

Ninja GUI Project



Ninja GUI Project



Per Åsberg
per@op5.com

System Developer
Master of Science In Informatics

@ op5 since may 2007:
Senior Developer / Project Manager



Agenda



- Background
- Dependencies
- MVC
- API
- Authentication
- Modifications
- Add-on Integration
- Differences to CGI's
- DEMO
- Future

Background



- Support: 70 % related to GUI
- New GUI been discussed a long time
- Lack of good search and filtering
- Needs a facelift
- Enable development of new ideas
- Release the potential of Nagios.

Mission



- New PHP versions of the cgi's
 - Put effort on “obvious problems”
 - Good search
 - Filtering, filter out stuff based on anything
 - Scalability. Ways to present large data sets, host lists, service lists, etc
- Multi language support (gettext)
- Template/skin support
- Multi DBMS support
- Object API
- Authentication, support common authentication protocols

Goal



- To create the leading OpenSource web frontend for nagios

Nagios Is Now Just (even more) Awesome!

How?



- Data storage
- PHP Framework
- Why?
 - ✓ Structure
 - ✓ Security
 - ✓ Simplicity
 - ✓ Functionality
 - ✓ Maintainability

How?



- Which one to choose?
- Why Kohana?
 - Secure
 - Lightweight
 - Easy to use
 - Flexible
 - Extensible

ZEND framework, phpDrone, CakePHP, Symfony, Achievo, Codeigniter, Ambivalence, Biscuit, Caffeine, Seagull, Yll, Struts4PHP, Agavi, SolarPHP, PHP On Trax, Qcodo, ZOOB, Tigermouse, Kohana, Spring, phpPeanuts, PHPulse, Nexista, InterJinn, Jelix, Innomatic, Kolibri, Kumbia, Inek, Fusebox, Copix, bitweaver, Akelos, Halo, Aukyla...etc...

Kohana



Benefits:

- ✓ Multi language support (gettext)
- ✓ Template/skin support
- ✓ Object API
- ✓ Authentication, support common authentication protocols
- ✓ Multi DBMS support

Dependencies

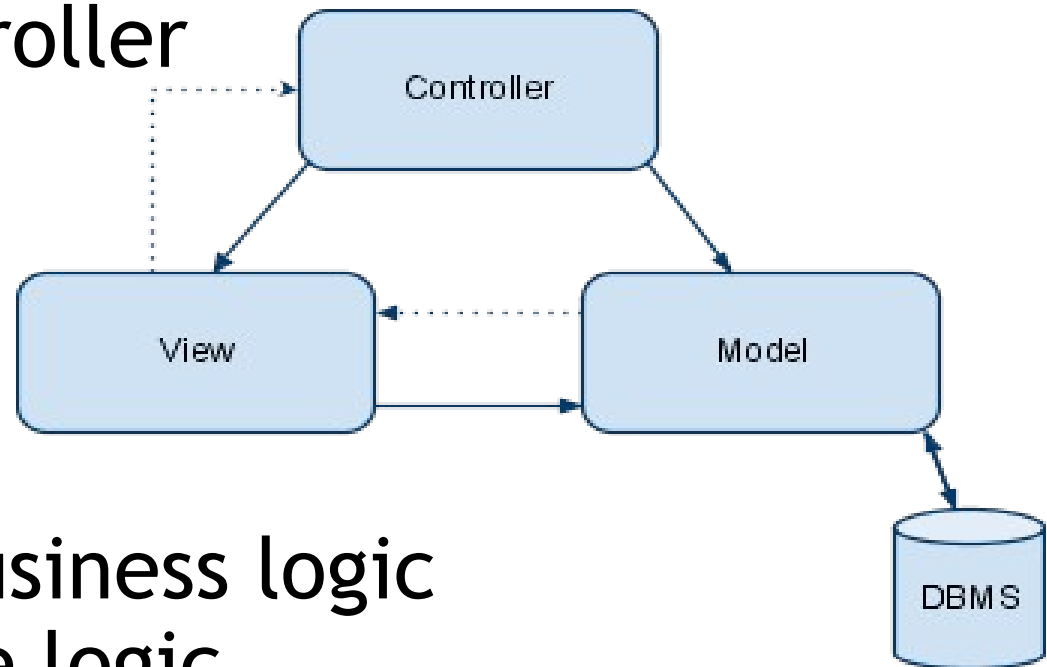


- Merlin
- PHP \geq 5.2.3
- Webserver:
 - ✓ Apache 1.3+
 - ✓ Apache 2.0+
 - ✓ Lighttpd
 - ✓ Nginx
 - ✓ MS IIS
- DBMS: (drivers)
 - ✓ MSSQL
 - ✓ MySQL
 - ✓ MySQLi
 - ✓ PostgreSQL
 - ✓ PDOsqlite

MVC

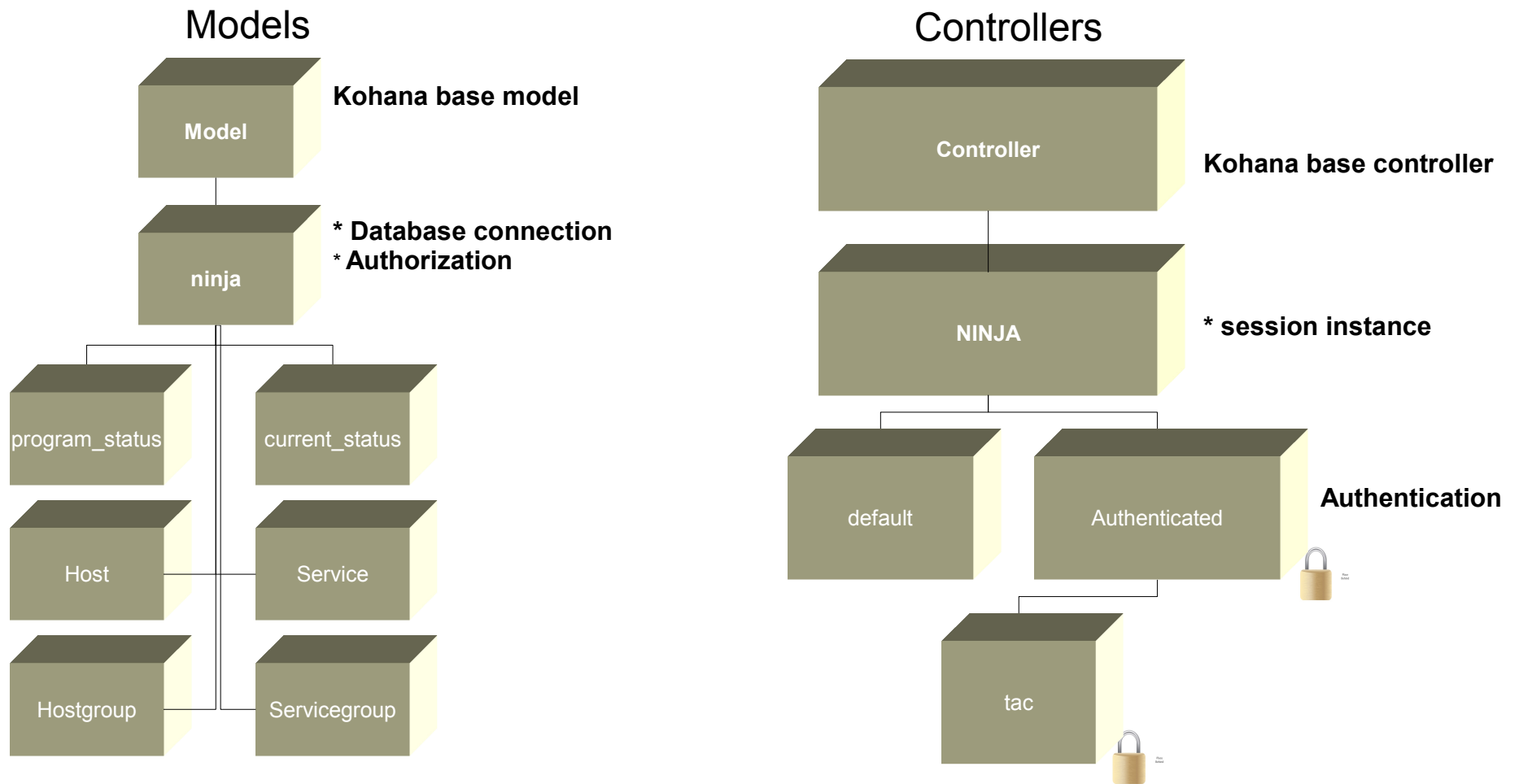


Model, View, Controller



Model = Data + business logic
Controllers = Page logic
Views = Page elements

Ninja MVC



API



API = Models

Ex:

```
Host_Model::fetch_all();  
Servicegroup_Model::fetch_all();  
Outages_Model::fetch_outage_data();
```

Authorization checks in Models

Authentication



Driver based authentication

Ex:

- Database (with .htaccess import)

- File based

- LDAP?

Uses `cgi.cfg`

Possible Modifications



- Multi language support
- Gettext (.po)
- Skins/Themes
- Widgets

Several included, possible to have customized
How to create?

- Self contained folder
- controller + view
- uses widget helper



Add-on integration



- NagVis
- NaCoMa (Nagios Configuration Manager)
- PNP4Nagios
- “OP5 reports”

Differences to CGI's



- Widgets
- Pagination
- Search/filter
- New info in Tactical Overview:
 - Unhandled problems
 - Scheduled downtime
 - Acknowledged problems
 - Disabled checks

DEMO



Future



- Authentication
 - LDAP
- Skins/Themes
- Translations
- Widgets
 - Upload
 - Share
- Oracle db driver
- SOAP/REST API

More info



<http://www.op5.org/community/projects/ninja>
per@op5.com

<http://ninja4nagios.blogspot.com>

Twitter: @ninja4nagios

Documentation: ???

Controller example



```
class Tac_Controller extends Authenticated_Controller
{
    public $model = false;

    public function index()
    {
        // Add the template (view)
        $this->template->content = $this->add_view('tac/index');
        // the add_view() method makes the location of the current
        // theme transparent and makes it possible to globally switch themes

        // create an instance of our current status model
        $this->model = new Current_status_Model();

        // fetch current status
        $this->model->analyze_status_data();
        // do something with the data and assign to template variables
    }
}
```

Cascading Resources



KOHANA MODULES

Infographic by Geert De Deckere

