Misusing OOP in mvc frameworks

How to conveniently develop broken apps
Disclaimer

- No new super-puper attack techniques
- No new mind-blowing vulnerabilities
- Not about OOP misuse in general, but rather a single OOP principle
Disclaimer

- No new super-puper attack techniques
- No new mind-blowing vulnerabilities
- Not about OOP misuse in general, but rather a single OOP principle
- Which is Inheritance. in MVC frameworks.
No new super-puper attack techniques
No new mind-blowing vulnerabilities
Not about OOP misuse in general, but rather a single OOP principle
Which is Inheritance. in MVC frameworks.

Model–View–Controller (usually known as MVC) is a software design pattern\(^\text{[1]}\) commonly used for developing user interfaces which divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the use

(c) wikipedia
MVC in 3 words

Controller

View (template)

Model

User
MVC in 3 words

Request → URL matching → Controller

Routing table:

<table>
<thead>
<tr>
<th>URL</th>
<th>Handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>url1</td>
<td>handler1</td>
</tr>
<tr>
<td>url2</td>
<td>handler2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>urlN</td>
<td>handlerN</td>
</tr>
</tbody>
</table>
Disclaimer

- No new super-puper attack techniques
- No new mind-blowing vulnerabilities
- Not about OOP misuse in general, but rather a single OOP principle
- Which is **Inheritance** in **MVC** frameworks.

In **object-oriented programming**, **inheritance** is the mechanism of basing an **object** or **class** upon another object (**prototype-based inheritance**) or class (**class-based inheritance**), retaining similar implementation.

(c) wikipedia
Inheritance is the thing

Private Controller

PrivateAPI

- stuff

# fetchStuff
+ getStuff
+ editStuff
+ deleteStuff

Allows authorized users to manipulate stuff
Inheritance is the thing

Private Controller

<table>
<thead>
<tr>
<th>PrivateAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>- stuff</td>
</tr>
<tr>
<td># fetchStuff</td>
</tr>
<tr>
<td>+ getStuff</td>
</tr>
<tr>
<td>+ editStuff</td>
</tr>
<tr>
<td>+ deleteStuff</td>
</tr>
</tbody>
</table>

Allows authorized users to manipulate stuff

Now you want to show some public stuff to anonymous users as well

- But you don't want to re-invent code
Inheritance is the thing

**Private Controller**

<table>
<thead>
<tr>
<th>PrivateAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>stuff</td>
</tr>
</tbody>
</table>

```python
# fetchStuff
+ getStuff
+ editStuff
+ deleteStuff
```

**Public Controller**

<table>
<thead>
<tr>
<th>PublicAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ getPublicStuff</td>
</tr>
</tbody>
</table>

Inherits
Inheritance is the thing

Now we can utilize protected methods of a parent class

<table>
<thead>
<tr>
<th>PublicAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>- stuff</td>
</tr>
</tbody>
</table>

# fetchStuff
+ getPublicStuff
Inheritance is the thing

Now we can utilize protected methods of a parent class
• But also all the other methods as well

<table>
<thead>
<tr>
<th>PublicAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>- stuff</td>
</tr>
<tr>
<td># fetchStuff</td>
</tr>
<tr>
<td>+ getPublicStuff</td>
</tr>
<tr>
<td>+ getStuff</td>
</tr>
<tr>
<td>+ editStuff</td>
</tr>
<tr>
<td>+ deleteStuff</td>
</tr>
</tbody>
</table>
How to find
(or more like how do I find it. Which is probably the worst way of doing it)

• Gather a list of registered routes
  - Using debug and monitoring features of frameworks

• Parse the list for similar endpoints
  Such as:
  - /admin/whatever/endpoint
  - /public/whatever/endpoint

• Try to access similar routes with the same user role
Examples

The above can turn out to be really sad, as the examples below might (or might not) show
Examples. PHP + Symfony

Private Controller
- Has action mapped to:

Controller prefix + action route = /admin/user-list
Examples. PHP + Symfony

Private Controller

- Has action mapped to:

  Controller prefix + action route = /admin/user-list

- Has access control policies properly set:

  access to /admin is only allowed to users with role ROLE_ADMIN
Examples. PHP + Symfony
Examples. PHP + Symfony

Public Controller
- Inherited from Private controller (because reasons)
Public Controller

- Inherited from Private controller (because reasons)

- Has access control policies properly set:

```
access_control:
- { path: '/admin', roles: [ROLE_ADMIN] }
- { path: '/user', roles: [ROLE_USER] }
- { path: '/security', roles: [IS_AUTHENTICATED_ANONYMOUSLY] }
```

access to `/user` is allowed to any user with role ROLE_USER
Examples. PHP + Symfony
How to catch Symfony debug command (through bin/console)

```
php bin/console debug:router
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Method</th>
<th>Scheme</th>
<th>Host</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_twig_error_test</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_error/{code}.{_format}</td>
</tr>
<tr>
<td>wdt</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_wd/{token}</td>
</tr>
<tr>
<td>_profiler_home</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/</td>
</tr>
<tr>
<td>_profiler_search</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/search</td>
</tr>
<tr>
<td>_profiler_search_bar</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/search_bar</td>
</tr>
<tr>
<td>_profiler_phpinfo</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/phpinfo</td>
</tr>
<tr>
<td>_profiler_search_results</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/{token}/search/results</td>
</tr>
<tr>
<td>_profiler_open_file</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/open</td>
</tr>
<tr>
<td>_profiler</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/{token}</td>
</tr>
<tr>
<td>_profiler/router</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/{token}/router</td>
</tr>
<tr>
<td>_profiler_exception</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/{token}/exception</td>
</tr>
<tr>
<td>_profiler_exception_css</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>_profiler/{token}/exception.css</td>
</tr>
<tr>
<td>admin_app_admin_index</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/admin/admin/index</td>
</tr>
<tr>
<td>admin_app_admin_users</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/admin/user-list</td>
</tr>
<tr>
<td>security_register</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/security/register</td>
</tr>
<tr>
<td>security_login</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/security/login</td>
</tr>
<tr>
<td>security_logout</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/security/logout</td>
</tr>
<tr>
<td>user_root</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/user/</td>
</tr>
<tr>
<td>user_app_user_users</td>
<td>ANY</td>
<td>ANY</td>
<td>ANY</td>
<td>/user/user-list</td>
</tr>
</tbody>
</table>
Examples. Java + Spring MVC

Private Controller
- Has action mapped to:

Controller prefix + action route = /private/admin

```java
@RequestMapping("/private")
public class PrivateController {
    @RequestMapping(value = "/profile", method = RequestMethod.GET)
    public ModelAndView profile() {
        ModelAndView model = new ModelAndView();
        model.setViewName("home");
        return model;
    }

    @RequestMapping(value = "/admin", method = RequestMethod.GET)
    @ResponseBody
    public String admin(@RequestParam String cmd) {
        String output = "";
        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader input = new BufferedReader(new InputStreamReader(p.getInputStream()));
            String line;
            while ((line = input.readLine()) != null) {
                output += line + "\n";
            }
            input.close();
        } catch (Exception err) {
            output = "Error executing cmd:" + err;
        }
        return output;
    }
}
```
Examples. Java + Spring MVC

Private Controller

• Has action mapped to:

Controller prefix + action route = /private/admin

• Has access control policies properly set:

access to /private is only allowed to users with ADMIN role
Examples. Java + Spring MVC
Examples. Java + Spring MVC

Public Controller

• Inherited from Private controller (because reasons)

```java
@Controller
@RequestMapping("/public")
public class PublicController extends PrivateController{
    @RequestMapping(value = "/", method = RequestMethod.GET)
    @ResponseBody
    public String echo() {
        return "This is a public page";
    }
}
```
Examples. Java + Spring MVC

Public Controller

- Inherited from Private controller (because reasons)

- Has access control policies properly set
  
  access to /user is allowed to any user with role ROLE_USER
Examples. Java + Spring MVC
### How to catch. Spring boot actuator

http://localhost:8080/mappings

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
</table>
| 26   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello.PrivateController.profile()"
        predicate: "GET /private/profile"
        details: ```|
| 27   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /private/admin"
        details: ```|
| 28   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello.PrivateController.profile()"
        predicate: "GET /public/profile"
        details: ```|
| 29   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /public/admin"
        details: ```|
| 30   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello.PrivateController.profile()"
        predicate: "GET /public/profile"
        details: ```|

---

**Examples. Java + Spring MVC**

---

**How to catch. Spring boot actuator**

http://localhost:8080/mappings

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
</table>
| 26   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello.PrivateController.profile()"
        predicate: "GET /private/profile"
        details: ```|
| 27   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /private/admin"
        details: ```|
| 28   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello/PrivateController/profile()"
        predicate: "GET /public/profile"
        details: ```|
| 29   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /public/admin"
        details: ```|
| 30   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello/PrivateController/profile()"
        predicate: "GET /public/profile"
        details: ```|

---

**Examples. Java + Spring MVC**

---

**How to catch. Spring boot actuator**

http://localhost:8080/mappings

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
</table>
| 26   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello.PrivateController.profile()"
        predicate: "GET /private/profile"
        details: ```|
| 27   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /private/admin"
        details: ```|
| 28   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello/PrivateController/profile()"
        predicate: "GET /public/profile"
        details: ```|
| 29   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /public/admin"
        details: ```|
| 30   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello/PrivateController/profile()"
        predicate: "GET /public/profile"
        details: ```|

---

**Examples. Java + Spring MVC**

---

**How to catch. Spring boot actuator**

http://localhost:8080/mappings

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
</table>
| 26   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello.PrivateController.profile()"
        predicate: "GET /private/profile"
        details: ```|
| 27   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /private/admin"
        details: ```|
| 28   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello/PrivateController/profile()"
        predicate: "GET /public/profile"
        details: ```|
| 29   | ```java
        handler: "public java.lang.String hello.PrivateController.admin(java.lang.String)"
        predicate: "GET /public/admin"
        details: ```|
| 30   | ```java
        handler: "public org.springframework.web.servlet.ModelAndView hello/PrivateController/profile()"
        predicate: "GET /public/profile"
        details: ```|
Examples. Other frameworks
(Well, potential examples to be honest)

- ruby on rails (+ devise + cancan) - to some extent
- node.js (stuff like sails.js)
THANKS FOR ATTENTION

@Osakaaa