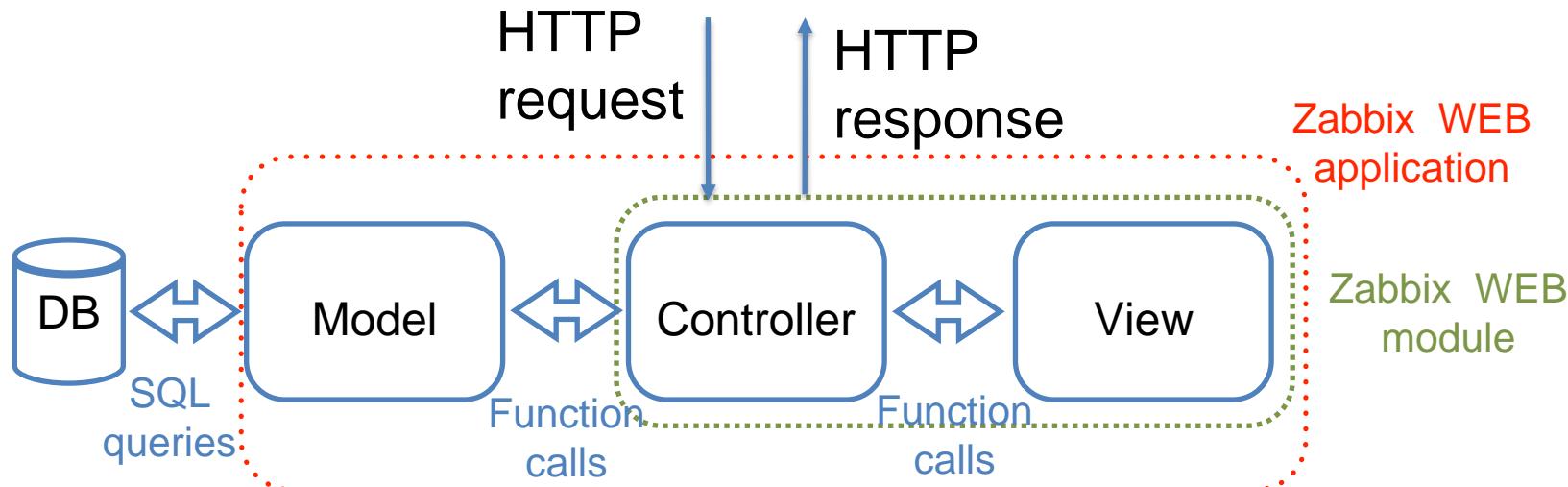


Main principles of operations:

- OpenSource whenever it makes sense
- Automation whenever it is possible
- Zabbix ~~whenever it is...~~ always ok... 99% cases
- We do things properly or don't engage at all

Model - Controller - View (MVC)

Browser: 192.168.148.200/zabbix/zabbix.php?action=dashboard.view



Zabbix does not use any third party MVC framework

Zabbix Web Modules

Example <https://github.com/BGmot/zabbix-module-hosts-tree>



A screenshot of the Zabbix web interface showing the left-hand navigation sidebar. The sidebar has several sections: Monitoring (Dashboard, Problems, Hosts, Latest data, Maps, Discovery), Services (Services, Inventory, Reports, Configuration). The 'Hosts' section is currently selected, indicated by a blue background. The 'Monitoring' section is expanded, showing 'Applications (1)', 'Backend servers (1)', and 'Databases (3)'. The 'Databases' section is expanded, showing 'MySQL servers (2)' and 'PostgreSQL servers (1)'. The 'Services' section is expanded, showing 'Frontend servers (2)' and 'Zabbix servers (1)'. Each item in the tree view has a small icon next to its name.

Hosts

Name	Interface	Availability	Tags	Problems	data	Problems
▶ Applications (1)						
▶ Backend servers (1)						
▼ Databases (3)			1 2		Iata 4	1
▼ MySQL servers (2)			2		Iata 4	1
MYSQL01	zabbix-agent:10050	ZBX	1		Iata 4	1
MYSQL02	zabbix-agent:10050	ZBX	1		Iata 4	Problems
▼ PostgreSQL servers (1)			1		Iata 4	1
PSQL01	zabbix-agent:10050	ZBX	1		Iata 4	1
▼ Frontend servers (2)			1 1		Iata 4	1
Zabbix FE01	zabbix-agent:10050	ZBX	1		Iata 153	8
Zabbix FE02	zabbix-agent:10050	ZBX	1			
▶ Zabbix servers (1)			8			



<https://bgmot.com>

Zabbix Web Modules

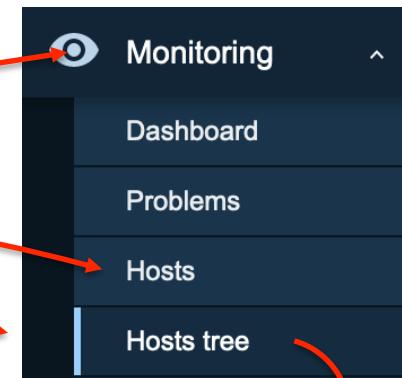
Main menu



- Your module must be a child of `Core\CMModule` class. `init()` method of all enabled modules is executed regardless of what menu item is selected so add your menu items defining what actions needs to be executed for this menu item here:

```
# ./modules/zabbix-module-hosts-tree/Module.php
class Module extends \Core\CMModule {
    public function init(): void {
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Monitoring'))
            ->getSubmenu()
                ->insertAfter('Hosts', (new \CMenuItem(_('Hosts tree'))))
                    ->setAction('bghost.view')
    }
}

# cat ./modules/zabbix-module-hosts-tree/manifest.json
{
    "actions": {
        "bghost.view": {
            "class": "CControllerBGHostView",
            "view": "module.monitoring.bghost.view"
        },
    }
}
```



<http://.../zabbix.php?action=bghost.view>

Zabbix Web Modules

Action mapping



Request: 192.168.148.200:8080/zabbix.php?action=host.view

- What needs to be done for every action is defined in so called Router which is an array defined in `./include/classes/mvc/CRouter.php` - for each action we see:
 - controller (a class that will be used to prepare data)
 - layout (in which form to present data generated by controller)
 - view (how to present/show the data to end-user/requestor)

```
private $routes = [
    // action           controller          layout        view
    ...
    'host.view'      =>      ['CControllerHostView',           'layout.htmlpage',   'monitoring.host.view'],

```

- All “built-in” layouts are defined in these files:

```
./app/views/layout.csv.php
./app/views/layout.export.php
./app/views/layout.htmlpage.php
./app/views/layout.javascript.php
./app/views/layout.json.php
./app/views/layout.warning.php
./app/views/layout.widget.php
```

- You can define your own layouts in **views** folder of your module

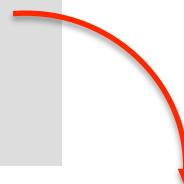
Zabbix Web Modules

Action mapping



- All actions defined in your module's `manifest.json` file are added to the Router:

```
# cat ./modules/zabbix-module-hosts-tree/manifest.json
{
  ...
  "actions": {
    "bghost.view": {
      "class": "CControllerBGHostView",
      "view": "module.monitoring.bghost.view"
    },
  }
}
```



```
private $routes = [
  // action           controller           layout          view
  'host.view'     =>      ['CControllerHostView',   'layout.htmlpage',   'monitoring.host.view'],
  'bghost.view'   =>      ['CControllerBGHostView', 'layout.htmlpage',   'module.monitoring.bghost.view'],
]
```

- Default layout is “`layout.htmlpage`”.
- An action with same name defined in Module **overwrites** action definition in Zabbix.

Zabbix Web Modules



- Name of your controller file must match controller class and must be in `actions/` folder.

```
# cat ./modules/zabbix-module-hosts-tree/manifest.json
{
    ...
    "actions": {
        "bghost.view": {
            "class": "CControllerBGHostView",
            "view": "module.monitoring.bghost.view"
        },
    }
}
```

```
/usr/share/zabbix/modules/zabbix-module-hosts-tree# tree
.
|-- Module.php
|-- actions
|   |-- CControllerBGHost.php
|   |-- CControllerBGHostView.php
|   `-- CControllerBGHostViewRefresh.php
|-- manifest.json
|-- partials
|   |-- js
|   |   '-- monitoring.host.view.refresh.js.php
|   '-- module.monitoring.host.view.html.php
`-- views
    |-- js
    |   '-- monitoring.host.view.js.php
    |-- module.monitoring.bghost.view.php
    '-- module.monitoring.bghost.view.refresh.php
```

```
# ./modules/zabbix-module-hosts-tree/actions/CControllerBGHostView.php
class CControllerBGHostView extends CControllerBGHost {
```

Zabbix Web Modules

Controller



- All Zabbix controllers can be found in `./app/controllers/` folder so as a starting point to work on my module I just copied

`./app/controllers/CControllerHostView.php`

to my module's renaming according to my Controller class name

`./modules/zabbix-module-hosts-tree/actions/CControllerBGHostView.php`

- In my controller classes must be in my own namespace which defined in `manifest.json`:

```
# ./modules/zabbix-module-hosts-tree/manifest.json
{
  "namespace": "BGmotHosts",

# ./modules/zabbix-module-hosts-tree/actions/CControllerBGHostView.php

namespace Modules\BGmotHosts\Actions;

class CControllerBGHostView extends CControllerBGHost {
```

- Inheritance `CControllerBGHostView << CControllerBGHost << CController`



<https://bgmot.com>

Zabbix Web Modules

Controller



- First thing Zabbix does is calling your Controller's `checkInput()` method. It's the place where you can check that all the parameters passed in HTTP request are valid:

```
# ./modules/zabbix-module-hosts-tree/actions/CControllerBGHostView.php
class CControllerBGHostView extends CControllerBGHost {
    protected function checkInput(): bool {
        $fields = [
            'name' => 'string',
            'groupids' => 'array_id',
            'status' => 'in -1,' .HOST_STATUS_MONITORED.' , ' .HOST_STATUS_NOT_MONITORED,
            ...
        ];
        $ret = $this->validateInput($fields);
        return $ret;
    }
}
```

- All possible validation rules are defined in `./include/classes/validators/CNewValidator.php`.

```
# grep "case '" ./include/classes/validators/CNewValidator.php
        case 'not_empty':
        case 'json':
        case 'in':
        ...
        case 'array_id':
        ...
```

Zabbix Web Modules

Controller



- If `checkInput()` method returns **true** (input valid) then `doAction()` of your Controller is called. All data preparation happens here. At the end you need to prepare one massive associative array (usually it is named `$data`) and return it as shown here:

```
# ./modules/zabbix-module-hosts-tree/actions/CControllerBGHostView.php
protected function doAction(): void {
    ...
    $data = [ ... ];
    $response = new CControllerresponseData($data);
    $response->setTitle(_('Hosts'));
    $this->setResponse($response);
}
```

- What you put into associative array `$data` will be available later in your View code.

Zabbix Web Modules

View



- Zabbix constructs **CView** class (defined in `./include/classes/mvc/CView.php`) and then simply calls its `getOutput()` method which in turn performs PHP include of your view file, i.e. “executing” the file.
- You can put pure HTML/CSS into the view .php file but it's... meh.... Ugly!

```
./views/module.monitoring.bghost.view.php
<table width=300px>
<thead>
<tr>
<th>Name</th>
<th>Interface</th>
<th>Availability</th>
<th>Tags</th>
</tr>
</thead>
</table>
```

A screenshot of the Zabbix web interface. On the left is a sidebar with navigation links: Monitoring (which is active), Dashboard, Problems, Hosts, and Hosts tree. The main content area shows a table with four columns: Name, Interface, Availability, and Tags. A red arrow points from the code snippet above to the 'Name' column header in the table. A red oval highlights the entire row of column headers.

Zabbix Web Modules

View



- ... or you can (and you should!) use standard Zabbix HTML classes

```
<?php
$widget = (new CWidget())
    ->setTitle(_('Hosts'));
$table = (new CTableInfo());
$table->setHeader([
    (new CColHeader(_('Name'))),
    (new CColHeader(_('Interface'))),
    (new CColHeader(_('Availability'))),
    (new CColHeader(_('Tags'))));
];
$widget->addItem($table);
$widget->show();
?>
```

- Look at the source code how to use these classes. See the list of all out-of-the-box classes for different HTML elements here:

```
# ls -1 ./include/classes/html/
CActionButtonList.php
CBarGauge.php
CButton.php
CButtonCancel.php
...
```

The screenshot shows the Zabbix web interface with a navigation bar on the left and a main content area. In the content area, there is a table titled "Hosts" with columns for Name, Interface, Availability, and Tags. The table displays the message "No data found.".



Zabbix Web Modules

View



- All the data you prepared in Controller are available in View.

Controller:

```
# ./modules/zabbix-module-hosts-tree/actions/CControllerBGHostView.php
protected function doAction(): void {
    $data = [ 'hosts_count' => API::Host()->get(['countOutput' => true]) ];
    $response = new CControllerResponseData($data);
    $this->setResponse($response);
}
```

View:

```
# ./views/module.monitoring.bghost.view.php
<?php
$widget = (new CWidget())
    ->setTitle('Hosts (total number of hosts ' . $data['hosts_count'] . ')');
...
```

A screenshot of the Zabbix web interface. On the left, there's a sidebar with the Zabbix logo, 'Composed installation', a search bar, and a 'Monitoring' button. The main content area has a title 'Hosts (total number of hosts 8)'. Below the title is a table with two columns: 'Name' and 'Interface'. There is one row in the table, but its content is mostly obscured by a red arrow pointing from the code above. A red arrow also points from the code line '\$data['hosts_count']' to the '8' in the title.



<https://bgmot.com>

Zabbix Web Modules

View - CSS and JavaScript



- If you want to add some CSS files to the page do it in your View code this way:

```
# ./views/module.monitoring.bghost.view.php  
$this->addCssFile('modules/zabbix-module-hosts-tree/views/css/mycool.css');
```

Your CSS file should not contain <style> tags, just pure CSS.

- You can use `addJsFile()` method only to add JS files that come with Zabbix and located in root `/js` folder, e.g.:

```
# ./views/module.monitoring.bghost.view.php  
$this->addJsFile('multiselect.js');
```

- To include your JavaScript code use this function, note this `.php` file, not JS.

```
# ./views/module.monitoring.bghost.view.php  
$this->includeJsFile('monitoring.host.view.js.php', $data);
```

- this `.php` file will be searched by Zabbix in the `./views/js` subdirectory of your module and must “print” `JavaScript` code.
- A copy of `$data` variable will be available for using within the file.

Zabbix Web Modules

View - onBeforeAction() and onTerminate()



- Two special methods you can define in your module's controller (Module.php): `onBeforeAction()` and `onTerminate()`.
- `onBeforeAction()` is executed before code execution passed to your module's controller and whatever you “print” in this method will be put **directly after `<head>`** tag in final HTML page returned to browser.
- `onTerminate()` is executed the last (right before PHP interpreter exist). You can access to data generated in Controller (`$data`) using this code:

```
# ./Module.php
public function onTerminate(CAction $action): void {
    $action_name = $action->getAction();
    if ($action_name === 'bghost.view') {
        $data = $action->getResponse()->getData();
```

Whatever you “print” in `onTerminate()` method will be put **right before closing `</body>`** tag in final HTML page returned to browser.

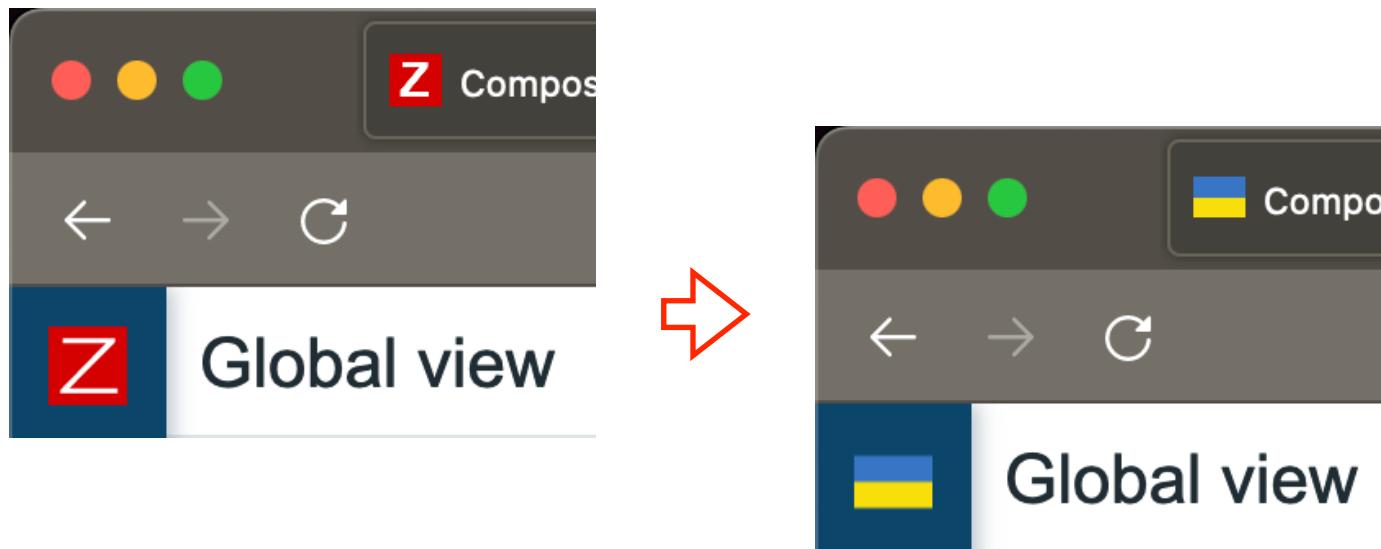
Zabbix Web Modules

View - onBeforeAction() and onTerminate()



- BeforeAction() and onTerminate() methods of all enabled modules executed regardless of menu item selected by end user so you can change Zabbix UI globally. I used this “feature” in my module that does not have any Controllers and Views:

<https://github.com/BGmot/zabbix-module-peace>



Zabbix Web Modules



My modules: <https://github.com/stars/BGmot/lists/zabbix-modules>

Full modules with Controllers and Views (add additional menu items):

- <https://github.com/BGmot/zabbix-module-hosts-tree>
- <https://github.com/BGmot/zabbix-module-avail-report>
- <https://github.com/BGmot/zabbix-module-latest-data>

Very simple modules (affect Zabbix UI as a whole):

- <https://github.com/BGmot/zabbix-module-menu>
- <https://github.com/BGmot/zabbix-module-peace>

Join us on Telegram <https://t.me/ZabbixTech>