KinqModules – Moduly pro Nette

Programátorská příručka

# Spouštění

Spuštění modulů provedete přidáním konfiguračního souboru a rozšíření do souboru bootstrap.php před vytvoření Systémového kontejneru.

Příklad:

$configurator->addConfig(\_\_DIR\_\_ . '/config/kinqModules.neon');

$configurator->onCompile[] = function ($configurator, $compiler) {

$compiler->addExtension('kinqModulesExtension', new KinqModulesExtension);

};

$container = $configurator->createContainer();

# Souborová struktura

Struktura souborů v aplikační složce je následující:

+ app

+ Modules

+ Prvni

+ components

- NejakaKomponenta.php

+ config

- config.neon

+ models

- PrvniModel.php

+ presenters

- PrvniPresenter.php

- DruhyPresenter.php

+ static

+ css

+ js

+ templates

- Prvni.default.latte

PrvniModule.php

+ První.Vnoreny

+ components

- NejakaKomponenta.php

+ config

- config.neon

+ models

- VnorenyModel.php

+ presenters

- VnorenyPresenter.php

- DruhyVnorenyPresenter.php

+ static

+ css

+ js

+ templates

- Vnoreny.default.latte

VnorenyModule.php

Povinné je pojmenování souborů modulu „%Nazev\_modulu%Module.php“, dále složka modulu „%Nazev\_modulu%“. Submodul se ve složce píše s tečkou za jeho rodičem „%Nazev\_modulu%.%Nazev\_sub\_modulu%“. Dále je možné přidávat config soubor ke každému modulu s názvem config.neon. Dále pak template je nazýván „%Nazev\_presenteru%.%Nazev\_view%.latte“

# Jmenné prostory

Je třeba dodržovat základní jmenné prostory. Každý modul je ve jmenném prostoru „App\Modules“ a je nazýván „%Nazev\_modulu%Module“ Pokud se jedná o sub module je jmenný prostor „App\Modules\%Nazev\_modulu\_rodice%“.

Presentery jsou umístěny v „App\Modules\%Nazev\_modulu%\%Nazev\_presenteru%Presenter„. Pokud se jedná o sub modul je prostor „App\Modules\%Nazev\_modulu\_rodice%\%Nazev\_modulu%\%Nazev\_presenteru%Presenter„. Každý presenter by měl dědit „\Katrine\UI\Presenter“ aby obsahoval potřebné služby. Hlavní moduly dědí „\Katrine\KinqModules\BaseModule“. Pokud se jedná o submodul, tak dědí svůj rodičovský modul.

# Konfigurace

Pro spuštění modulů je třeba v konfiguraci uvést, které moduly se mohou spustit. To se uvádí do config.neon, či v našem prvním ukázkovém loadovacím případě kinqModules.neon, ve kterém je následující konfigurace. V poli kinqModules jsou uvedeny moduly, které se spustí.

common:

kinqModulesExtension:

kinqModules: ['Admin', 'Front.LastMinute', 'Front']

services:

factories:

production < common:

development < common:

Další konfigurace je možná v rámci každého modulu či submodulu ve složce config. V rámci tohoto konfiguračního souboru se nastavují konfigurace pro daný modul např. popis událostí, jak bude popsáno v další kapitole. Nastavení by měly být v poli s názvem „kinqModules%Nazev\_modulu%“.

# Události

V každém modulu je možné přetížit statickou metodu setupHooks. Tato metoda získá jako parametr HookContainer, do kterého je možné vázat události metodou bind. Té je zadán parametr s typem události a callback funkcí, která se má v případě potřeby spustit. V prezenteru je k dispozici služba modulesHook se stejným kontejnerem, na který je možné zavolat metodu notify, která zavolá všechny metody zadaného typu události se zadanými parametry.

public static function setupHooks(IHookContainer $hook) {

$hook->bind('menu', array(\_\_CLASS\_\_, 'bindMenu'));

}

public static function bindMenu($nav, $presenter) {

$nav->add('home', "Domů", $presenter->link("Front:default"));

}

Příklad notify v presenteru:

$this->modulesHook->notify('menu', array($this->nav, $this));

Aby bylo možné události využívat, je třeba přidat výčet typů událostí ve statické proměnné modulu self::$events, a to takové typy událostí, které jsou příhodné danému modulu.

public static $events = array('menu', ‘dalsiUdalost‘);

V konfiguračním souboru modulu dané události je možno nastavit (eventTypes) pro každou událost, jaké má obsahovat parametry (typy) a to je pak kontrolováno při spuštění scriptu. Příklad config.neon

common:

kinqModulesFront:

eventTypes:

menu: [App\Components\Navigation, Katrine\UI\Presenter]

# Routování

Každý modul může přetížit metodu setupRouter, které je předán parametr Router. Do něj je možné přidat potřebné RouteList s názvem daného modulu. Ty jsou pak obsluhovány, při routování, přičemž přednost mají moduly uvedené první v konfiguraci v kinqModules.

public static function setupRouter(IRouter $router) {

$router[] = $moduleRouter = new RouteList('Front.LastMinute');

$moduleRouter[] = self::createRouter('last-minute[/<action>]', array(

'presenter' => 'LastMinute',

'action' => 'default',

));

}

K vytvoření Router je možné využít statickou metodu rodiče self::createRouter.