What is an X-Object?

The term X-Objects came from the fact that the original framework heavily used XML as a source language for Views and other controls. In fact, you can still save your views and view snippets as XML today, but that's no longer encouraged, and best to stick with PHP.

An **X-Object**, simply put, is a displayable web component, that has some kind of instantiated logic behind it, usually in the form of a PHP Class. In terms of code, the X-Object consists of 3 things:

1. A View file (usually as PHP)
2. A Web Component (like the Model)
3. The built-in class x_object to tie those together (The Controller)

You can display a web object easily anywhere on your page:

```
<?=(string) x_object::create('my-view-name')->html();?>
```

In the above example, we've created /app/views/my-view-name.php as the view for this component.

Here's a more interesting example:

```
<?php
$object = new my_class();
$object->member = "value";
// first syntax
echo (string) x_object::create('my-object-view')->html($object);
// second syntax
echo $object->html('my-object-view');
?>
```

In the above example, we've tied an object to the view, and any of its publicly available members can be displayed. Here's what my-object-view.php might look like:

```
<?php
global $business_object;  // grants access to the object
?>
<div class='my-object'><?=(string)$business_object->member?> <!-- shows 'value' as defined above-->
```

Now we can leverage a few built in X-Objects classes to make things even easier.

First off we can create a displayable object class, to tie this together, as:

```
<?php
class my_web_component extends xo_displayable_object {
	public function __construct(){
		$this->view = 'my-object-view';
		$this->object = new my_class();
	}
}
?>
```

Now anywhere in our views we can display it easily:

```
<?=(string) new my_web_component()?>
```

Obviously, to make it a more useful example, we would probably do more in the constructor, and pass along some parameters. But you get the idea.