leifos

ERWEITERTE METADATEN

Nutzung des ILIAS-Service

Version 0.1 | 22.04.2014
Jörg Lützenkirchen | leifos GmbH
luetzenkirchen@leifos.com
+49.221.1207.18922

leifos

1 EINFÜHRUNG

Die erweiterten Metadaten von ILIAS können von jedem Modul (oder auch Service) genutzt werden. Sie bieten eine fest definierte Anzahl von Datentypen, die in Datensätzen organisiert sind. Jedem Objekttyp, der die erweiterten Metadaten unterstützt, können 1-n dieser Datensätze zugeordnet werden. Danach stehen bei jeder Objekt-Instanz des Typs genau diese Datensätze zur Verfügung und können mit Instanz-spezifischen Daten versehen werden.

2 AKTIVIERUNG

Dem entsprechenden Objekttyp ist das Attribut *„amet“* der module.xml (bzw. service.xml) hinzuzufügen. Nach einem Reload der Kontrollstruktur sollte der Objekttyp in *Administration* > *Metadaten* > *Erweiterte Metadaten* verfügbar sein.

Beispiel:

```
<object id="crs" class_name="Course" dir="classes" default_pos="20" default_pres_pos="30"
checkbox="1" inherit="1" translate="0" allow_copy="1" rbac="1" grp="crs" export="1"
amet="1">
        […]
</object>
```

3 EINBINDUNG IN DIE EINSTELLUNGEN

Das Settings-Formular sollte über eine eigene Methode aufgebaut werden, z.B. *initSettingsForm()*. Um die erweiterten Metadaten hinzuzufügen sind nur wenige Zeilen nötig. Es sind 3 Dinge entscheidend:

- der korrekte Objekttyp
- der Modus (*MODE_EDITOR*)
- die Verwendung von *$this->record_gui* (siehe Kapitel 4)

Beispiel:

```
include_once('Services/Form/classes/class.ilPropertyFormGUI.php');
$form = new ilPropertyFormGUI();
$form->setFormAction($ilCtrl->getFormAction($this));
        […]
include_once('Services/AdvancedMetaData/classes/class.ilAdvancedMDRecordGUI.php');
$this->record_gui = new
ilAdvancedMDRecordGUI(ilAdvancedMDRecordGUI::MODE_EDITOR,'<Objekttyp>',$this->object-
>getId());
$this->record_gui->setPropertyForm($form);
$this->record_gui->parse();
```

leifos

4 SPEICHERN DER EINSTELLUNGEN

Beim Absenden des Einstellungsformulars sollten die Metadaten-Werte entsprechend der Definition der Datensätze validiert und verarbeitet werden. Falls außer den Metadaten noch andere Werte im Formular editiert werden können, ist auf eine korrekte Reihenfolge zu achten:

- *checkInput()* muss **immer** (zuerst) aufgerufen werden
- die Metadaten validieren bzw. reloaden sich über *importEditFormPostValues()*
- erst wenn beide Teile validiert sind, sollte gespeichert werden
- *setValuesByPost()* muss - sofern nötig – **vor** den Metadaten gesetzt werden
- da evtl. vorhandene Fehlermeldungen dem Formular hinzugefügt werden, ist bei einer erneuten Anzeige diese Formularinstanz zu verwenden und nicht etwa neu aufzubauen

Diese Limitierungen sind leider nötig, da die Metadaten auf Datenebene validiert werden und sich nicht auf das Formular verlassen.

Beispiel:

```
$form = $this->initSettingsForm();

$valid = true;

If(!$form->checkInput())
{
	$form->setValuesByPost();
	$valid = false;
}

if($this->record_gui->importEditFormPostValues())
{
	$this->record_gui->writeEditForm();
}
else
{
	$valid = false;
}

If($valid)
{
	// save non-metadata settings here

	ilUtil::sendSuccess($lng->txt("settings_saved"), true);
	$ilCtrl->redirect($this, "editSettings");
}

$this->editSettings($form);
```

leifos

5 AUSBLICK

Ein simples Beispiel zur Verwendung der gespeicherten Daten zu einer Objekt-Instanz findet sich in *ilAdvancedMDRecordGUI::parseInfoPage()*.

Dort werden sämtliche zugeordneten Datensätze ausgelesen und dem Infoscreen-Widget hinzugefügt.