Yii raamistiku standardiseeritud lahendused

Koostaja: Erik Uus

Ver	Kuupäev	Muutja	Kirjeldus
1.0	17.12.2009	Erik Uus	**Esimene versioon**
1.1	08.03.2010	Erik Uus	**Lisatud**: trükivaate rakendamine, portletite rakendamine erinevates küljendustes. **Eemaldatud**: vormide standardlahendused.
2.0	01.07.2010	Erik Uus	*Kogu dokumendi ulatuses on tehtud fundamentaalseid muudatusi ja olulisi täiendusi.* **Eemaldatud**: kujunduse standardlahendused (seda osa peaks edasipidi katma näidiskeskkond). **Lisatud**: IP aadressi põhine juurdepääsukontroll, klassifikaatorite rakendamine, abitekstide kuvamine, Saaga veebiteenused.
3.0	04.11.2010	Erik Uus	*Muudetud on dokumendi skoopi. Käesolev dokument on nüüd eelkõige standardiseeritud lahenduste ja laienduste installeerimise ja kasutamise juhend; see ei sisalda enam üldisi õpetusi Yii raamistiku kohta.* **Eemaldatud**: Autentimine ja autoriseerimine (Andmebaasipõhine autentimine, Paroolide krüptimine, Lihtne autoriseerimine kasutajanime põhjal, RBAC - rollipõhine autoriseerimine), Klassifikaatorite rakendamine, Abitekstide kuvamine. **Lisatud**: Standardiseeritud moodulid (Kasutajate moodul, Klassifikaatorite moodul, Abitekstide moodul).
3.1	20.12.2010	Erik Uus	**Muudetud**:kasutajate mooduli installeerimise õpetus, keelevaliku rakendamine (mitmekeelsus) **Lisatud**: trükivaate rakendamine, kontekstimenüü portlet, tõstutundetu otsing ja metasümolitega otsing
3.2	15.08.2011	Erik Uus	**Täiendatud**: koodi genereerimne (tõlkefailide genereerimine), abitekstide moodul (*OnSite* redigeerimise võimalus), trükivaade (lingi lisamine). **Muudetud**: Veebiteenuste aadress **Lisatud**: Haldusüksuste moodul
3.3	16.08.2012	Erik Uus	**Eemaldatud**: Veebiteenused (dokumentatsioon nende kohta asub nüüd spetsiaalsel veebilehel)

Sissejuhatus

Käesolev dokument on Yii raamistiku (http://www.yiiframework.com/) *standardiseeritud lahenduste ja laienduste installeerimise ja kasutamise juhend.*

Käesolev dokument

- eeldab, et lugejal on olemas põhjalikud teadmised Yii raamistiku kohta;
- ei dubleeri Yii raamistiku dokumentatsiooni (http://www.yiiframework.com/doc/).

Yii kaust arendusserveris

Arendusserveris, veebi juurkaustas, asub yii kaust, mis sisaldab järgmisi alamkaustu:

Kaust	**Kommentaar**
docs	Dokumentatsiooni kaust.
extensions	Sisaldab Yii raamistiku laiendusi. Dubleerimise vältimiseks viitavad kõigi Yii rakenduste extension kaustad *symlinkidena* sellele kaustale.
framework	Sisaldab Yii raamistikku. Selles kaustas asuva yii.php faili kaudu, millele viitavad rakenduste index.php failid, ühendatakse rakendused raamistikuga.
repository	Sisaldab standardiseeritud skripte, mida saab vastavalt vajadusele rakendustesse kopeerida.
services	Keskne SOAP veebiteenuseid pakkuv Yii rakendus.
xwebapp	Sisaldab Yii rakenduse laiendatud skeletti.

Uue rakenduse loomine

Skeleti genereerimine käsurealt

1. Ava käsurealt raamistiku kaust:

```
cd /var/www/html/yii/framework
```

2. Jooksuta (määrates rakenduse sihtkausta vastavalt vajadusele):

```
./yiic xwebapp /var/www/html/rakendus
```

3. Kirjuta Y ja vajuta ENTER:

```
Create a Web application under '/var/www/html/rakendus'? [Yes|No]
```

4. Ilmub teade:

```
Your application has been created successfully under
/var/www/html/xxx/rakendus.
```

5. Ava veebilehitsejaga aadress: http://xxx/rakendus

VAU-ga integreerimine

1. Ava fail **rakendus/protected/config/params.php**

2. Kui soovid integreerida rakenduse VAU lingimärkmikuga, muuda:

```
'vauBookmark'=>true,
```

3. Kui soovid integreerida rakenduse VAU tagasiside haldmise süsteemiga, muuda:

```
'vauHelpdesk'=>true,
```

Google Analytics integreerimine

1. Ava fail **rakendus/protected/config/params.php** ja muuda**:**

```
'googleAnalytics'=>true,
```

2. Muuda *trackeri* koodi vastavalt vajadusele, näiteks:

```
'googleAnalyticsTracker'=>'UA-4477704-2',
```

Koodi genereerimine

1. Ava konfiguratsioonifail: **`rakendus/protected/config/main.php`**

Kommenteeri sisse andmebaasi komponent ja lisa vajalikud andmed ühenduse loomiseks (asenda küsimärgid):

```
'db'=>array(
	'class'=>'CDbConnection',
	'connectionString'=>'pgsql:host=localhost;port=5432;dbname=?',
	'username'=>'?',
	'password'=>'?',
	'schemaCachingDuration'=>3600,
	'enableProfiling'=>false,
),
```

2. Ava veebilehitsejaga Gii moodul: `http://xxx/rakendus/index.php/gii` ja logi sisse (parool on määratud rakenduse konfiguratsioonifailis).

3. Kasutades erinevaid templaate, genereeri vajalikud *model*, *crud* ja *form* skriptid.

Templaatide seletus:

- **Model Generator**
  - default
    - tüüpiline mudel, kuhu on lisatud ka *search* meetod, mille abil saab luua *gridview* filtreid
    - sobib kasutamiseks näiteks nende tabelite juures, mis peavad olema haldusliideses eraldi hallatavad
  - compact
    - minimaalne mudel, kus võrreldes *default* mudeliga puudub *search* meetod
    - sobib kasutamiseks näiteks nende tabelite juures, mille andmed ei pea olema haldusliideses eraldi hallatavad
  - search
    - mudel, kuhu on lisatud spetsiaalne *search* meetod keerukate otsingute jaoks
    - sobib kasutamiseks otsisüsteemide põhitabelite juures
  - searchForm
    - mudel, millel põhinevad otsivormid (mis valideerib ja filtreerib otsivorme)
    - sobib kasutamiseks otsisüsteemide otsinguvormide juures
  - tree
    - mudel, mis on loodud spetsiaalselt puustruktuuri jaoks
    - saab kasutada nende tabelitega, kus on väljad *id* ja *parent_id*

- **Form Generator**

    - default
        - gerereerib valitud tabeli väljadele vastava html vormi ja sellega seotud *aktsiooni*, mida saab kohandada vastavalt vajadusele

    - search
        - genereerib valitud tabeli väljadele vastava otsivormi ja sellega seotud *aktsiooni,* mida saab kasutada kontrollerites otsingu meetodina
        - on mõeldud kasutamiseks koos koodiga, mida genereeritakse Model Generator'i *search* ja *searchForm* templaatide järgi

- **Crud Generator**

    - default
        - genereerib tüüpilise *crud* komplekti: avalik *listview* ja *detailview*, sisselogimist nõudev *filtreeritav ajax gridview* , lisamise ja muutmise *vorm*
        - eeldab mudelit, mis on genereeritud Model Generator'i *default* templaadi järgi

    - admin
        - genereerib ainult sisselogimist nõudva *filtreeritava ajax gridview*, lisamise ja muutmise *vormi*
        - eeldab mudelit, mis on genereeritud Model Generator'i *default* templaadi järgi
        - sobib kasutamiseks näiteks nende tabelite juures, mille andmed peavad olema haldusliideses eraldi hallatavad

    - search
        - genereerib otsingutulemuste kuvamiseks mõeldud *mitte-ajax gridview* (mis pakub sisseloginud kasutajale ka haldusfunktsioone), detailview, lisamise ja muutmise *vormi*
        - eeldab mudelit, mis on genereeritud Model Generator'i *search* templaadi järgi
        - sobib otsisüsteemide loomiseks

    - tree
        - genereerib puustruktuuri sirvimise ja haldamise mahuka komplekti, mis võimaldab puud kuvada *breadcrumbs* menüüga varustatud nimekirjana, *autocomplete* otsingu kaudu, *ajax treeview* abil või lihtsalt *multilevel* menüüna (viimast on mõistlik kasutada ainult väikesemahuliste puude puhul)
        - eeldab mudelit, mis on genereeritud Model Generator'i *tree* templaadi järgi

4. Pärast koodi genereerimist täienda tõlkefaili: **`rakendus/protected/messages/et/ui.php`**

Seda saab teha ka **käsitsi**. Näiteks, kui genereeriti CRUD failid mudelist *User*, siis tuleb lisada tõlkefaili:

```
'Users' => 'Kasutajad',
'Manage Users' => 'Halda kasutajaid',
'New User' => 'Uus kasutaja',
'Update User' => 'Muuda kasutaja andmeid',
'View User' => 'Vaata kasutaja andmeid',
```

Aga mugavam on kasutada **yiic message** käsku. XWebapp käsuga genereeritud rakendusel on kohe olemas ka spetsiaalne konfiguratsioonifail
rakendus/protected/config/messages.php,
mida saab kasutada koos käsuga
yiic message path/to/rakendus/protected/config/messages.php

Standardiseeritud moodulid

Kasutajate moodul

Sissejuhatus

Kasutajate moodul lisab rakendusele *kasutajate autentimise ja haldamise* funktsionaalsuse. Kasutajate moodul asub keskses laienduste kaustas **yii/extensions/modules/user**. Selleks, et ühendada kasutajate moodul rakendusega, ei ole vaja kopeerida ühtegi faili, vaid piisab ainult rakenduse konfiguratsioonifaili täiendamisest. Samuti ei ole vaja luua andmebaasi tabelit, sest seda saab teha moodulis endas *install* kontrolleri kaudu.

Installeerimine

Kasutajate mooduli installeerimise eelduseks on, et rakendusele on juba loodud ühendus andmebaasiga.

1. Minimaalne kood, mis tuleb kasutajate mooduli rakendamiseks lisada konfiguratsioonifaili **rakendus/protected/config/main.php** on järgmine:

```
// modules
 'modules'=>array(
       'user'=>array(
             'class'=>'ext.modules.user.UserModule',
       ),
 ),
```

2. Salvesta muudatused, logi sisse *admin* kasutajana (see sisselogimine kasutab veel *xwebapp* käsu poolt loodud lihtsat autentimise komponenti **rakendus/protected/components/UserIdentity.php**, kus kasuatajanimi ja salasõna on kirjutatud otse PHP faili) ja ava lehekülg http://xxx/rakendus/index.php/user/install

3. Järgi juhiseid ja loo mooduli jaoks vajalik andmebaasi tabel. Tabel luuakse automaatselt järgmise käsuga:

```
CREATE TABLE $userTable
(
  id serial NOT NULL,
  username character varying(64) NOT NULL,
```

```
    password character varying(64) NOT NULL,
    role character varying(16),
    salt character varying(64),
    firstname character varying(64),
    lastname character varying(64),
    usergroup character varying(64),
    CONSTRAINT pk_user PRIMARY KEY (id)
);
```

Ja koos tabeli loomisega lisatakse sinna ka kasutaja admin (salasõna samuti admin).

4. Logi välja.

5. Lisa konfiguratsioonifaili read, mis impordivad rakendusse kasutajate mooduli mudelid ja komponendid:

```
'import'=>array(
    'ext.modules.user.models.*',
    'ext.modules.user.components.*',
),
```

Ja kustuta rakenduses fail **`rakendus/protected/components/UserIdentity.php`**

6. Nüüd on moodul kasutamiseks valmis. Logi sisse kasutajana *admin/admin*.

Aadressil http://xxx/rakendus/index.php/user/account avaneb lehekülg "Minu konto", mis on ligipääsetav kõigile autenditud kasutajatele, kus nad saavad muuta oma andmeid ja salasõna (NB! Ära unusta seal kohe ära muuta ka *admin* kasutaja parool).

Aadressil http://xxx/rakendus/index.php/user avaneb kasutajate haldmise lehekülg, mis on ligipääsetav ainult *admin* kasutajale.

Konfigureerimine

Kasutajate mooduli juures saab konfigureerida veel järgmisi muutujaid:

```
public $userTable='user';
public $userLayout;
public $leftPortlets=array();
public $rightPortlets=array();
public $userGroups=array();
public $rbac=false;
```

Seletame seda järgmise konfiguratsiooni näitel:

```
// autoloading
'import'=>array(
    'ext.modules.user.models.*',
    'ext.modules.user.components.*',
),
// modules
'modules'=>array(
    'user'=>array(
```

```
            'class'=>'ext.modules.user.UserModule',
            'userTable'=>'tbl_user',
            'rbac'=>'manageUsers',
            'userLayout'=>'application.views.layouts.sidebars',
            'leftPortlets'=>array(
                'ptl.LeftMenuTop'=>array(),
                'ptl.LeftMenuBottom'=>array(),
            ),
            'rightPortlets'=>array(
                'ptl.RightMenuTop'=>array(),
                'ptl.RightMenuBottom'=>array(),
            ),
            'userGroups'=>array(
                'group_name_a',
                'group_name_b',
            ),
        ),
    ),
```

1. Kasutajate moodul loob vaikimisi tabeli nimega "user". Kui on vaja kasutada mõnda muud tabeli nime, saab seda määrata muutuja *userTable* kaudu:

```
'userTable'=>'tbl_user',
```

2. Kasutajate moodulit saab kasutada ka koos rollipõhise juurdepääsukontrolliga. Kui muutuja *rbac* on määramata (st on vaikimisi false), siis kontrollitakse juurdepääsu moodulile kasutajanime põhjal:

- kõigil autenditud kasutajatel on ligipääs leheküljele "Minu konto", kus nad saavad muuta oma andmeid ja salasõna
- ainult kasutajal *admin* on ligipääs kasutajate haldamise leheküljele, kus ta saab kasutajaid lisada, muuta ja kustutada

Aga oletame, et rakenduses on võetud kasutusele RBAC definitsiooniga:

```
$auth = Yii::app()->authManager;
$auth->createOperation('manageUsers', 'View, add and update users');
$role = $auth->createRole('userManager');
$role->addChild('manageUsers');
$role = $auth->createRole('siteManager');
$role->addChild('userManager');
$auth->save();
```

Selleks et ligipääs kasutajate moodulile oleks ainult neis rollides, millele on omistatud operatsioon *manageUsers* (antud juhul on need siis rollid *userManager* ja *siteManager*), tuleb mooduli konfiguratsioonis määrata:

```
'rbac'=>'manageUsers',
```

Nüüd on ligipääs moodulile reguleeritud järgmiselt:

- kõigil autenditud kasutajatel on ligipääs leheküljele "Minu konto", kus nad saavad muuta oma andmeid ja salasõna

- kõigil autenditud kasutajatel *userManager* või *siteManager* rollis on ligipääs kasutajate haldamise leheküljele, kus ta saavad kasutajaid (va. *admin* kasutaja) lisada ja muuta
- ainult kasutajal *admin* on võimalik lisaks ka kasutajaid kustutada

3. Rakenduse konfiguratsioonis saab määrata, millises küljenduses ja koos milliste portletitega mooduli lehekülgi selle rakenduse kontekstis kuvatakse. Näitena toodud konfiguratsioon

```
'userLayout'=>'application.views.layouts.sidebars',
'leftPortlets'=>array(
    'ptl.LeftMenuTop'=>array(),
    'ptl.LeftMenuBottom'=>array(),
),
'rightPortlets'=>array(
    'ptl.RightMenuTop'=>array(),
    'ptl.RightMenuBottom'=>array(),
),
```

määrab, et kasutajate mooduli lehekülgi kuvatakse antud rakenduses nii, et mõlemal pool on alati nähtavad kaks portletit.

4. Kui on vaja kasutajaid jagada gruppideks, saab grupinimed lisada konfiguratsiooni järgmiselt:

```
'userGroups'=>array(
    'group_name_a',
    'group_name_b',
),
```

Mooduli kasutajaliidesesse ilmub seepeale juurde grupinime jaoks vajalik funktsionaalsus.

Grupinimesid saab tõlkida rakenduse tõlkefailis: **`rakendus/protected/messages/et/ui.php`**

```
 'Group Name A'=>'Grupi A nimetus',
 'Group Name B'=>'Grupi B nimetus',
```

Autenditud kasutaja grupinime saab rakenduses kasutada järgmiselt:

```
if(Yii::app()->user->usergroup=='group_name_a') // do something
```

Klassifikaatorite moodul

Sissejuhatus

Klassifikaatorite moodul lisab rakendusele *klassifikaatorite kuvamise ja haldamise* funktsionaalsuse. Klassifikaatorite moodul asub keskses laienduste kaustas **`yii/extensions/modules/lookup`**. Selleks, et ühendada klassifikaatorite moodul rakendusega, ei ole vaja kopeerida ühtegi faili, vaid piisab ainult rakenduse konfiguratsioonifaili täiendamisest. Samuti ei ole vaja luua andmebaasi tabelit, sest seda saab teha moodulis endas *install* kontrolleri kaudu.

Installeerimine

Klassifikaatorite mooduli installeerimise eelduseks on, et rakendusele on juba loodud ühendus andmebaasiga.

1. Minimaalne kood, mis tuleb klassifikaatorite mooduli rakendamiseks lisada konfiguratsioonifaili **`rakendus/protected/config/main.php`** on järgmine:

```
'import'=>array(
    'ext.modules.lookup.models.*',
),

'modules'=>array(
    'user'=>array(
        'class'=>'ext.modules.lookup.LookupModule',
    ),
),
```

2. Salvesta muudatused, logi sisse *admin* kasutajana ja ava lehekülg http://xxx/rakendus/index.php/lookup/install

3. Järgi juhiseid ja loo mooduli jaoks vajalik andmebaasi tabel. Tabel luuakse automaatselt järgmise käsuga:

```
CREATE TABLE $lookupTable
(
  id serial NOT NULL,
  code integer,
  name_et character varying(256),
  name_en character varying(256),
  type character varying(64),
  position integer,
  CONSTRAINT pk_lookup PRIMARY KEY (id)
);
```

4. Moodul on nüüd kasutamiseks valmis.

Aadressil http://xxx/rakendus/index.php/lookup avaneb lehekülg "Klassifikaatorid", kus on kõigepealt menüü kõigist klassifikaatorite tüüpidest. Tüübinimel klõpsates avaneb lehekülg, kus saab hallata antud tüüpi klassifikaatoreid.

Juhul, kui Sa alles mooduli kaudu genereerisid uue tabeli, ei ole tabelis muidugi veel ühtegi klassifikaatorit, mistõttu on ka tüüpide avamenüü esialgu tühi.

Kuna klassifikaatorite tüübid fikseeritakse arendaja poolt, kes sisestab tabelisse ka esmased klassifikaatorite väärtused, siis võib seda teha muidugi ka otse andmebaasis. Aga kui oled rakenduses sisseloginud *admin* kasutajana, saad uut tüüpi klassifikaatori lisada ka nii, et kirjutad aadressireale http://xxx/rakendus/index.php/lookup/default/admin?type=eyecolor ja klõpsid avanenud leheküljel lingil *uus*, mis viib sisestusvormi juurde (*eyecolor* on siin muidugi lihtsalt üks tüübinime näide). Pärast esimese klassifikaatori lisamist ilmub selle tüübinimi juba menüüsse, nii et järgmisi sama tüüpi klassifikaatoreid saad lisada mooduli avamenüüst lähtudes.

Konfigureerimine

Klassifikaatorite mooduli juures saab konfigureerida veel järgmisi muutujaid:

```
public $lookupTable='lookup';
public $lookupLayout;
public $leftPortlets=array();
public $rightPortlets=array();
public $safeTypes=array();
public $rbac=false;
```

Seletame seda järgmise konfiguratsiooni näitel:

```
// autoloading
'import'=>array(
    'ext.modules.lookup.models.*',
),
// modules
'modules'=>array(
    'lookup'=>array(
        'class'=>'ext.modules.lookup.LookupModule',
        'lookupTable'=>'tbl_lookup',
        'rbac'=>'manageLookups',
        'lookupLayout'=>'application.views.layouts.leftbar',
        'leftPortlets'=>array(
            'ptl.LeftMenuTop'=>array(),
            'ptl.LeftMenuBottom'=>array(),
        ),
        'safeTypes'=>array(
            'eyecolor',
            'haircolor',
        ),
    ),
),
```

1. Klassifikaatorite moodul loob vaikimisi tabeli nimega "lookup". Kui on vaja kasutada mõnda muud tabeli nime, saab seda määrata muutuja *lookupTable* kaudu:

```
'lookupTable'=>'tbl_lookup',
```

2. Klassifikaatorite moodulit saab kasutada ka koos rollipõhise juurdepääsukontrolliga. Kui muutuja *rbac* on määramata (st on vaikimisi false), siis kontrollitakse juurdepääsu moodulile kasutajanime põhjal:

- kõigil autenditud kasutajatel on ligipääs mooduli lehekülgedele, kus nad saavad lisada, muuta ja järjestada klassifikaatoreid
- ainult kasutajal *admin* on lisaks võimalik ka klassifikaatoreid kustutada

Aga oletame, et rakenduses on võetud kasutusele RBAC definitsiooniga:

```
$auth = Yii::app()->authManager;
$auth->createOperation('manageLookups','View, add and update lookups');
$role = $auth->createRole('contentManager');
```

```
$role->addChild('manageLookups');
$role = $auth->createRole('siteManager');
$role->addChild('contentManager');
$auth->save();
```

Selleks, et ligipääs klassifikaatorite moodulile oleks ainult neis rollides, millele on omistatud operatsioon *manageLookups* (antud juhul on need siis rollid *contentManager* ja *siteManager*), tuleb mooduli konfiguratsioonis määrata:

```
 'rbac'=>'manageLookups',
```

Nüüd on ligipääs moodulile reguleeritud järgmiselt:

- ainult neil autenditud kasutajatel, kes on *contentManager* või *siteManager* rollis, on ligipääs mooduli lehekülgedele, kus nad saavad lisada, muuta ja järjestada klassifikaatoreid
- ainult kasutajal *admin* on võimalik lisaks ka klassifikaatoreid kustutada

3. Rakenduse konfiguratsioonis saab määrata, millises küljenduses ja koos milliste portletitega mooduli lehekülgi selle rakenduse kontekstis kuvatakse. Näitena toodud konfiguratsioon

```
'lookupLayout'=>'application.views.layouts.leftbar',
'leftPortlets'=>array(
    'ptl.LeftMenuTop'=>array(),
    'ptl.LeftMenuBottom'=>array(),
),
```

määrab, et klassifikaatorite mooduli lehekülgi kuvatakse antud rakenduses nii, et vasakul pool on alati nähtavad kaks portletit.

4. Arendamise seisukohalt on mugav hoida samas tabelis ka süsteemseid klassifikaatoreid, mida süsteemi kasutajad ei tohi muuta saada. Sellisel juhul saab mooduli konfiguratsioonis defineerida klassifikaatorite tüübid, mida saavad muuta kõik kasutajad, kellel on ligipääs klassifikaatorite moodulile

```
'safeTypes'=>array(
    'eyecolor',
    'haircolor',
),
```

Neile klassifikaatoritele, mille tüüp ei ole *safeTypes* hulgas, on ligipääs ainult *admin* kasutajal.

Kui *safeTypes* ei ole defineeritud, saavad kõik kasutajad, kellel on ligipääs moodulile, muuta kõiki klassifikaatoreid.

Klassifikaatorite kasutamine rakenduses

Klassifikaatorite moodul toob rakendusse meetodid *Lookup::item()* ja *Lookup::items()*, mis võimaldavad väga mugavalt klassifikaatoritega opereerida:

1. Klassifikaatori nimetuse kuvamine tüübi ja koodi järgi

```
echo Lookup::item("eyecolor",$model->eyecolor_code);
```

2. Rippmenüü kuvamine tüübi järgi

```
echo CHtml::activeDropDownList($model,'eyecolor_code',
Lookup::items('eyecolor'));
```

rippmenüü koos tühja valikuga ja valikuga "-lisa-"

```
echo CHtml::activeDropDownList($model,'eyecolor_code',
Lookup::add()+Lookup::items('eyecolor'),array('prompt'=>''));
```

3. Klassifikaatorite kasutamine *gridview* kontekstis

```
$this->widget('zii.widgets.grid.CGridView', array(
    'dataProvider'=>$model->search(),
    'columns'=>array(
        'firstname',
        'lastname',
        array(
            'name'=>'eyecolor_code',
            'value'=>'Lookup::item("eyecolor",$data->eyecolor_code)',
        ),
    ),
));
```

Abitekstide moodul

Sissejuhatus

Abitekstide moodul lisab rakendusele *abitekstide kuvamise ja haldamise* funktsionaalsuse. Abitekstide moodul asub keskses laienduste kaustas **`yii/extensions/modules/help`**. Selleks, et ühendada abitekstide moodul rakendusega, ei ole vaja kopeerida ühtegi faili, vaid piisab ainult rakenduse konfiguratsioonifaili täiendamisest. Samuti ei ole vaja luua andmebaasi tabelit, sest seda saab teha moodulis endas *install* kontrolleri kaudu.
Installeerimine

Abitekstide mooduli installeerimise eelduseks on, et rakendusele on juba loodud ühendus andmebaasiga.

1. Minimaalne kood, mis tuleb abitekstide mooduli rakendamiseks lisada konfiguratsioonifaili **`rakendus/protected/config/main.php`** on järgmine:

```
// autoloading
'import'=>array(
    'ext.modules.help.models.*',
),
// modules
'modules'=>array(
    'help'=>array(
        'class'=>'ext.modules.help.HelpModule',
    ),
```

```
),
```

2. Salvesta muudatused, logi sisse *admin* kasutajana ja ava lehekülg http://xxx/rakendus/index.php/help/install

3. Järgi juhiseid ja loo mooduli jaoks vajalik andmebaasi tabel. Tabel luuakse automaatselt järgmise käsuga:

```
CREATE TABLE $helpTable
(
  id serial NOT NULL,
  code character varying(64),
  title_et character varying(256),
  content_et text,
  title_en character varying(256),
  content_en text,
  CONSTRAINT pk_help PRIMARY KEY (id)
);
```

4. Moodul on nüüd kasutamiseks valmis.

Aadressil http://xxx/rakendus/index.php/help avaneb lehekülg "Abitekstid", kus kõik autenditud kasutajad saavad abitekste lisada ja muuta.

Abitekstide kustutamise õigus on ainult *admin* kasutajal.

Konfigureerimine

Abitekstide mooduli juures saab konfigureerida veel järgmisi muutujaid:

```
public $helpTable='lookup';
public $helpLayout;
public $leftPortlets=array();
public $rightPortlets=array();
public $rbac=false;
```

Seletame seda järgmise konfiguratsiooni näitel:

```
// autoloading
'import'=>array(
    'ext.modules.help.models.*',
),
// modules
'modules'=>array(
    'help'=>array(
        'class'=>'ext.modules.help.HelpModule',
        'helpTable'=>'tbl_help',
            'rbac'=>'manageHelps',
            'lookupLayout'=>'application.views.layouts.rightbar',
        'rightPortlets'=>array(
            'ptl.RightMenuTop'=>array(),
            'ptl.RightMenuBottom'=>array(),
        ),
    ),
```

```
),
```

1. Abitekstide moodul loob vaikimisi tabeli nimega "help". Kui on vaja kasutada mõnda muud tabeli nime, saab seda määrata muutuja *helpTable* kaudu:

```
'helpTable'=>'tbl_help',
```

2. Abitekstide moodulit saab kasutada ka koos rollipõhise juurdepääsukontrolliga. Kui muutuja *rbac* on määramata (st on vaikimisi false), siis kontrollitakse juurdepääsu moodulile kasutajanime põhjal:

- kõigil autenditud kasutajatel on ligipääs mooduli lehekülgedele, kus nad saavad lisada ja muuta abitekste
- ainult kasutajal *admin* on lisaks võimalik ka abitekste kustutada

Aga oletame, et rakenduses on võetud kasutusele RBAC definitsiooniga:

```
$auth = Yii::app()->authManager;
$auth->createOperation('manageHelps','View, add and update helps');
$role = $auth->createRole('contentManager');
$role->addChild('manageHelps');
$role = $auth->createRole('siteManager');
$role->addChild('contentManager');
$auth->save();
```

Selleks, et ligipääs abitekstide moodulile oleks ainult neis rollides, millele on omistatud operatsioon *manageHelps* (antud juhul on need siis rollid *contentManager* ja *siteManager*), tuleb mooduli konfiguratsioonis määrata:

```
 'rbac'=>'manageHelps',
```

Nüüd on ligipääs moodulile reguleeritud järgmiselt:

- ainult neil autenditud kasutajatel, kes on *contentManager* või *siteManager* rollis, on ligipääs mooduli lehekülgedele, kus nad saavad lisada ja muuta abitekste
- ainult kasutajal *admin* on võimalik lisaks ka abitekste kustutada

3. Rakenduse konfiguratsioonis saab määrata, millises küljenduses ja koos milliste portletitega mooduli lehekülgi selle rakenduse kontekstis kuvatakse. Näitena toodud konfiguratsioon

```
'helpLayout'=>'application.views.layouts.rightbar',
'rightPortlets'=>array(
    'ptl.RightMenuTop'=>array(),
    'ptl.RightMenuBottom'=>array(),
),
```

määrab, et abitekstide mooduli lehekülgi kuvatakse antud rakenduses nii, et paremal pool on alati nähtavad kaks portletit.

Abitekstide kasutamine rakenduses

1. Abitekstide moodul toob rakendusse meetodi *Help::item()*, mis on eriti sobilik

kasutamiseks staatilistel lehekülgedel (mis on hõlpsasti kuvatavad *CViewAction* kaudu).

Oletame, et rakenduses on vaja luua leheküljed "Rakenduse kohta" ja "Autorite kohta".

Selleks tuleb esmalt need tekstid abitekstide moodulis luua, andes neile koodiks näiteks vastavalt "about_application" ja "about_authors".

Seejärel tuleb kausta **rakendus/protected/views/site/page**s lisada vastavad *view*-failid, näiteks "about_application.php" ja "about_authors.php"

Neis *view*-failides saab siis hõlpsalt kuvada vajalikku teksti. Näiteks faili about_application.php võib kirjutada:

```
<h1><?php echo Help::item('about_application','title'); ?></h1>
<?php echo Help::item('about_application','content'); ?>
```

Juhul, kui soovitakse, et kohe abiteksti kõrvale ilmuks ikoon, mis avab abiteksti muutmise vormi (*OnSite* redigeerimine), tuleb määrata kolmas (ehk *edit*) parameeter *true:*

```
<?php echo Help::item('about_application','title',true); ?>
```

Vajadusel saab selle parameetri määramiseks kasutada ka tingimust:

```
<?php echo Help::item('about_application','title',
!Yii::app()->user->isGuest); ?>
```

2. Juhul, kui rakenduse skelett on genereeritud *xwebapp* käsuga, on rakendusega kohe seotud ka javascript *common.js,* kus on juba olemas vajalik funktsioon abitekstide mooduli kasutamiseks *Ajax-i* kaudu.

Niisiis, kui rakenduses on vaja kuvada linki, mis loeb Ajax-i kaudu andmebaasist vajaliku abiteksti ja kuvab selle eraldi aknakeses, on vaja programmi kirjutada ainult:

```
echo CHtml::link(Yii::t('ui','Explain something'),
array('/help/default/view','code'=>'explain_something'),
array('class'=>'openhelp'));
```

Haldusüksuste moodul

Sissejuhatus

Rahvusarhiivi haldusüksusi hallatakse keskselt rakenduses http://www.eha.ee/labs/haldusyksused/. Kõik teised rakendused, kus on vaja kasutada haldusüksusi, impordivad haldusüksuste hierarhia andmed kesksest andmebaasist rakenduse andmebaasi. Haldusüksuste moodul lihtsustab andmete importimist ja kasutamist.

Installeerimine

Haldusüksuste mooduli installeerimise eelduseks on, et rakendusele on juba loodud ühendus andmebaasiga.

1. Minimaalne kood, mis tuleb abitekstide mooduli rakendamiseks lisada konfiguratsioonifaili **`rakendus/protected/config/main.php`** on järgmine:

```
// autoloading
'import'=>array(
    'ext.modules.au.models.*',
),
// modules
'modules'=>array(
    'au'=>array(
        'class'=>'ext.modules.au.AuModule',
        'databaseHost'=>'192.168.113.3' // vajalik, kui ei ole localhost
    ),
),
```

2. Salvesta muudatused, logi sisse *admin* kasutajana ja ava lehekülg http://xxx/rakendus/index.php/au/install

3. Järgi juhiseid ja loo mooduli jaoks vajalikud andmebaasi tabelid. Tabelid luuakse automaatselt järgmiste käsudega:

```
CREATE TABLE $adminUnitTypeTable
(
  id integer,
  name_et character varying(256),
  name_en character varying(256),
  CONSTRAINT pk_admin_unit_type PRIMARY KEY (id)
);


CREATE TABLE $adminUnitTable
(
  id integer,
  parent_id integer,
  lft integer,
  rgt integer,
  type_id integer,
  name_et character varying(256),
  name_de character varying(256),
  name_en character varying(256),
  name_alias character varying(256),
  year_start integer,
  year_end integer,
  remarks text,
  ce_lat numeric,
  ce_lon numeric,
  zoom integer,
  sw_lon numeric,
  sw_lat numeric,
  ne_lon numeric,
  ne_lat numeric,
  CONSTRAINT pk_admin_unit PRIMARY KEY (id)
);
```

4. Aadressil http://xxx/rakendus/index.php/au avaneb nüüd lehekülg, kus on kolm valikut:

Impordi andmed
Sellel lingil klõpsates imporditakse kesksest andmebaasist kõik haldusüksuste andmed rakenduse andmebaasi (ülaltoodud tabelitesse).

Uuenda indekseid
Haldusüksuste hierarhia on üsna mahukas puu-struktuur, mistõttu seda on teatud juhtudel (näiteks juhul, kui on vaja teostada otsingut ühest harust või alampuust) tõhusam kasutada nn. nested-set algoritmi kaudu (selle kohta uuri vajadusel internetist). Nested-set algoritm eeldab spetsiaalseid *left* ja *right* väärtusi iga rea kohta. Neid väärtusi keskses andembaasis ei ole, kuna neid on tülikas uuendada, kui puustruktuuri muudetakse. Seepärast, kui rakenduses on vaja kasutada nested-set algoritmi, tuleb pärast andmete importimist uuendada indekseid ehk genereerida *left* ja *right* väärtused.

Kärbi puud
Juhtudel, kui rakenduses on vaja kasutada ainult üht haldusüksuste puu haru, klõpsi sellel lingil ja vali avanevas aknas vajaliku haru ülemine tipp ning vajuta "Saada".

Haldusüksuste kasutamine rakenduses

1. Selleks, et luua *autocomplete* väli, mis pakub haldusüksuste hierarhiast *distinct* (koha)nimesid, kasuta järgmist koodi:

In (request) controller

```
public function actions()
{
    return array(
        'suggestAuPlaces'=>array(
            'class'=>'ext.actions.XSuggestAction',
            'modelName'=>'AdminUnit',
            'methodName'=>'suggestPlaces',
        ),
    );
}
```

In view

```
$this->widget('zii.widgets.jui.CJuiAutoComplete', array(
    'model'=>$model,
    'attribute'=>'name',
    'source'=>$this->createUrl('request/suggestAuPlaces'),
    'options'=>array(
        'minLength'=>2,
        'delay'=>100
    ),
    'htmlOptions'=>array(
        'size'=>'20'
    ),
```

```
));
```

2. Selleks, et luua *autocomplete* väli, mis pakub haldusüksuste hierarhiast üksuseid koos kuuluvusahelaga kasuta järgmist koodi:

In (request) controller

```
public function actions()
{
    return array(
        'suggestAuPlaces'=>array(
            'class'=>'ext.actions.XSuggestAction',
            'modelName'=>'AdminUnit',
            'methodName'=>'suggestHierarchy',
        ),
    );
}
```

In view

```
$this->widget('zii.widgets.jui.CJuiAutoComplete', array(
    'model'=>$model,
    'attribute'=>'name',
    'source'=>$this->createUrl('request/suggestAuHierarchy'),
    'options'=>array(
       'minLength'=>2,
       'delay'=>100
    ),
    'htmlOptions'=>array(
        'size'=>'20'
    ),
));
```

3. Selleks, et kuvada haldusüksuste hierarhiat dünaamilise puuna, kasuta järgmist koodi:

```
public function actions() {
    return array(
        'fillAuTree'=>array(
            'class'=>'ext.actions.XFillTreeAction',
            'modelName'=>'AdminUnit',
            'showRoot'=>false,
        ),
    );
}
```

4. Kui on vaja kuvada üksuse kuuluvusahelat, saab kasutada järmisi võimalusi

```
echo $model->getPathText(11074042,false); // ilma ülemise üksuseta
```
Näide: Eestimaa kubermang / Harju maakond / Hageri kihelkond / Kohila vald (1866-1917)

```
echo $model->pathText;
```
Näide: Eesti (kuni 1917) / Eestimaa kubermang / Harju maakond / Hageri kihelkond / Kohila vald (1866-1917)

5. Kui on vaja kuvada üksuse navigatsiooniahelat, saab kasutada järgmisi võimalusi

```
echo $model->getBreadcrumbs(11074042,false); // ilma ülemise üksuseta
```
Näide: Eestimaa kubermang / Harju maakond / Hageri kihelkond / Kohila vald (1866-1917)

```
echo $model->breadcrumbs;
```
Näide: Eesti (kuni 1917) / Eestimaa kubermang / Harju maakond / Hageri kihelkond / Kohila vald (1866-1917)

Kasutajaliidese standardlahendused

Portletite rakendamine

Sissejuhatus

Yii raamistik lubab hoida portleteid ka kaustas "components", aga parim praktiga (mida nõuab käesolev standard ja millest lähtub *xwebapp* käsk) on hoida portletid eraldi kaustas **`rakendus/portected/portlets`**

Portletid võivad olla Yii baasklassi CWidget laiendused (nagu näiteks keelevaliku portlet,

millest oli juttu eespool) või CWidget klassi laiendava XPortlet klassi laiendused.

XPortlet

Kasutajaliidestes on sageli vaja luua spetsiifilise kujundusega "kaste", mis näitavad ühte ja sama sisu erinevatel lehekülgedel.

Selliste "kastide" loomiseks saab kasutada laiendust **`yii/extensions/widgets/portlet/XPortlet.php`**, mis on juba vaikimisi ühendatud *xwebapp* käsu abil loodud rakenduse skeletiga.

1. Loo portleti jaoks uus fail **`rakendus/protected/portlets/SomeName.php`** ja kirjuta sinna portleti klass, mis laiendab XPortlet klassi

```
class SomeName extends XPortlet
```

Selles klassis võid Sa vajadusel ülekirjutada baasklassi atribuudid:

```
public $cssFile;
public $visible=true;
public $title;
public $width='100%';
public $cssClass='portlet';
public $headerCssClass='header';
public $contentCssClass='content';
```

Selles klassis võid Sa näiteks luua meetodeid, mis pöörduvad mudelite poole:

```
public function getContentFromModel()
{
    return ModelName::model()->findContent();
}
```

Lõpuks tuleb ülekirjutada baasklassi renderContent meetod. Näiteks:

```
protected function renderContent()
{
    $this->render('someName',array(
        'data'=>$this->getContentFromModel())
    );
}
```

2. Meetod renderContent võib väljastada HTML koodi ka otse või mõne *widgeti* kaudu, aga juhul kui on vaja kasutada eraldi *view*-faili, nagu ülaltoodud näites, loo uus fail

rakendus/protected/portlets/views/someName.php

ja kuva seal andmeid, mis on defineeritud portleti *renderContent* meetodis:

```
<?php echo $data; ?>
```

või kasuta portleti spetsiifilisi meetodeid andmete kuvamiseks:

```
<?php foreach($this->getContentFromModel() as $key=>$value): ?>
<?php echo $key.$value; ?>
<?php endforeach; ?>
```

3. Portleti sisestamiseks view failidesse kasuta käsku:

```
 <?php $this->widget('ptl.SomePortlet'); ?>
```

Ka sisestamisel on võimalik vaikimisi seatud atribuutide väärtused üle kirjutada:

```
$this->widget('ptl.SomePortlet', array(
    'title'=>'Pealkiri',
    'width'=>'500px',
    'cssClass'=>'xpotlet',
    'headerCssClass'=>'xheader',
    'contentCssClass'=>'xcontent',
    'visible' => !Yii::app()->user->isGuest,
));
```

Portleteid on võimalik ka cache'sse salvestada:

```
if ($this->beginCache('examplePortlet', array('duration'=>3600)))
{
    $this->widget('ptl.ExamplePortlet');
    $this->endCache();
}
```

aga selle eelduseks on, et Sa oled konfiguratsioonifailis **rakendus/protected/config/main.php** lisanud (või kommenteerinud sisse) read:

```
'cache'=>array(
    'class'=>'CDbCache',
),
```

Standardiseeritud portletid

Repositooriumis **`yii/repository/portlets`** on järgmised standardiseeritud portletid:

- Search
  - otsivorm, mida saab näidata püsivalt kõigil lehekülgedel
  - muuda portleti klassi ja view-faili vastavalt vajadusele
  - view-faili saab genereerida ka *Form Generatori search template* järgi

- StaticMenu
  - staatiline multilevel menüü
  - linkide nimed ja aadressid määra meetodis *getMenuData()*
  - muuda portleti faili- ja klassinimi (*StaticMenu)* informatiivsemaks

- ManageMenu
  - staatiline multilevel menüü moodulite jaoks

- UserLogin
  - sisselogimise vorm, mida saab näidata püsivalt kõigil lehekülgedel
  - portleti sisestamisel määra *'visible' => Yii::app()->user->isGuest*

- UserMenu
  - sisseloginud kasutaja menüü (kus on muuhulgas ka väljalogimise link)
  - portleti sisestamisel määra *'visible' => !Yii::app()->user->isGuest*

- ContextMenu.php
  - menüü, mis on mõeldud kasutamiseks mistahes view-faili kontekstis, rakenduse kontrolleris defineeritud $this->menu muutuja kaudu.
  - näiteks, kui soovid kuvada menüüd parempoolses xportletis, tuleb view failis täita $this->menu muutuja CMenu formaadis:

```
$this->menu=array(
    array(
       'label'=>,
       'url'=>,
       'visible'=>,
    ),
```

  - ja määrata *layout* ja *rightPortlets* muutujad järgmiselt:

```
$this->layout='sidebars';
$this->rightPortlets['ptl.ContextMenu']=array();
```

Portletite rakendamine erinevates küljendustes

Portleteid on sageli vaja kuvada erinevates küljendustes, mitu tükki korraga.
Tüüpilised küljendused portletite kuvamiseks on järgmised:

Rightbar küljendus paigutab portletid lehekülje sisu suhtes parempoolsesse tulpa.
Leftbar küljendus paigutab portletid lehekülje sisu suhtes vasakpoolsesse tulpa.
Sidebars küljendus paigutab portletid lehekülje sisu suhtes nii vasakule kui ka paremale.

Portletite rakendamine erinevates küljendustes on standardiseeritud järgmiselt:

1. Rakenduse skeletiga tulevad kaasa vajalikud kujunduse mallid:

rakendus/protected/views/layouts/leftbar.php
rakendus/protected/views/layouts/rightbar.php
rakendus/protected/views/layouts/sidebars.php

Ja rakenduse kontrolleris **rakendus/protected/components/Controller.php**
on eeldefineeritud vajalikud muutujad:

```
public $leftPortlets=array();
public $rightPortlets=array();
```

2. Kui lehekülje kontrolleri kõik vaated (index, admin, update, create, view jne.) peavad kasutama sama *layouti*, siis tuleb kontrollerisse lisada näiteks järgmised read:

```
public $layout='sidebars';

function init()
{
    parent::init();

    $this->leftPortlets['ptl.Search']=array(
        'title'=>Yii::t('ui','Search'),
    );
    $this->leftPortlets['ptl.Links']=array(
        'title'=>Yii::t('ui','Links'),
    );
    $this->rightPortlets['ptl.UserLogin']=array(
        'title'=>Yii::t('ui','Login'),
        'visible'=>Yii::app()->user->isGuest
    );
    $this->rightPortlets['ptl.UserMenu']=array(
        'title'=>Yii::app()->user->name,
        'visible'=>!Yii::app()->user->isGuest
    );
}
```

3. Aga vajadusel saab layouti määrata ka igas *view*-failis eraldi:

```
$this->layout='sidebars';
$this->leftPortlets['ptl.Search']=array();
$this->rightPortlets['ptl.Link']=array();
```

Otsisüsteemi arendamine

Otsisüsteemiks nimetame süsteemi, mis koosneb järgmistest osadest:

- Lihtotsingu vorm, mida saab kuvada erinevatel lehekülgedel, näiteks avalehel ja tulemuste tabeli kohal.
- Täpsema otsingu vorm eraldi leheküljel.
- Tulemuste tabel, millel on kaks vaadet:
  - *sisseloginud kasutaja vaade*, mille puhul näidatakse iga rea juures muutmise ja kustutamise nuppe;
  - *avalik vaade*, mille puhul muutmise ja kustutamise nuppe ei näidata.

Nõuded otsisüsteemile:

- Ei tohi kasutada *AJAX*-it.
- Igal leheküljel peab olema *URL*, mille kaudu saab seda lehekülge *bookmark*-ida.
- Otsingu tulemuste lehekülje *URL-is* ei tohi olla tühje parameetreid.
- Täidetud täpsema otsingu vormi juurde peab saama tagasi pöörduda asukoha menüü kaudu.

Kuigi otsisüsteem pärib reeglina andmeid mitmest erinevast andmebaasi tabelist, on alati olemas üks põhitabel, kus on enamus otsitavatest andmetest. Järgnev õpetus eeldab, et selle tabeli nimi on "unit".

1. Kopeeri repositooriumist **`repository/portlets/Search.php`** rakendusse **`rakendus/portlets/Search.php`** ja kirjuta seal `'controller/action'` asemele `'unit/search'`

Sellel klassil põhineb lihtotsingu portlet.

2. Sisene **Gii** moodulisse, ava **Model Generator** ja täida vormi väljad järgmiselt:

- Table Name = unit
- Model Class = Unit
- Base Class = CActiveRecord
- Model Path = application.models
- Code Template = search

ja genereeri fail **`models/Unit.php`**
Selle klassi kaudu päritakse andmeid "unit" tabelist.

3. Ava uuesti **Model Generator** ja täida vormi väljad järgmiselt:

- Table Name = unit
- Model Class = SearchForm
- Base Class = CActiveRecord
- Model Path = application.models
- Code Template = searchForm

ja genereeri fail **`models/SearchForm.php`**
See klass on vajalik otsivormide valideerimiseks ja filtreerimiseks.

4. Ava **Form Generator** ja täida vormi väljad järgmiselt:

- Model Class = SearchForm
- View Name = search
- View Path = application.portlets.views
- Scenario = search
- Code Template = search

ja genereeri fail **`portlets/views/search.php`**
Seda faili kasutab lihtotsingu portlet otsivormi kuvamiseks.

5. Ava uuesti **Form Generator** ja täida vormi väljad järgmiselt:

- Model Class = SearchForm
- View Name = searchAdvanced
- View Path = application.views.site
- Scenario = searchAdvanced
- Code Template = search

ja genereeri fail **`views/site/searchAdvanced.php`**
See fail kuvab täpsema otsingu vormi.

Pärast faili genereerimist pakub generaator kopeerimiseks ka aktsiooni koodi

```
public function actionSearchAdvanced()
```

Kopeeri see meetod *SiteController* klassi

`rakendus/protected/controllers/SiteController.php`

ja kirjuta seal '`controller/action`' asemele '`unit/searchAdvanced`'

6. Ava **Crud Generator** ja täida vormi väljad järgmiselt:

- Model Class = Unit
- Controller ID = unit
- Base Controller Class = Controller Code
- Template = search

ja genereeri kõik *controller* ja *view* failid.
Need failid kuvavad tulemuste tabelit, lisamise ja muutmise vormi.

7. Ava rakenduse avalehe *view* fail **rakendus/portected/views/site/index.php** ja lisa sinna lihtotsingu portlet

```
<?php $this->widget('ptl.Search'); ?>
```

8. Ava rakenduse põhimenüü portlet

rakendus/portected/portlets/MainMenu.php

ja lisa täpsema otsingu lehekülg põhimenüüsse

```
array(
    'label'=>Yii::t('ui', 'Advanced Search'),
    'url'=>array('/site/searchAdvanced')
),
```

Sellega on otsisüsteemi skelett kasutamiseks valmis. Loomulikult tuleb kõiki genereeritud faile muuta ja täiendada lähtudes käesoleva rakenduse spetsiifilistest vajadustest.

Tõstutundetu otsing ja metasümbolitega otsing

Kui rakendus on UTF-8 kodeeringus, siis PostgreSql LIKE päringud on tõstutundlikud. See tähendab, et kui Gii moodul genereerib mudelisse otsingu meetodi kriteeriumi järgmiselt:

```
$criteria=new CDbCriteria;
$criteria->compare('lastname', $this->lastname, true);
```

siis otsing "ab" leiab "Baber", aga ei leia "Abby".

Tõstutundetu otsingu tegemiseks tuleb kasutada XDbCriteria klassi (mis on CDbCriteria laiendus), mis asub kaustas *extensions/components/database*

1. Ava rakenduse konfiguratsioonifail **rakendus/protected/config/main.php** ja impordi *XDbCriteria* klass:

```
'import'=>array(
    'ext.components.database.*',
),
```

2. Muuda otsingu meetodeid nii, et nad kasutaksid *XDbCriteria* klassi *icompare* meetodit, mis tagab tõstutundetu otsingu:

```
$criteria=new XDbCriteria;
$criteria->icompare('lastname', $this->lastname, true);
```

Lisaks tõstutundetule otsingule võimaldab *XDbCriteria* klass ka metasümolitega otsingut.

Metasümbolitega otsing tähendab, et tärniga (*) võib tähistada ükskõik millist kogust (sealhulgas ka 0) ükskõik milliseid tähti ning küsimärgiga (?) ükskõik millist ühte tähte.

Näiteks:

- **tamm* leiab Karotamm, Tamm, Veskitamm.
- *tamm** leiab Tamm, Tammekunn, Tammeveski, Tammsaar.
- **man** leiab Tammann, Tamman, Strandmann, Mannerg.
- *???mann* leiab Tammann, Aasmann, Eermann, aga ei leia pikemaid ega lühemaid nimesid.

Metasümbolitega otsingut võimaldab *XDbCriteria* klassi *mcompare* meetod:

```
$criteria=new XDbCriteria;
$criteria->mcompare('lastname', $this->lastname);
```

Tagasipöördumine

Kasutajaliideses tuleb peaaegu alati ette vajadus:

- kuvada "Tagasi" link või "Loobu" nupp, mis pöördub tagasi eelmisele leheküljele, kust antud lehekülje poole pöörduti;

- pöörduda pärast "Uus" või "Muuda" operatsiooni tagasi leheküljele, kust vastava vormi poole pöörduti.

Kuna ühe lehekülje poole võivad pöörduda mitu erinevat lehekülge, peab lehekülg, kuhu on vaja tagasi pöörduda (edaspidi: algataja lehekülg) saatma enda aadressi URL'iga kaasa leheküljele, kus rakendatakse tagasipöödumist (edaspidi: tagasipöörduv lehekülg).

Sellise funktsionaalsuse loomiseks saab kasutada laiendust **`yii/extensions/behaviors/XReturnableBehavior.php`**, mis on juba vaikimisi ühendatud *xwebapp* käsu abil loodud rakenduse skeletiga.

1. Algataja lehekülje view-failis moodusta link tagasipöörduvale leheküljele järgmiselt:

```
<?php echo CHtml::link('Update',
$this->createReturnableUrl('post/update',array('id'=>$post->id))); ?>
```

ja tagasipöörduva lehekülje view-failis moodusta tagasi link algataja leheküljele nii:

```
<?php echo CHtml::link('Back', $this->getReturnUrl());?>
```

2. Juhul, kui algataja leheküljelt pöörduti loomise või muutmise vormi poole, siis saab vastavas kontrolleri aktsioonis, pärast andmete salvestamist, pöörduda tagasi algataja leheküljele:

```
if($this->goBack()) $this->goBack();
```

IP aadressi põhine juurdepääsukontroll

Rakenduse skelett, mis on genereeritud *xwebapp* käsuga, on kohe valmis IP aadressi põhise juurdepääsukontrolli rakendamiseks.

1. Ava **rakendus/protected/components/Controller.php** ja täienda vastavalt vajadusele:

```
public $ips = array(
    '192.168.111.*',   // Liivi sisev
    …
);
```

2. Kontrollerites saab piirata juurdepääsu aktsioonidele järgmiselt:

```
public function accessRules()
{
    return array(
        array('allow',
            'actions'=>array('index'),
                'users'=>array('*'),
            ),
            array('allow',
                'actions'=>array('admin'),
                'ips'=>$this->ips,
            ),
            array('deny',
                'users'=>array('*'),
            ),
     );
}
```

3. View-failides saab kontrollida juurdepääsu html-sisule järgmiselt:

```
<?php if($this->isIpMatched()): ?>
    <p>Seda näevad ainult lubatud ip aadressidelt</p>
<?php endif; ?>
```

4. Ligipääs portletitele määratakse järgmiselt:

```
$this->widget('ptl.SomePortlet', array(
    'visible'=>$this->isIpMatched()
));
```

Trükivaade

Rakenduse skelett, mis on genereeritud *xwebapp* käsuga, on trükivaate lisamiseks valmis:

- trükivaate küljendus on failis **views/layouts/print.php**
- ja rakenduse kontrolleri **rakendus/protected/components/Controller.php** `init()` meetodis on read, mis võtavad kasutusele trükivaate, kui rakenduse poole pöördutakse päringuga, mis sisaldab parameetrit *printview*

Niisiis türkivaate lisamiseks rakendusele tuleb lisada ainult vastav link. Eeldades, et me soovime lisada lingi, mis avab prindivaate leheküljest *http://xxx/rakendus/controller/view?id=1*, peame kirjutama:

```
<?php echo CHtml::link(Yii::t('ui','Print'),
 array('view','id'=>1,'printview'=>1),
 array('rel'=>'external')
); ?>
```

Parim viis printimise lingi lisamiseks on kasutada rakenduse kontrolleri **`rakendus/protected/components/Controller.php`** `createPrintUrl()` meetodit:

```
<?php echo CHtml::link(Yii::t('ui','Print'),
$this->createPrintUrl(),array('rel'=>'external')); ?>
```

või kui linki on vaja lisada menüüsse, saab seda meetodit kasutada järgmiselt:

```
$this->menu=array(
    array(
        'label'=>Yii::t('ui', 'Print'),
        'url'=>$this->createPrintUrl(),
        'linkOptions'=>array('rel'=>'external'),
    ),
);
```

NB! Ülaltoodud näidetes esinev *rel=external* (mis tagab, et prindivaade avaneb uues aknas) eeldab vastava javaskripti olemasolu (*xwebapp* käsuga geneeritud rakendustes on see skript js/common.js sees). Selle asemel võib ka kasutada *target=_blank*

Kui vaates on vaja määrata osad, mida näidatakse või ei näidata ainult prindivaates, siis saab seda teha näiteks nii:

```
<?php if(Yii::app()->layout=='print'):?>
    Seda näeb ainult prindivaates
<?php endif; ?>
```

Mitmekeelsus

Kuna tõlkefail põhineb Inglise keelel, on *xwebapp* käsu poolt loodud rakendus lihtsalt muudetav kakskeelseks (Eesti-Inglise).

Kõige lihtsam oleks mitmekeelsust rakendada *cookie*'de kaudu (nagu seda soovitasid ka käesoleva dokumendi varasemad versioonid), aga see pole kasutaja seisukohast kõige parem lahendus, kuna sel juhul ei saa url-ga viidata erinevatele keeleversioonidele. Näiteks kui ingliskeelne kasutaja salvestab ingliskeelse lehekülje lingi, viitab see hiljem (*cookie* aegudes) eestikeelsele leheküljele.

Seepärast on parem hoida keelevalikut lehekülje aadressis. Sellise lahenduse jaoks on olemas standardiseeritud laiendus **`extensions/components/language/`** mida saab kasutada järgmiselt:

1. Ava rakenduse konfiguratsioonifail **`rakendus/protected/config/main.php`** ja defineeri *urlManager* komponent järgmiselt:

```
'urlManager'=>array(
    'class' => 'ext.components.language.XUrlManager',
    'urlFormat'=>'path',
    'showScriptName'=>true,
    'appendParams'=>false,
    'rules'=>array(
        '<language:\w{2}>' => 'site/index',
        '<language:\w{2}>/<_c:\w+>' => '<_c>',
        '<language:\w{2}>/<_c:\w+>/<_a:\w+>'=>'<_c>/<_a>',
        '<language:\w{2}>/<_m:\w+>' => '<_m>',
        '<language:\w{2}>/<_m:\w+>/<_c:\w+>' => '<_m>/<_c>',
        '<language:\w{2}>/<_m:\w+>/<_c:\w+>/<_a:\w+>' => '<_m>/<_c>/<_a>',
    ),
),
```

NB! Suhteliselt keerukad reeglid (rules) on vajalikud selleks, et keelaparameeter ei risustakse lehekülgede aadresse kujul `&language=et`, vaid ilmuks lühidalt kujul `http://xxx/rakendus/et/jne`

2. Ava rakenduses fail: **`rakendus/protected/views/layout/main.php`** ja lisa sinna keelemenüü:

```
$this->widget('ext.components.language.XLangMenu', array(
    'items'=>array('et'=>'Eesti','en'=>'In English'),
));
```

NB! Keelemenüü kuvamise erinevate võimaluste kohta vaata näidisrakendust.