MISP CORE DEVELOPMENT CRASH COURSE

HOW I LEARNED TO STOP WORRYING AND LOVE THE PHP

CIRCL / TEAM MISP PROJECT



MISP PROJECT



MISP core development crash course

MISP CORE DEVELOPMENT CRASH
COURSE
HOW LIFAMMED TO STOP WORRYING AND LOVE THE PHP

CIRCL / TEAM MISP PE





SOME THINGS TO KNOW IN ADVANCE...

- MISP is based on PHP 7.3+
- Using the MVC framework CakePHP 2.x
- What we'll look at now will be a quick glance at the structuring / layout of the code

MISP core development crash course

-Some things to know in advance...

Using the MVC framework CakePHP 2.x
 What we'll look at now will be a quick glance at the

■ MISP is based on PHP 7.3+

MVC FRAMEWORKS IN GENERAL

- separation of business logic and views, interconnected by controllers
- main advantage is clear separation of the various components
- lean controllers, fat models (kinda...)
- domain based code reuse
- No interaction between Model and Views, ever

MISP core development crash course

-MVC frameworks in general

2022-

MVC FRAMEWORKS IN GENERAL

separation of business logic and views, interconnected controllers

main advantage is clear separation of the various components

domain based code reuse

m No interaction between Model and Views, e

STRUCTURE OF MISP CORE APP DIRECTORIES

- Config: general configuration files
- Console: command line tools
- Controller: Code dealing with requests/responses, generating data for views based on interactions with the models
- Lib: Generic reusable code / libraries
- Model: Business logic, data gathering and modification
- Plugin: Alternative location for plugin specific codes, ordered into controller, model, view files
- View: UI views, populated by the controller

MISP core development crash course

2022-09-13

-Structure of MISP Core app directories

OF MISP CORE APP DIRECTORIES

g: general configuration files

 Controller: Code dealing with requests/responses, generating data for views based on interactions with

Lib: Generic reusable code / libraries

■ Model: Business logic, data gathering

 Plugin: Alternative location for plugin specific ordered into controller, model, view files

View: UI views, populated by the controller

CONTROLLERS - SCOPE

- Each public function in a controller is exposed as an API action
- request routing (admin routing)
- multi-use functions (POST/GET)
- request/response objects
- contains the action code, telling the application what data fetching/modifying calls to make, preparing the resulting data for the resulting view
- grouped into controller files based on model actions
- Accessed via UI, API, AJAX calls directly by users
- For code reuse: behaviours
- Each controller bound to a model

MISP core development crash course

-Controllers - scope

2022-09

m request routing (admin routing

data for the resulting view

m grouped into controller files based on model action

CONTROLLERS - FUNCTIONALITIES OF CONTROLLERS

- pagination functionality
- logging functionality
- Controllers actions can access functionality / variables of Models
- Controllers cannot access code of other controller actions (kind of...)
- Access to the authenticated user's data
- beforeFilter(), afterFilter() methods
- Inherited code in AppController

MISP core development crash course

2077-09-

-Controllers - functionalities of controllers

CONTROLLERS - FUNCTIONALITIES OF CONTROLLERS

ng functionality

Controllers actions can access functionality / variable Models

 Controllers cannot access code of other controller actio (kind of...)

Access to the authenticated user's dat

m Access to the authenticated user's da m beforeFilter(), afterFilter() methods

oreHiter(), atterHiter() method: erited code in AppController

CONTROLLERS - COMPONENTS

- Components = reusable code for Controllers
 - ► Authentication components
 - ► RestResponse component
 - ► ACL component
 - ► Cidr component
 - ► IOCImport component (should be moved)

MISP core development crash course

2022-

-Controllers - components

■ Components = reusable code for Controllers

CONTROLLERS - ADDITIONAL FUNCTIONALITIES

- Handling API responses (RestResponseComponent)
- Handling API requests (IndexFilterComponent)
- auth/session management
- ACL management
- CRUD Component
- Security component
- important: quertString/PyMISP versions, MISP version handler
- future improvements to the export mechanisms

MISP core development crash course

2022

-Controllers - additional functionalities

future improvements to the export mechanism

MODELS - SCOPE

- Controls anything that has to do with:
 - ► finding subsets of data
 - ► altering existing data
 - ► inherited model: AppModel
 - reusable code for models: Behaviours
 - regex, trim

MISP core development crash course

└─Models - scope

2022-09

MODELS - SCOPE

- Controls anything that has to do with: ► finding subsets of data
- altering existing data
 inherited model: AppMode
- reusable code for models: Beh.
 regex, trim

MODELS - HOOKING SYSTEM

- Versatile hooking system
 - manipulate the data at certain stages of execution
 - code can be located in 3 places: Model hook, AppModel hook, behaviour

MISP core development crash course

–Models - hooking system

Versatile hooking system
 maniputate the data at certain stages of execution

MODEL - HOOKING PIPELINE (ADD/EDIT)

- Hooks / model pipeline for data creation / edits
 - beforeValidate() (lowercase all hashes)
 - validate() (check hash format)
 - ► afterValidate() (we never use it
 - could be interesting if we ever validated without saving)
 - beforeSave() (purge existing correlations for an attribute)
 - afterSave() (create new correlations for an attribute / zmg)

MISP core development crash course

2022-09-13

-Model - hooking pipeline (add/edit)

AODEL - HOOKING PIPELINE (ADD/EDIT)

- Hooks / model pipeline for data creation / edits
 ▶ beforeValidate() (lowercase all hashes)
- ➤ validate() (check hash format)
- afterValidate() (we never use it
 could be interesting if we ever validated without sav
- beforeSave() (purge existing correlations for an attribute.
 afterSave() (greate new correlations for an attribute. / zm

MODELS - HOOKING PIPELINE (DELETE/READ)

- Hooks for deletions
 - ► beforeDelete() (purge correlations for an attribute)
 - ► afterDelete() (zmq)
- Hooks for retrieving data
 - beforeFind() (modify the find parameters before execution, we don't use it)
 - ► afterFind() (json decode json fields)

MISP core development crash course

-Models - hooking pipeline (delete/read)

WODELS - HOOKING PIPELINE (DELETE/READ)

■ Hooks for deletions

 beforeDelete() (purge correlations for an attribute)
 afterDelete() (zmg)

■ Hooks for retrieving data
▶ beforeFind() (modify the find parameters before ex

we don't use it)
➤ afterFind() (json decode json fields)

MODELS - MISC

- code to handle version upgrades contained in AppModel
- generic cleanup/data migration tools
- centralised redis/pubsub handlers
- (Show example of adding an attribute with trace)

MISP core development crash course

-Models - misc

2022-09

(Show example of adding an attribute with trace)

VIEWS - SCOPE AND STRUCTURE

- templates for views
- layouts
- reusable template code: elements
 - attribute list, rows (if reused)
- reusable code: helpers
 - commandhelper (for discussion boards), highlighter for searches, tag colour helper
- views per controller

MISP core development crash course

2022-09

-Views - scope and structure

TIEWS - SCOPE AND STRUCTURE

m templates for views m layouts

reusable template code: elements

 reusable code: neitpers
 commandhelper (for discussion boards), highlight searches, tag colour helper

searches, tag colour he wiews per controller

VIEWS - Types of views and helpers

- ajax views vs normal views
- data views vs normal views vs serialisation in the controller
- sanitisation h()
- creating forms
 - sanitisation
 - ► CSRF

MISP core development crash course

└─Views - Types of views and helpers

OF VIEWS AND HELPERS

VIEWS - GENERATORS

- Mostly in genericElements
- Preparing the move to Cake4
- Important ones
 - ► Form generate forms in a standardised way (/add, /edit, etc)
 - ► IndexTable index lists using Field templates (/index, etc)
 - ► SingleViews key-value lists with child elements (/view, etc)
 - ► Menues to be refactored, see Cerebrate

MISP core development crash course

└─Views - Generators

2022-

Preparing the move to Cake4
Important ones
Form - generate forms in a standardised way (/add, /edit

m Mostly in genericElements

IndexTable - index lists using Field templates (/index, - SingleViews - key-value lists with child elements (/vie - Menues - to be refactored, see Cerebrate

GENERAL REUSABLE LIBRARIES

- Located in app/Lib
- Code that is to be reused across several layers
- Important ones
 - ► Dashboard Dashboard widget backend code
 - ► EventReport Report generation
 - ► Export MISP -> external format converter modules
 - ► Tools List of generic helper libraries examples:
 - Attachment, JSON conversion, random generation, emailing, sync request generation
 - Kafka, ZMQ, AWS S3, Elastic integration, PGP encryption, CIDR operations

MISP core development crash course

2022-09-13

-General reusable libraries

GENERAL REUSABLE LIBRARIES

Located in app/Lib

- e that is to be reused across several lay
- portant ones
- EventReport Report generation
 Export MISP -> external format converter mode
- Export MISP -> external format converter modules
 Tools List of generic helper libraries examples:
 # Attachment, ISON conversion, random generation, en
 - Attachment, JSON conversion, random generation, emailing sync request generation.
 Kafka, ZMQ, AWS 53, Elastic integration, PGP encryption, CIC operations.

DISTRIBUTION

- algorithm for checking if a user has access to an attribute
- creator vs owner organisation
- distribution levels and inheritance (events -> objects -> attributes)
- shorthand inherit level
- sharing groups (org list, instance list)
- correlation distribution
- algorithms for safe data fetching (fetchEvents(), fetchAttributes(),...)

MISP core development crash course

2022-09

-Distribution

 algorithm for checking if a user has access to an attribute creator vs owner organisation

shorthand inherit level

m sharing groups (org list, instance list) correlation distribution

algorithms for safe data fetching (fetchEvents(),

TESTING YOUR CODE

- funtional testing
- Github actions
- impact scope
 - view code changes: only impacts request type based views
 - controller code changes: Should only affect given action
 - ▶ model code changes: can have impact on entire application
 - ▶ lib changes: can have affect on the entire application
- Don't forget: queryACL, change querystring

MISP core development crash course

2022-

-Testing your code

- # funtional testing ■ Github actions m impact scope

- Don't forget: queryACL, change querystring