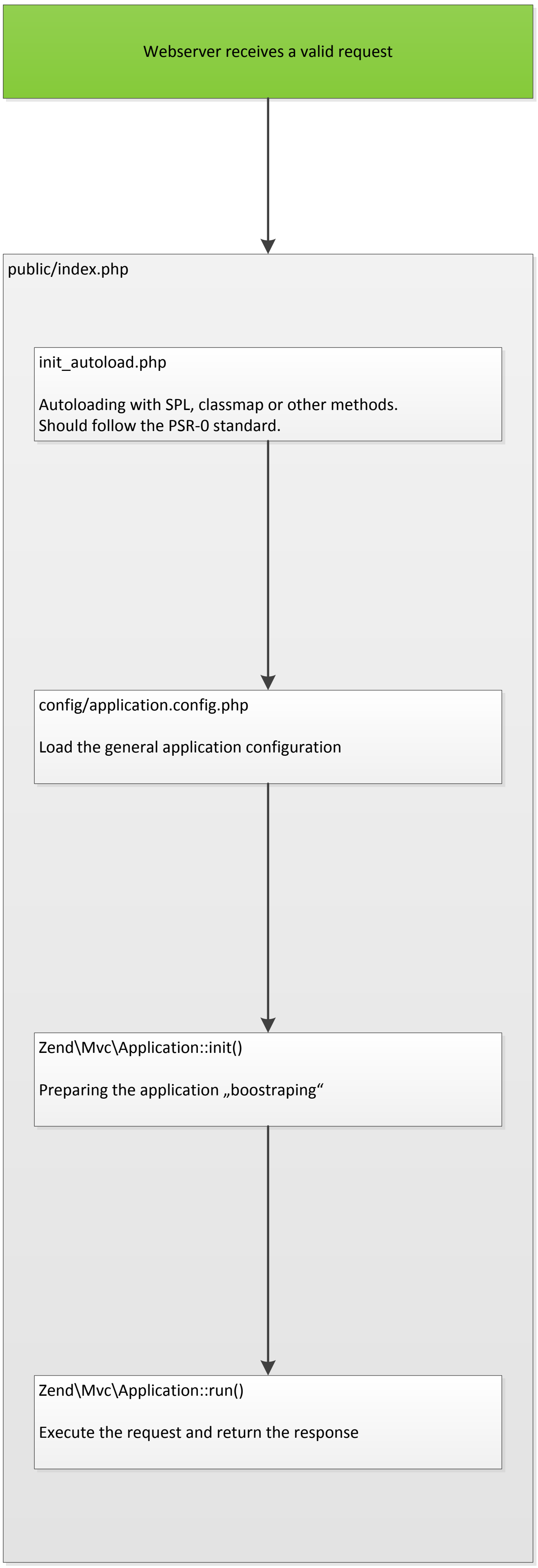


Webserver receives a valid request

public/index.php

init_autoload.php

Autoloading with SPL, classmap or other methods.
Should follow the PSR-0 standard.

config/application.config.php

Load the general application configuration

Zend\Mvc\Application::init()

Preparing the application „boostraping“

Zend\Mvc\Application::run()

Execute the request and return the response

What happens here?
The webserver receives a request from someone/somewhere.
The request will then get forwarded to the ZF2 application, if it matches the defineded configuration. This will execute the file „public/index.php“
With apache this is done in the <vhosts> section and/or with the .htaccess inside the public directory.

Performance tuning
Try to avoid a .htaccess file and define everything in the <vhosts> section and also define there „AllowOverride none“.
So Apache don‘t search for a .htaccess or .htpasswd file!

What happens here?
The index.php is the entry point for every ZF2 request.
Autoloading, configuration and execute the MVC is called here.

Performance tuning
Use generated classmaps for best autoloading speed.

Security
The index.php is in the public directory!
So define here nothing that is a potential security risk, like usernames, passwords and paths.
Generally keep the index.php as small as possible.

What happens here?
The configuration gets loaded for Zend\Mvc\Application.
There are definitions like: available modules, paths, custom services...

Performance tuning
Cache and load the file from memory (micro optimization)

Good to know
The modules are loaded in the order like you define it in the configuration! So if the same configuration is defined in two or more modules, the last one wins!

What happens here?
Basic services and objects are getting registered like:
- Request, Response, Route, ViewManager
- ApplicationConfig, ModuleManager, EventManager, …

Also all defined modules are getting „bootstraped“ over the <module_name>/Module.php, there the following methods are searched and executed if found:
- getAutoloaderConfig(), init(), getConfig(), onBootstrap()

Performance tuning
Keep the called methods in Module.php as small as possible! They will get called for EVERY request!
Cache the whole Zend\Mvc\Application object in memory after init() was called. Only the Request object has to changed for each request?!?!?!?!?

What happens here?
Now the request related action will get called (if found in the route).
Or a error action/page is shown: 404 NotFound, 500 internal error

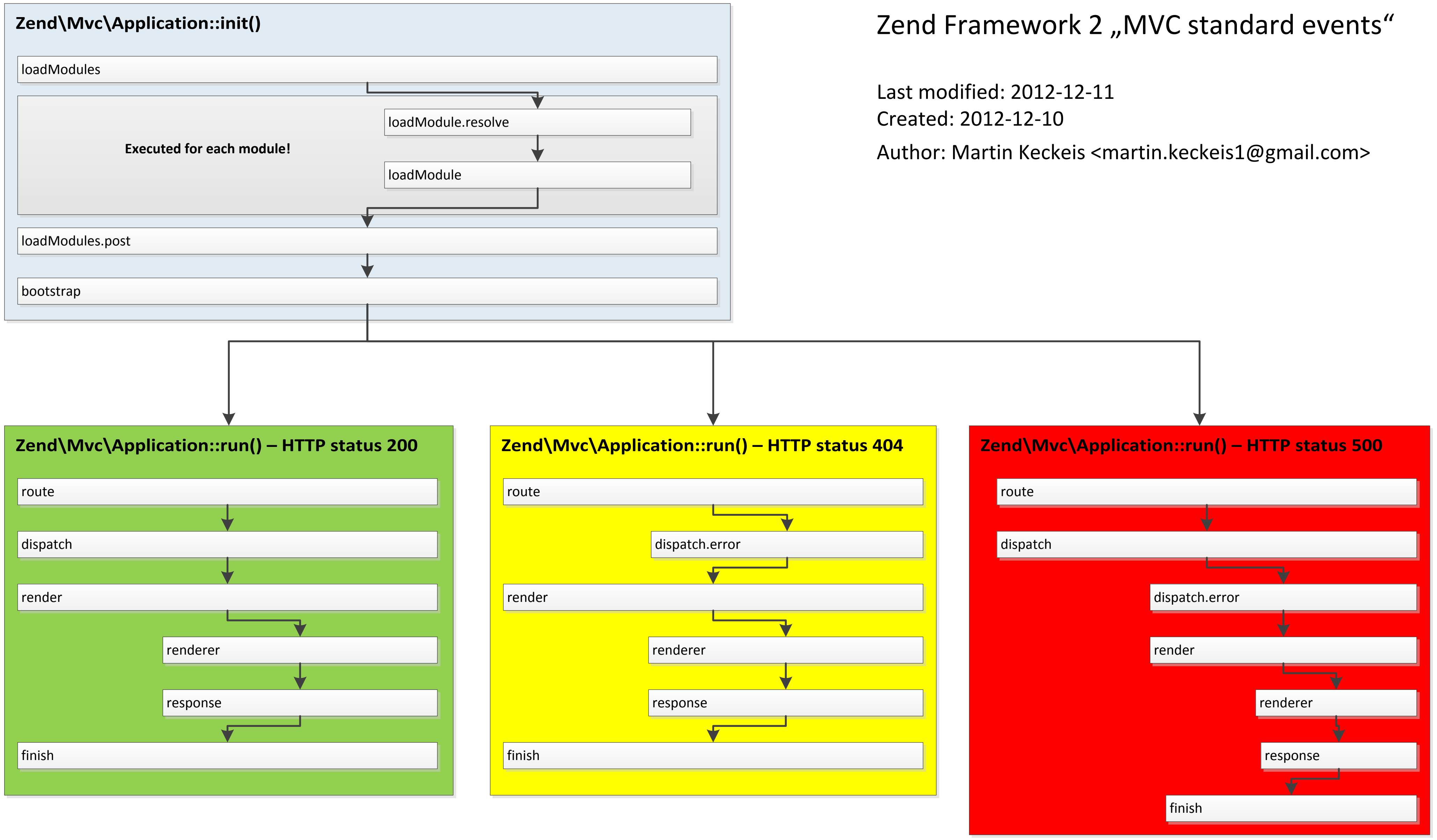
Performance tuning
Don‘t write to much code? ☺

Good to know
…..

Last modified: 2012-12-11
Created: 2012-12-10
Author: Martin Keckeis <martin.keckeis1@gmail.com>

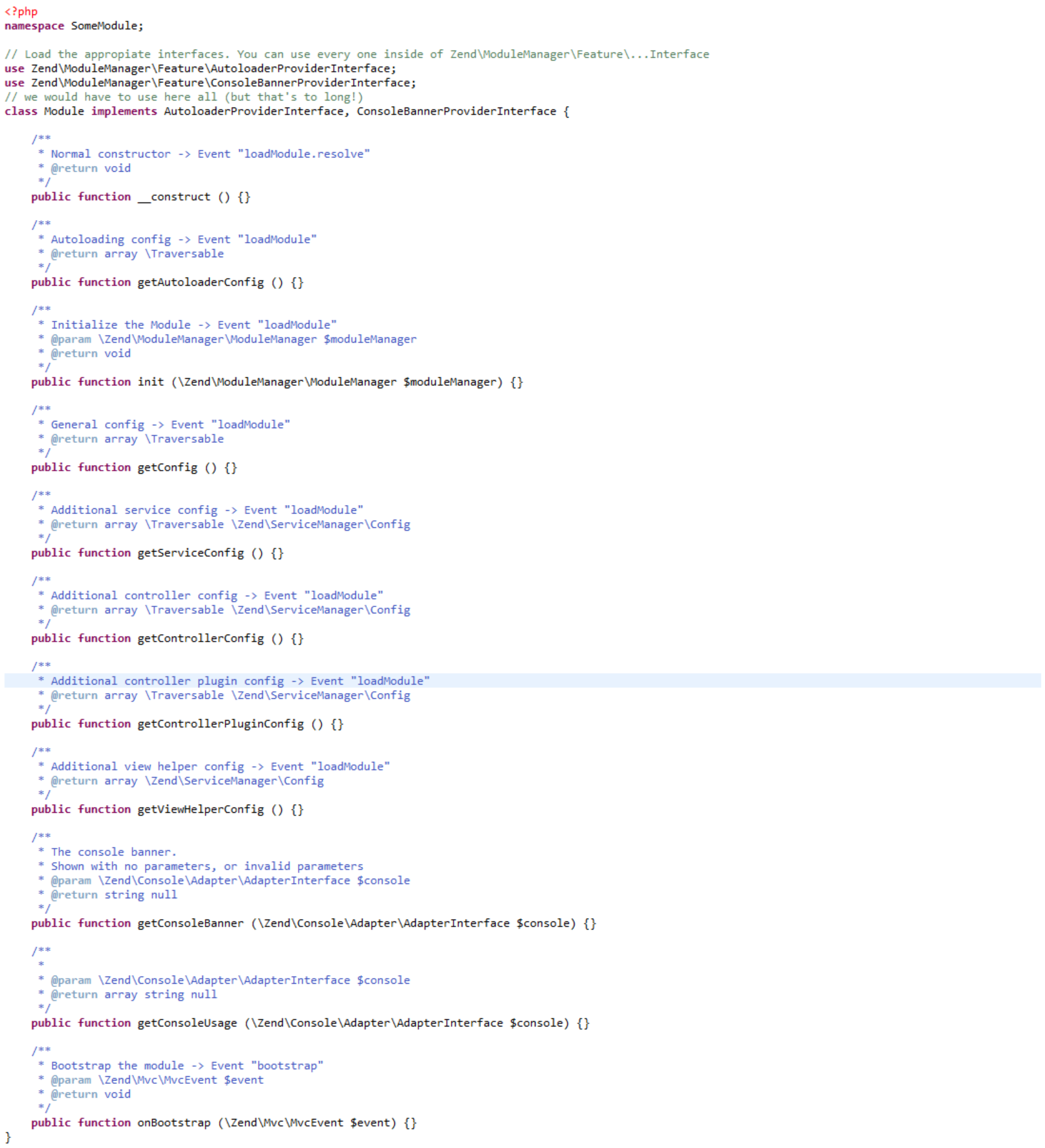
Zend Framework 2 „MVC standard events“

Last modified: 2012-12-11
Created: 2012-12-10
Author: Martin Keckeis <martin.keckeis1@gmail.com>



Module.php methods,

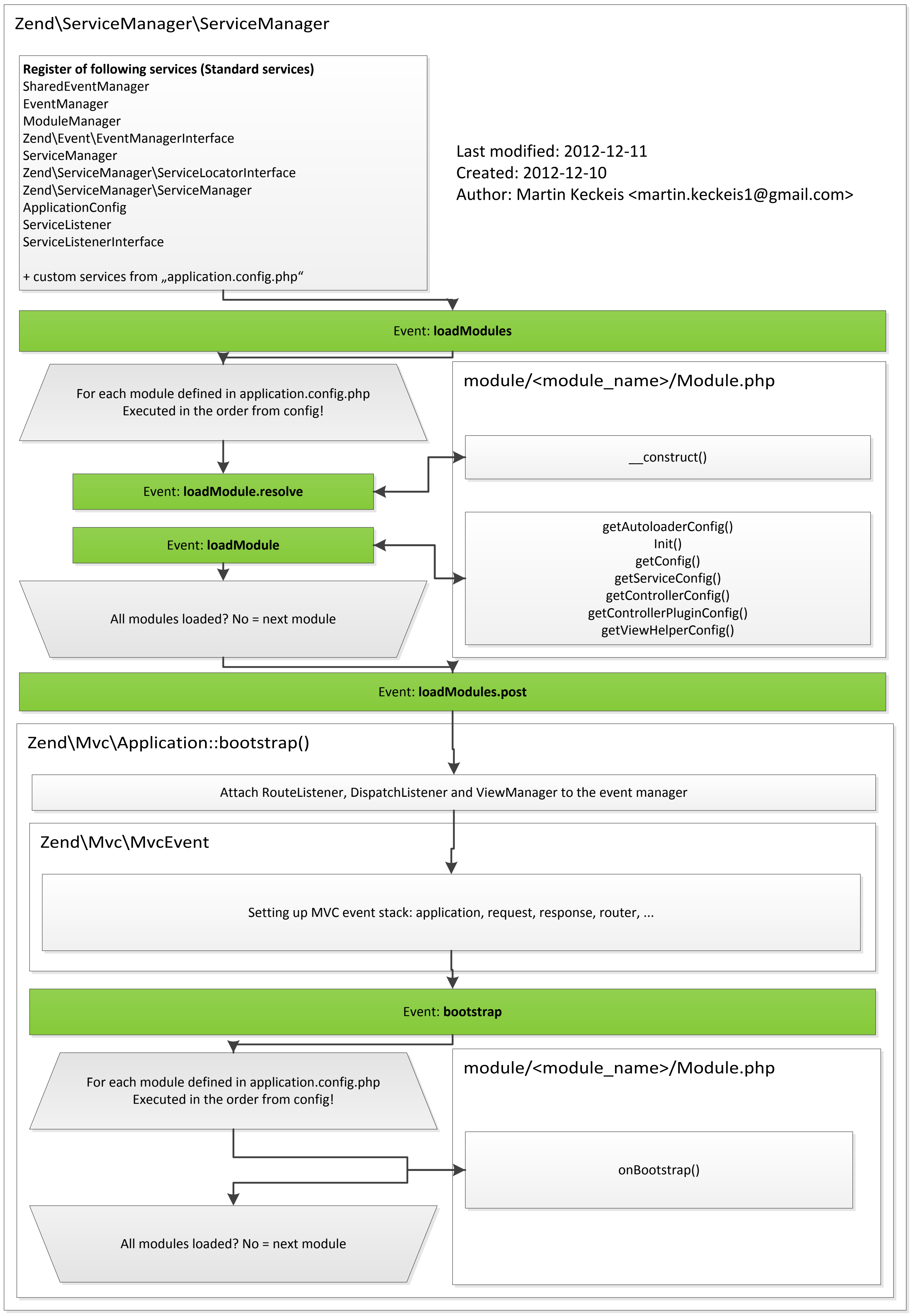
which get called….if they exist! The methods are declared in the order, they will get executed
The methods are called on three different events (see below): „loadModule.resolve“, „loadModule“ and „bootstrap“



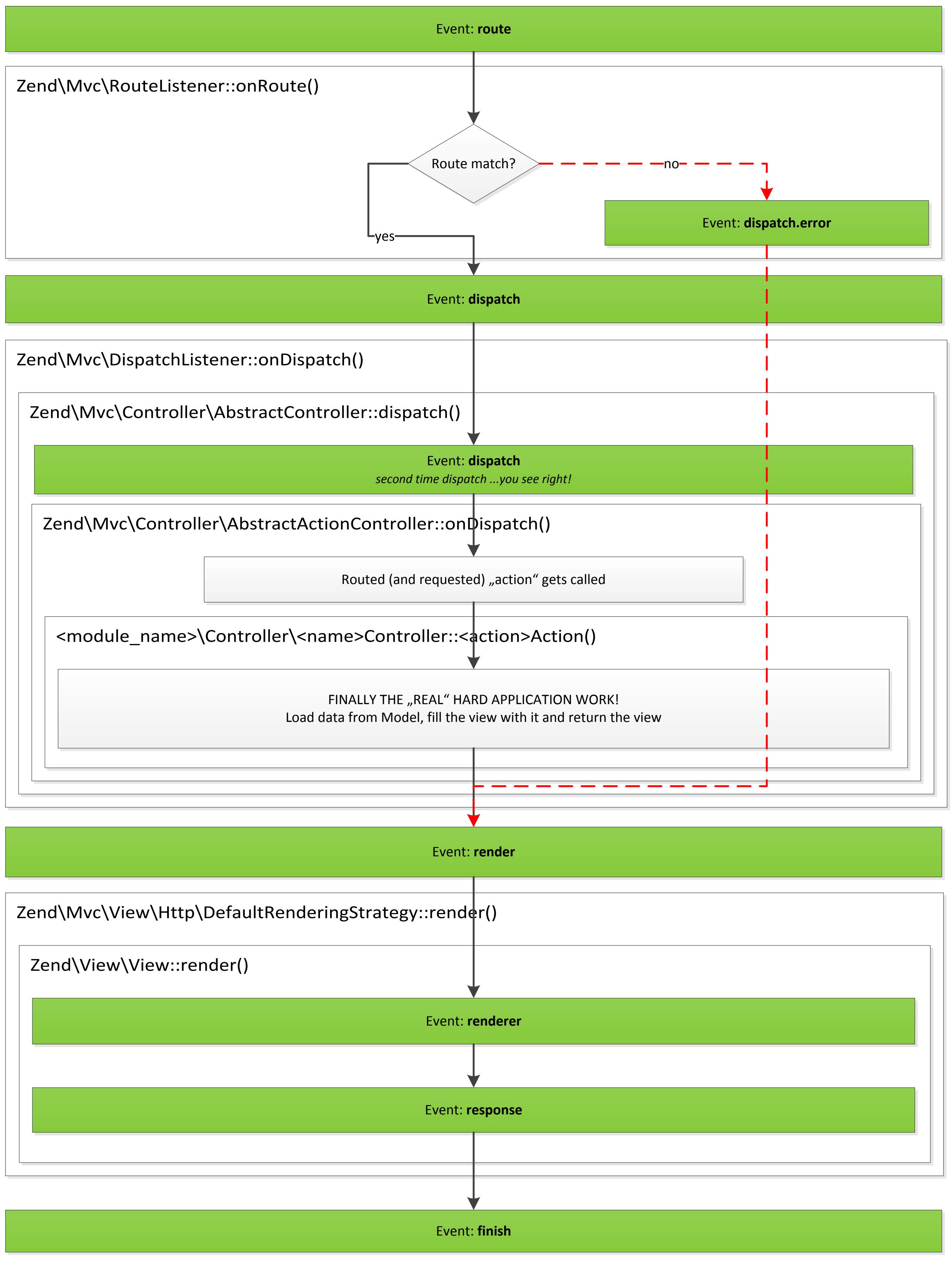
```php
<?php
namespace SomeModule;

// Load the appropiate interfaces. You can use every one inside of Zend\ModuleManager\Feature\...Interface
use Zend\ModuleManager\Feature\AutoloaderProviderInterface;
use Zend\ModuleManager\Feature\ConsoleBannerProviderInterface;
// we would have to use here all (but that's to long!)
class Module implements AutoloaderProviderInterface, ConsoleBannerProviderInterface {

    /**
     * Normal constructor -> Event "loadModule.resolve"
     * @return void
     */
    public function __construct () {}

    /**
     * Autoloading config -> Event "loadModule"
     * @return array \Traversable
     */
    public function getAutoloaderConfig () {}

    /**
     * Initialize the Module -> Event "loadModule"
     * @param \Zend\ModuleManager\ModuleManager $moduleManager
     * @return void
     */
    public function init (\Zend\ModuleManager\ModuleManager $moduleManager) {}

    /**
     * General config -> Event "loadModule"
     * @return array \Traversable
     */
    public function getConfig () {}

    /**
     * Additional service config -> Event "loadModule"
     * @return array \Traversable \Zend\ServiceManager\Config
     */
    public function getServiceConfig () {}

    /**
     * Additional controller config -> Event "loadModule"
     * @return array \Traversable \Zend\ServiceManager\Config
     */
    public function getControllerConfig () {}

    /**
     * Additional controller plugin config -> Event "loadModule"
     * @return array \Traversable \Zend\ServiceManager\Config
     */
    public function getControllerPluginConfig () {}

    /**
     * Additional view helper config -> Event "loadModule"
     * @return array \Zend\ServiceManager\Config
     */
    public function getViewHelperConfig () {}

    /**
     * The console banner.
     * Shown with no parameters, or invalid parameters
     * @param \Zend\Console\Adapter\AdapterInterface $console
     * @return string null
     */
    public function getConsoleBanner (\Zend\Console\Adapter\AdapterInterface $console) {}

    /**
     *
     * @param \Zend\Console\Adapter\AdapterInterface $console
     * @return array string null
     */
    public function getConsoleUsage (\Zend\Console\Adapter\AdapterInterface $console) {}

    /**
     * Bootstrap the module -> Event "bootstrap"
     * @param \Zend\Mvc\MvcEvent $event
     * @return void
     */
    public function onBootstrap (\Zend\Mvc\MvcEvent $event) {}
}
```

Last modified: 2012-12-11
Created: 2012-12-10
Author: Martin Keckeis <martin.keckeis1@gmail.com>

Zend\Mvc\Application::init() – Initialize the application – this will get executed for EVERY request!


Zend\ServiceManager\ServiceManager
Register of following services (Standard services)
SharedEventManager
EventManager
ModuleManager
Zend\Event\EventManagerInterface
ServiceManager
Zend\ServiceManager\ServiceLocatorInterface
Zend\ServiceManager\ServiceManager
ApplicationConfig
ServiceListener
ServiceListenerInterface
+ custom services from „application.config.php“
Last modified: 2012-12-11
Created: 2012-12-10
Author: Martin Keckeis <martin.keckeis1@gmail.com>
Event: loadModules
For each module defined in application.config.php
Executed in the order from config!
module/<module_name>/Module.php
__construct()
Event: loadModule.resolve
Event: loadModule
getAutoloaderConfig()
Init()
getConfig()
getServiceConfig()
getControllerConfig()
getControllerPluginConfig()
getViewHelperConfig()
All modules loaded? No = next module
Event: loadModules.post
Zend\Mvc\Application::bootstrap()
Attach RouteListener, DispatchListener and ViewManager to the event manager
Zend\Mvc\MvcEvent
Setting up MVC event stack: application, request, response, router, ...
Event: bootstrap
For each module defined in application.config.php
Executed in the order from config!
module/<module_name>/Module.php
onBootstrap()
All modules loaded? No = next module

Zend\Mvc\Application::run() – Execute the requested action – run the application itself


Event: **route**
Zend\Mvc\RouteListener::onRoute()
Route match?
no
yes
Event: **dispatch.error**
Event: **dispatch**
Zend\Mvc\DispatchListener::onDispatch()
Zend\Mvc\Controller\AbstractController::dispatch()
Event: **dispatch**
second time dispatch ...you see right!
Zend\Mvc\Controller\AbstractActionController::onDispatch()
Routed (and requested) „action“ gets called
<module_name>\Controller\<name>Controller::<action>Action()
FINALLY THE „REAL“ HARD APPLICATION WORK!
Load data from Model, fill the view with it and return the view
Event: **render**
Zend\Mvc\View\Http\DefaultRenderingStrategy::render()
Zend\View\View::render()
Event: **renderer**
Event: **response**
Event: **finish**

Last modified: 2012-12-11
Created: 2012-12-10
Author: Martin Keckeis <martin.keckeis1@gmail.com>