MetaModels for developers

Christian Schiffler

Contao Konferenz 2013

9. Mai 2013

Christian „Xtra“ Schiffler
Wers'n das?

- Haupt-Entwickler von Catalog und nun MetaModels
- seit 2004 selbständig
- seit 2008 bei ~~TYPOlight~~ Contao

Contact:
please.chris@metamodel.me
http://cyberspectrum.de
http://twitter.com/discordier
http://github.com/discordier

Ablauf des Workshops
Etwas Struktur

Ablauf des Workshops
Etwas Struktur

Ablauf des Workshops
Etwas Struktur

Übersicht

1. Motivation
   - Warum MetaModels?
2. Grundlegender Aufbau
   - Begriffe in MetaModels
   - IMetaModel
   - Abfragen von Daten
   - Interfaces
3. Eigenes Attribut
   - config.php
   - Attr. Klasse

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Probleme beim Catalog

- War eigentlich mal was ganz anderes.
- Codebasis teilweise von 2007.
- Ging durch mehrere Entwickler.
- Erst mit Version 2 individuelle Feldtypen teilweise(!) möglich.
- Monolithische **ModuleCatalog.php**
- Nie wirklich stabil gewesen.

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Probleme beim Catalog

- War eigentlich mal was ganz anderes.
- Codebasis teilweise von 2007.
- Ging durch mehrere Entwickler.
- Erst mit Version 2 individuelle Feldtypen teilweise(!) möglich.
- Monolithische **ModuleCatalog.php**
- Nie wirklich stabil gewesen.

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Probleme beim Catalog

- War eigentlich mal was ganz anderes.
- Codebasis teilweise von 2007.
- Ging durch mehrere Entwickler.
- Erst mit Version 2 individuelle Feldtypen teilweise(!) möglich.
- Monolithische **ModuleCatalog.php**
- Nie wirklich stabil gewesen.

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Probleme beim Catalog

- War eigentlich mal was ganz anderes.
- Codebasis teilweise von 2007.
- Ging durch mehrere Entwickler.
- Erst mit Version 2 individuelle Feldtypen teilweise(!) möglich.
- Monolithische **ModuleCatalog.php**
- Nie wirklich stabil gewesen.

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Probleme beim Catalog

- War eigentlich mal was ganz anderes.
- Codebasis teilweise von 2007.
- Ging durch mehrere Entwickler.
- Erst mit Version 2 individuelle Feldtypen teilweise(!) möglich.
- Monolithische **ModuleCatalog.php**
- Nie wirklich stabil gewesen.

Warum MetaModels? Catalog reicht doch!?

MacKP (2010)

„aber beim nächsten Mal nehm ich einfach den Catalog... da hab ich weniger Arbeit mit!“

Probleme beim Catalog

- War eigentlich mal was ganz anderes.
- Codebasis teilweise von 2007.
- Ging durch mehrere Entwickler.
- Erst mit Version 2 individuelle Feldtypen teilweise(!) möglich.
- Monolithische **ModuleCatalog.php**
- Nie wirklich stabil gewesen.

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (Rapid Application Development).

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (Rapid Application Development).

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (Rapid Application Development).

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (Rapid Application Development).

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (**R**apid **A**pplication **D**evelopment).

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (**R**apid **A**pplication **D**evelopment).

MacKP (2013)

„bin ich froh, das es jetzt die MetaModels gibt. Die sind besser als Catalog.“

Andreas Isaak (2013)

„ich hab die Demo bisher aufgebaut ohne auch nur ein Listen-Template anfassen zu müssen das ist extrem beim Catalog war das erste Template schon nach 5min angepasst.“

Darum MetaModels. Catalog reicht eben nicht!

Lösungen durch MetaModels

- Mehrsprachigkeit von Anfang an.
- Varianten eines Eintrags werden automatisch verwaltet.
- Streng modularer Aufbau.
- Es gibt keine „*Core-Attribute*“.
- Fokus auf RAD für „*Frontendschlampen*“ (**R**apid **A**pplication **D**evelopment).

MacKP (2013)

„bin ich froh, das es jetzt die MetaModels gibt. Die sind besser als Catalog.“

Andreas Isaak (2013)

„ich hab die Demo bisher aufgebaut ohne auch nur ein Listen-Template anfassen zu müssen das ist extrem beim Catalog war das erste Template schon nach 5min angepasst.“

Übersicht

1. Motivation
   - Warum MetaModels?
2. Grundlegender Aufbau
   - Begriffe in MetaModels
   - IMetaModel
   - Abfragen von Daten
   - Interfaces
3. Eigenes Attribut
   - config.php
   - Attr. Klasse

Begriffe in MetaModels

MetaModel

- Ist die „Tabelle“.
- „Besteht“ aus Attributen.
- Kommunikations-Schnittstelle aller Klassen.
- Per Factory beziehbar.

MetaModel Attribut

- Verwaltet seine Daten.
- Definiert Widget Informationen.
- Weiss einzig um die Natur seiner Daten.
- Wird abgefragt und befüllt.

MetaModel „RenderSetting“

- Formatiert Daten aus einem Attribut.
- Verwendet Template.
- Freie Ausgabeformate (Fallback: Plaintext).
- Kommuniziert mit Filter für jumpTo.

MetaModel „FilterSetting“

- Erzeugt Filterregeln.
- Kann Frontendfilterwidgets bereitstellen.
- Erzeugt jumpTo URL auf sich selbst.
- Unterstützt Parameter.

MetaModel „Item“

- Kapselt Daten aller Attribute eines Eintrags.
- Einfache Schnittstelle zum rendern eines Eintrags.
- Schnelles speichern und Ändern von Daten.
- Weiss ob es Variante ist und wer seine Geschwister.

Begriffe in MetaModels

MetaModel

- Ist die „Tabelle“.
- „Besteht“ aus Attributen.
- Kommunikations-Schnittstelle aller Klassen.
- Per Factory beziehbar.

MetaModel Attribut

- Verwaltet seine Daten.
- Definiert Widget Informationen.
- Weiss einzig um die Natur seiner Daten.
- Wird abgefragt und befüllt.

MetaModel „RenderSetting“

- Formatiert Daten aus einem Attribut.
- Verwendet Template.
- Freie Ausgabeformate (Fallback: Plaintext).
- Kommuniziert mit Filter für jumpTo.

MetaModel „FilterSetting“

- Erzeugt Filterregeln.
- Kann Frontendfilterwidgets bereitstellen.
- Erzeugt jumpTo URL auf sich selbst.
- Unterstützt Parameter.

MetaModel „Item“

- Kapselt Daten aller Attribute eines Eintrags.
- Einfache Schnittstelle zum rendern eines Eintrags.
- Schnelles speichern und Ändern von Daten.
- Weiss ob es Variante ist und wer seine Geschwister.

Begriffe in MetaModels

MetaModel

- Ist die „Tabelle“.
- „Besteht“ aus Attributen.
- Kommunikations-Schnittstelle aller Klassen.
- Per Factory beziehbar.

MetaModel Attribut

- Verwaltet seine Daten.
- Definiert Widget Informationen.
- Weiss einzig um die Natur seiner Daten.
- Wird abgefragt und befüllt.

MetaModel „RenderSetting“

- Formatiert Daten aus einem Attribut.
- Verwendet Template.
- Freie Ausgabeformate (Fallback: Plaintext).
- Kommuniziert mit Filter für jumpTo.

MetaModel „FilterSetting“

- Erzeugt Filterregeln.
- Kann Frontendfilterwidgets bereitstellen.
- Erzeugt jumpTo URL auf sich selbst.
- Unterstützt Parameter.

MetaModel „Item“

- Kapselt Daten aller Attribute eines Eintrags.
- Einfache Schnittstelle zum rendern eines Eintrags.
- Schnelles speichern und Ändern von Daten.
- Weiss ob es Variante ist und wer seine Geschwister.

Begriffe in MetaModels

MetaModel

- Ist die „Tabelle“.
- „Besteht“ aus Attributen.
- Kommunikations-Schnittstelle aller Klassen.
- Per Factory beziehbar.

MetaModel Attribut

- Verwaltet seine Daten.
- Definiert Widget Informationen.
- Weiss einzig um die Natur seiner Daten.
- Wird abgefragt und befüllt.

MetaModel „RenderSetting“

- Formatiert Daten aus einem Attribut.
- Verwendet Template.
- Freie Ausgabeformate (Fallback: Plaintext).
- Kommuniziert mit Filter für jumpTo.

MetaModel „FilterSetting“

- Erzeugt Filterregeln.
- Kann Frontendfilterwidgets bereitstellen.
- Erzeugt jumpTo URL auf sich selbst.
- Unterstützt Parameter.

MetaModel „Item“

- Kapselt Daten aller Attribute eines Eintrags.
- Einfache Schnittstelle zum rendern eines Eintrags.
- Schnelles speichern und Ändern von Daten.
- Weiss ob es Variante ist und wer seine Geschwister.

Begriffe in MetaModels

MetaModel

- Ist die „Tabelle“.
- „Besteht“ aus Attributen.
- Kommunikations-Schnittstelle aller Klassen.
- Per Factory beziehbar.

MetaModel Attribut

- Verwaltet seine Daten.
- Definiert Widget Informationen.
- Weiss einzig um die Natur seiner Daten.
- Wird abgefragt und befüllt.

MetaModel „RenderSetting“

- Formatiert Daten aus einem Attribut.
- Verwendet Template.
- Freie Ausgabeformate (Fallback: Plaintext).
- Kommuniziert mit Filter für jumpTo.

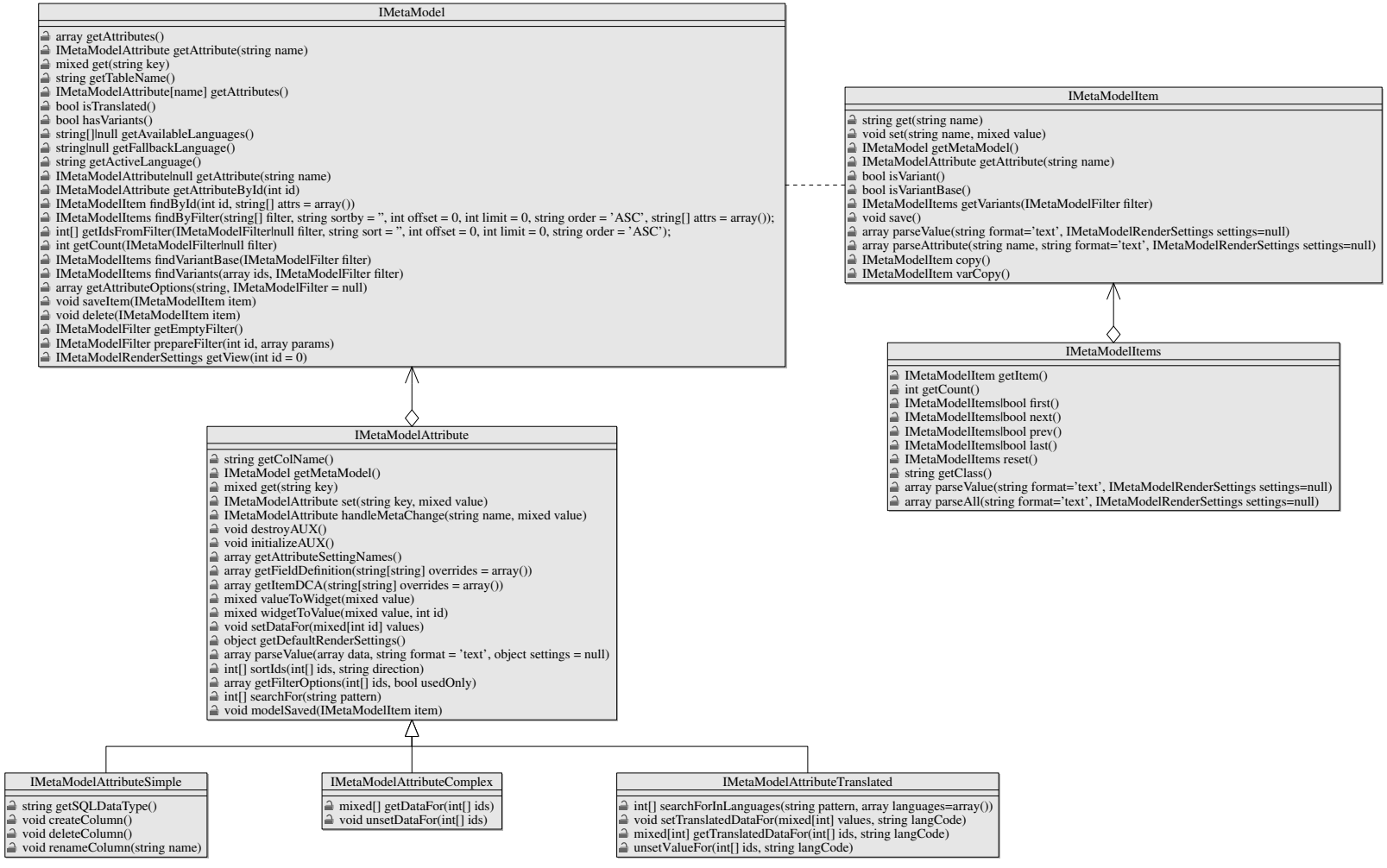
MetaModel „FilterSetting“

- Erzeugt Filterregeln.
- Kann Frontendfilterwidgets bereitstellen.
- Erzeugt jumpTo URL auf sich selbst.
- Unterstützt Parameter.

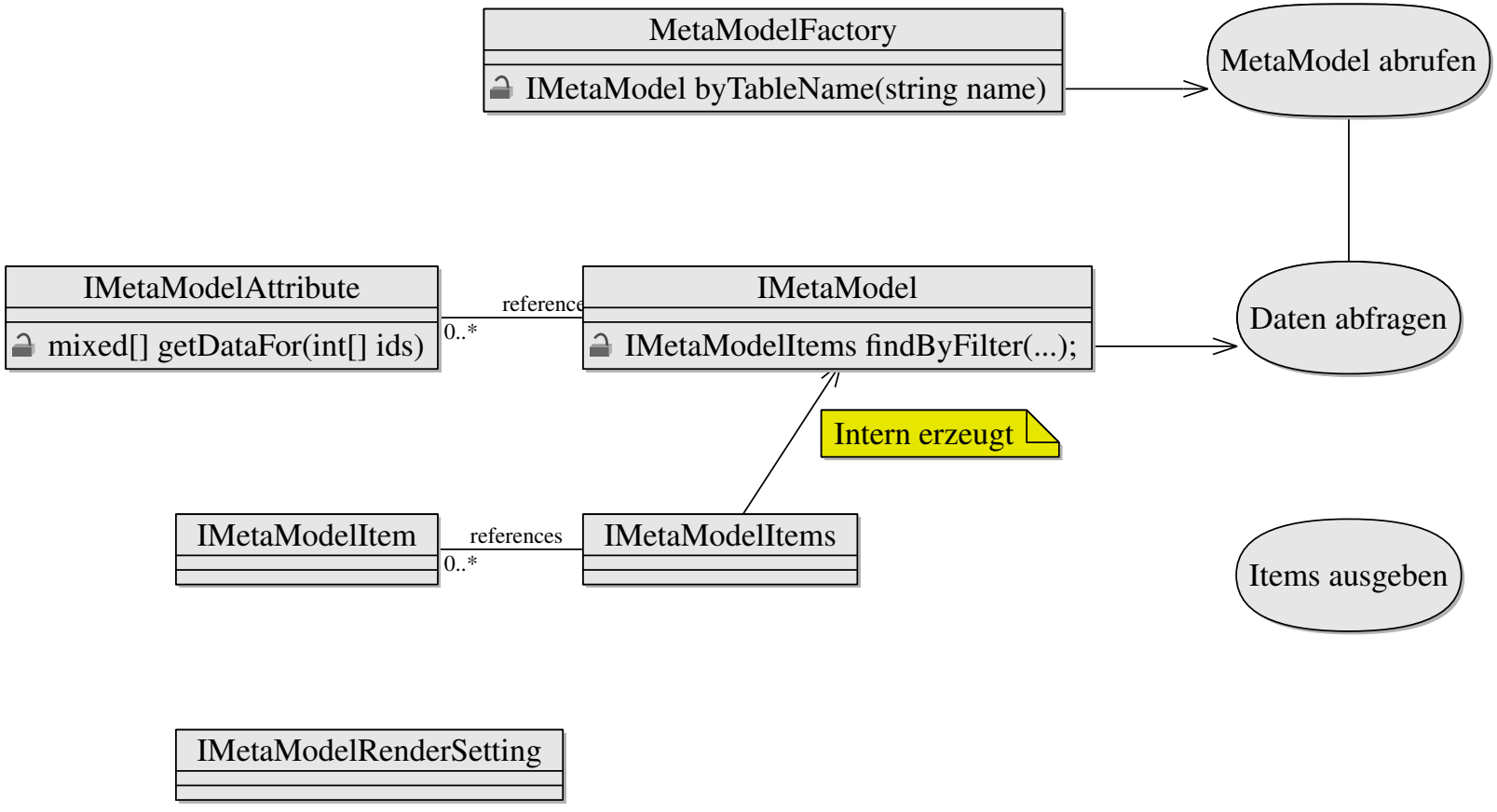
MetaModel „Item“

- Kapselt Daten aller Attribute eines Eintrags.
- Einfache Schnittstelle zum rendern eines Eintrags.
- Schnelles speichern und Ändern von Daten.
- Weiss ob es Variante ist und wer seine Geschwister.

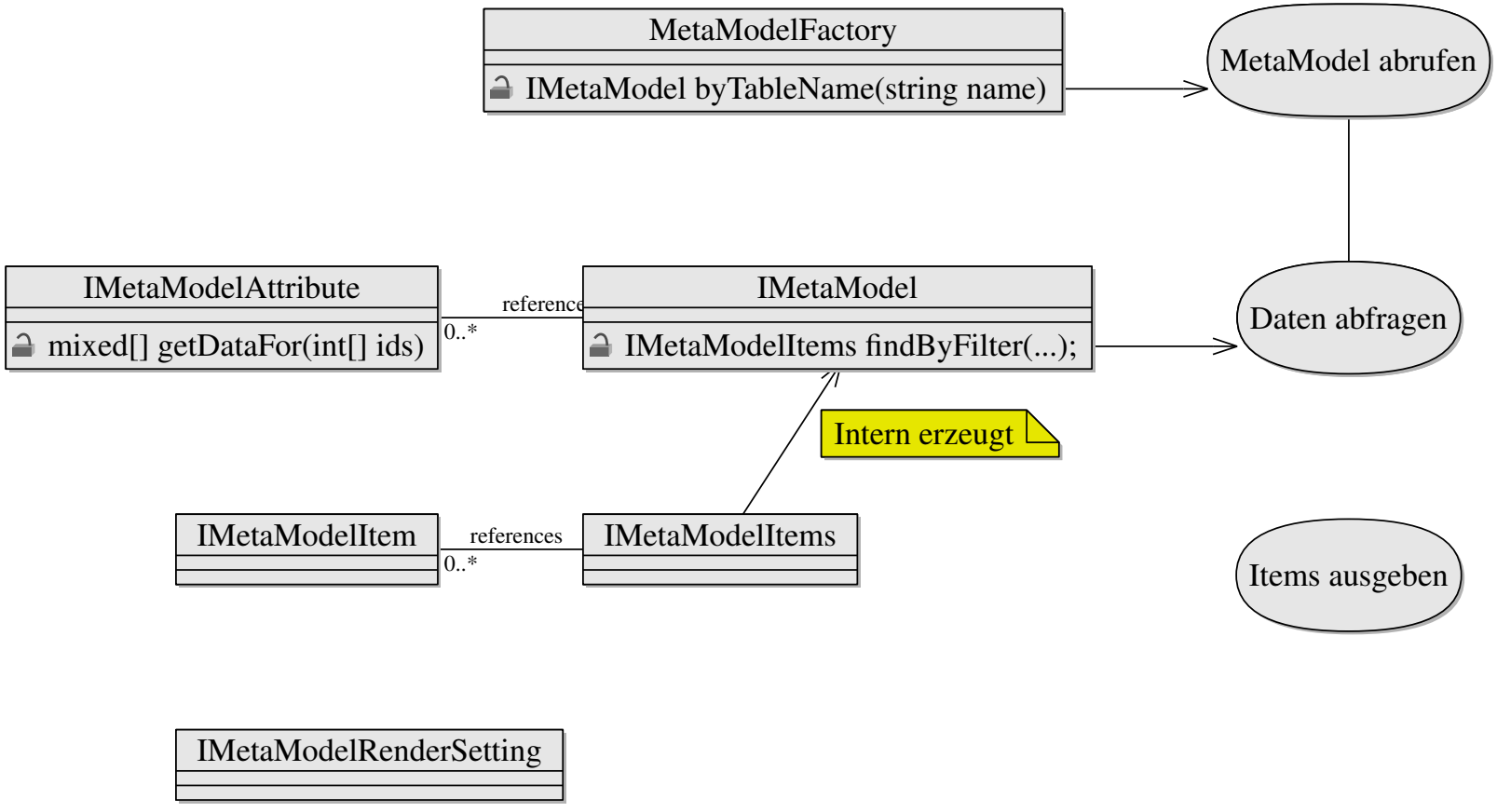
Die Interfaces dazu



Schematischer Ablauf



Schematischer Ablauf



Und nun mit Code


```php
/**
 * Query our model.
 */
$objMetaModel = MetaModelFactory::byTableName('mm_testmodel');

/**
 * Fetch an empty filter from it.
 */
$objFilter = $objMetaModel->getEmptyFilter();

/**
 * Add an Filter rule to search the attribute "alias"
 * for representations of "ck_2012", in all available
 * languages of the MetaModel.
 */
$objFilter->addFilterRule(
    new MetaModelFilterRuleSearchAttribute(
        $objMetaModel->getAttribute('alias'),
        'ck_2012',
        $objMetaModel->getAvailableLanguages()
    )
);

/**
 * Finally fetch the matching Ids from the filter and
 * tell the MetaModel to retrieve the items.
 */
$objItems = $objMetaModel->findByFilter($objFilter);
```

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Interfaces

- IMetaModel
- IMetaModelAttribute
  - IMetaModelAttributeSimple
  - IMetaModelAttributeComplex
  - IMetaModelAttributeTranslated
- IMetaModelItem
- IMetaModelItems
- IMetaModelRenderSettings

Übersicht

Interfaces

hulla

config.php


```php
<?php

/**
 * Register the attribute into the system
 */
$GLOBALS['METAMODELS']['attributes']['ck2013'] = array
(
    // Class
    'class' => 'MetaModelAttributeCK2013',
    // image to use in backend
    'image' => 'system/modules/metamodelsattribute_ck2013/html/icon.png'
);
```

MetaModelAttributeCK2013.php