {
        $this->setHeight($valH);
        $this->setWidth($valW);
    }
    function area()
    {
        return $this->width * $this->height;
    }
    function setWidth($val)
    {
        if($val>0)
        {
            $this->width = $val;
        }
    }
```

The `ReflectionMethod` Class

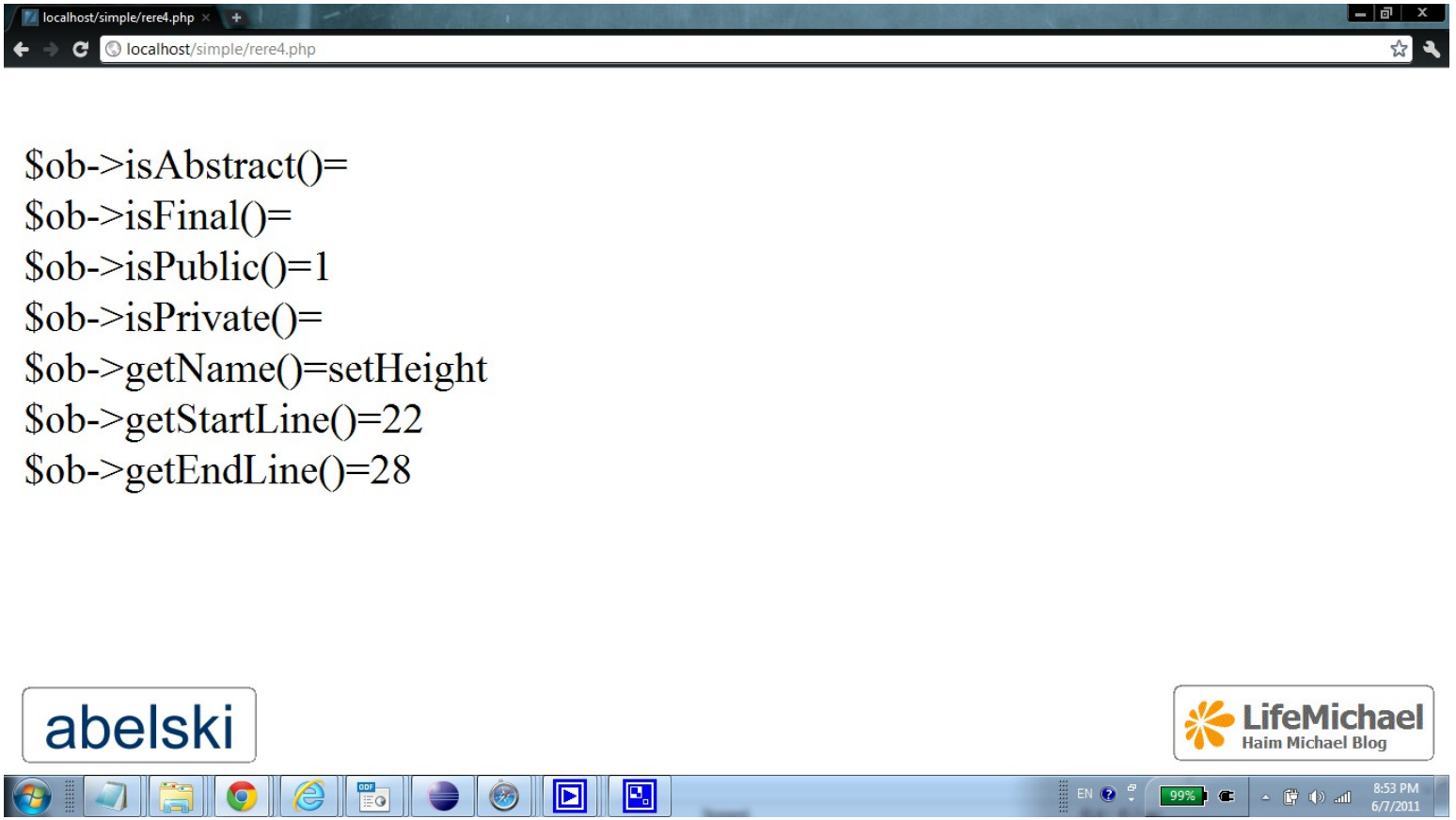
```
    function setHeight($val)
    {
        if($val>0)
        {
            $this->height = $val;
        }
    }
}

$ob = new ReflectionMethod('Rectangle','setHeight');

echo "<br>\$ob->isAbstract()=".$ob->isAbstract();
echo "<br>\$ob->isFinal()=".$ob->isFinal();
echo "<br>\$ob->isPublic()=".$ob->isPublic();
echo "<br>\$ob->isPrivate()=".$ob->isPrivate();
echo "<br>\$ob->getName()=".$ob->getName();
echo "<br>\$ob->getStartLine()=".$ob->getStartLine();
echo "<br>\$ob->getEndLine()=".$ob->getEndLine();

?>
```

The `ReflectionMethod` Class



The `ReflectionProperty` Class

- Objects of this class represents specific a property.

  Once we get a ReflectionProperty object we can call various methods on it and get detailed information about the property it represents.

The `ReflectionProperty` Class

```
<?php
class Rectangle
{
    private $width;
    private $height;
    function Rectangle($valW,$valH)
    {
        $this->setHeight($valH);
        $this->setWidth($valW);
    }
    function area()
    {
        return $this->width * $this->height;
    }
    function setWidth($val)
    {
        if($val>0)
        {
            $this->width = $val;
        }
    }
```

The `ReflectionProperty` Class

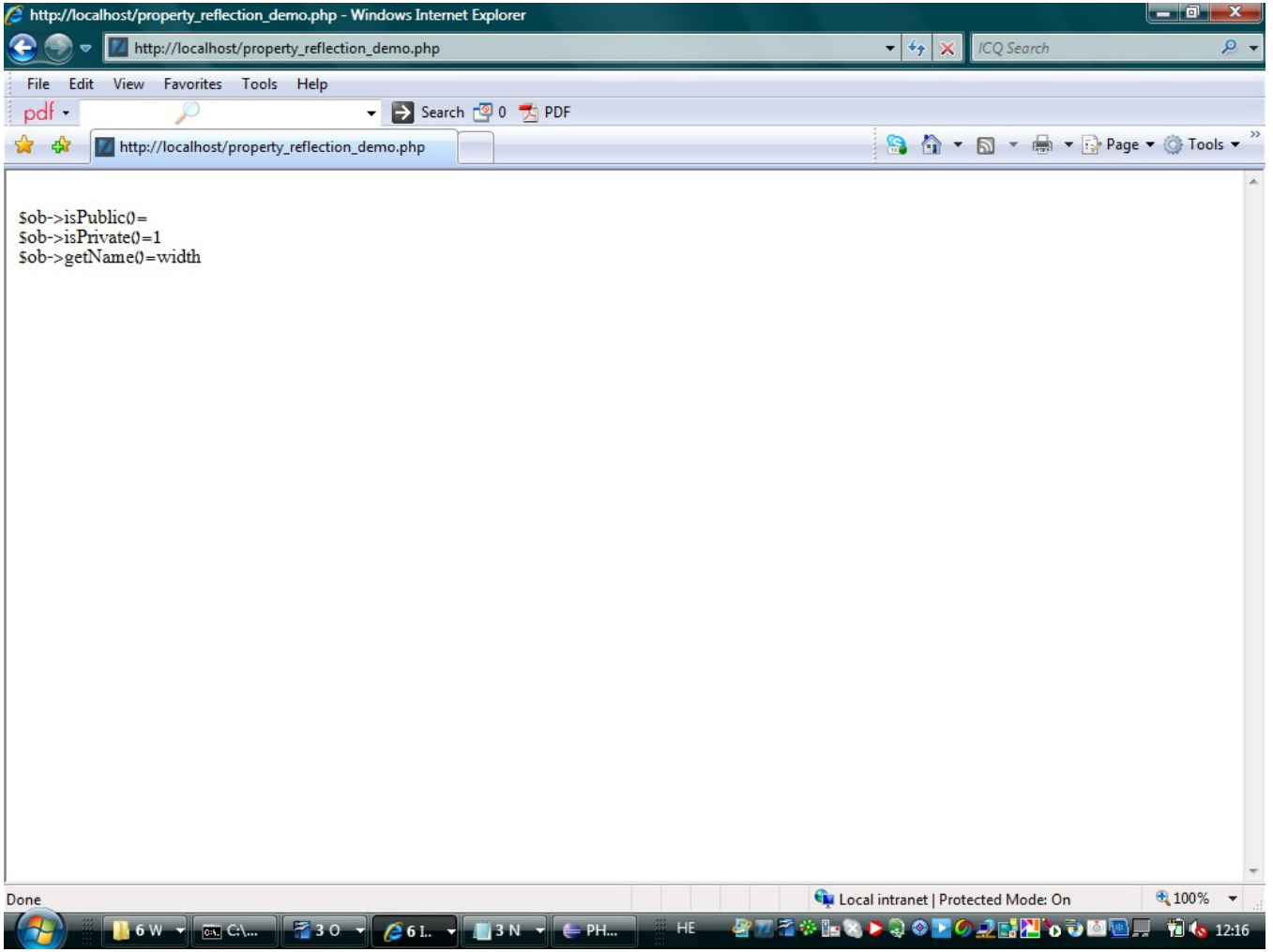
```
    function setHeight($val)
    {
        if($val>0)
        {
            $this->height = $val;
        }
    }
}

$ob = new ReflectionProperty('Rectangle','width');

echo "<br>\$ob->isPublic()=".$ob->isPublic();
echo "<br>\$ob->isPrivate()=".$ob->isPrivate();
echo "<br>\$ob->getName()=".$ob->getName();

?>
```

The `ReflectionProperty` Class



The `ReflectionExtension` Class

- Objects of this class represent extensions installed on our PHP execution environment.

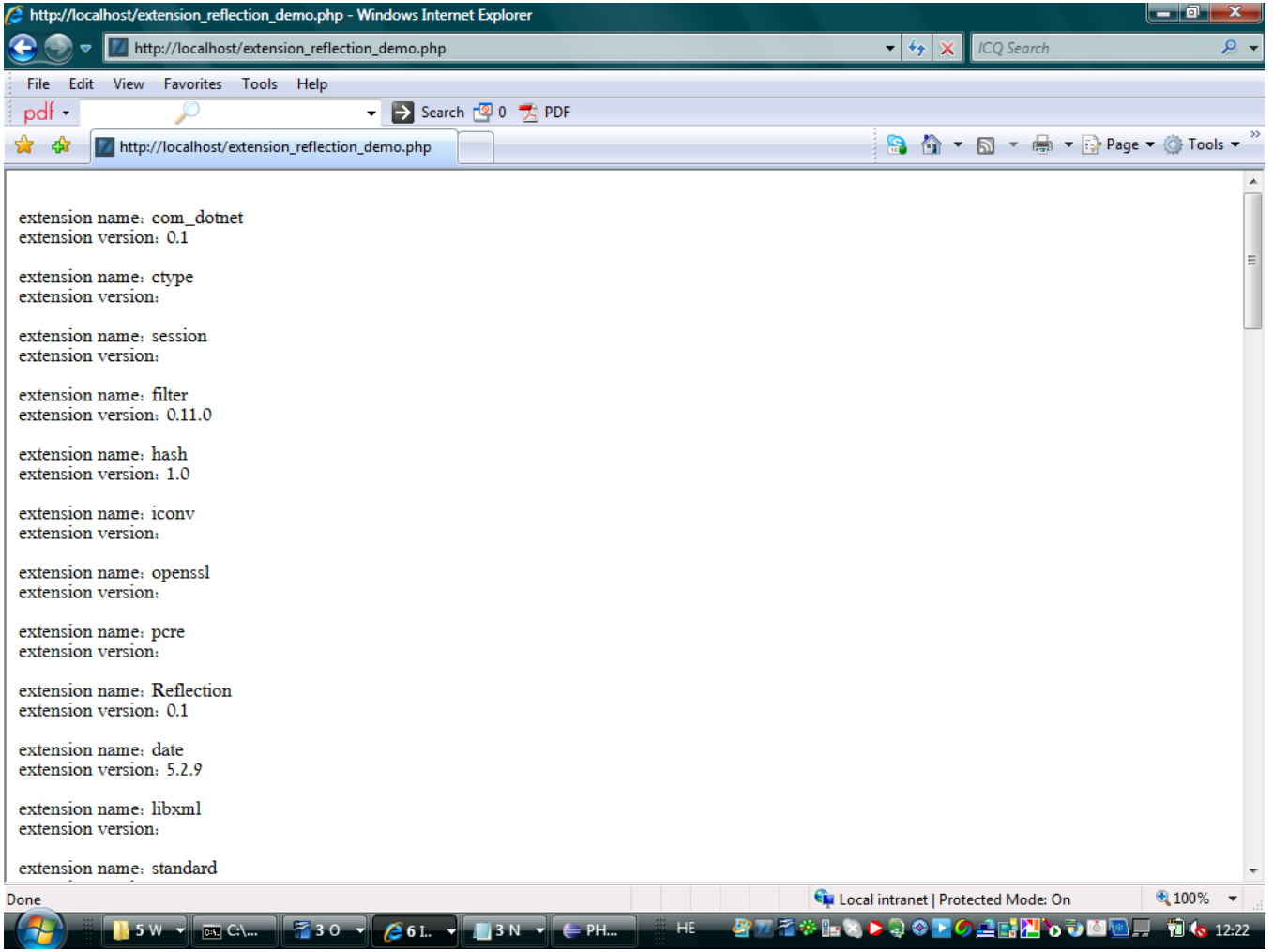
  Once we get a ReflectionExtension object we can call various methods on it and get detailed information about the extension it represents.

The `ReflectionExtension` Class

```
<?php
$extensions = get_loaded_extensions();

foreach($extensions as $key => $value)
{
    $ob = new ReflectionExtension($value);
    echo "<br>extension name: ".$value;
    echo "<br>extension version: ".$ob->getVersion();
    echo "<br>";
}
?>
```

The `ReflectionExtension` Class



The `ReflectionExtension` Class

- Objects of this class represent extensions installed on our PHP execution environment.

  Once we get a ReflectionExtension object we can call various methods on it and get detailed information about the extension it represents.

The `::class` Keyword

- When adding ::class to a class name we will get the fully qualified name of that class.

The `::class` Keyword

```
<?php
namespace com\lifemichael\samples;
class Rectangle{}
echo Rectangle::class;
?>
```

The `::class` Keyword