

icoya
OpenContent

**Aktive Websites
schnell erstellen
und einfach pflegen**

Layouts/Skins

- ↘ **Erstellung**
- ↘ **Konfiguration**
- ↘ **Anpassung**

icoya OpenContent Skin Konfiguration

Über dieses Dokument

Dieses Dokument ist eine Übersicht wie man schnell und effizient vollständige, eigene Layouts für eine icoya OpenContent Installation erstellt. Diese Layouts werden nachfolgend Skins genannt. Eine Skin definiert sowohl Look & Feel als auch die komplette Anordnung der einzelnen Teile einer Seite. Für diese Anleitung ist die Kenntnis von Zope PageTemplates und METAL-Makros von enormem Vorteil. Konsultieren Sie hierzu das icoya/Zope PageTemplate Handbuch.

Über das Skins Tool

Das icoya OpenContent System basiert auf dem Zope Content Management Framework (CMF) welches zur Interaktion zwischen Content und System eine Reihe von Tools zur Verfügung stellt. Diese Tools erledigen Dienste für Inhaltsobjekte, so daß die Inhalte selbst wirklich nur Inhalte sind. Eines dieser Tools ist das Skins Tool, `portal_skins`. Das Skins Tools trennt das Site abhängige Verhalten von den eigentlichen Inhalten im System. Dies macht es möglich das Layout komplett zu verändern ohne irgend etwas an den Inhalten ändern zu müssen.

Wie funktioniert das Skins Tool

Layer Definition

eine gültige Skin besteht aus einer Layerdefinition

Was ist ein Layer?

Ein Layer ist eine Liste von verzeichnisartigen Objekten, die wiederum PageTemplates, DTML, Bilder oder Properties enthalten.

Eine Skin namens blue könnte seine Layer als blue, basic_css, general definiert haben. Das bedeutet, daß eine Datei namens standard_html_header zuerst in blue, dann in basic_css und dann in general gesucht wird, wenn das Skin blue aktiv ist.

Es gibt standardmäßig zwei Möglichkeiten, diese Objekte in icoya OpenContent zu integrieren.

Dateisystem-basierend

Dies sind spezielle Objekte, die eine Ansicht auf Inhalte des Dateisystems repräsentieren. Das wichtigste hierbei ist, daß diese Objekte vom Web Interface der Manager Oberfläche aus nicht verändert werden können. Solche Objekte werden von einem Zope Produkt zur Verfügung gestellt. Sie werden in Zope als gesperrte (*"locked"*) Objekte dargestellt. Soll ein solches Objekt in Zope verändert werden, so muss eine Kopie davon in einem Custom Folder (s.u.) erstellt werden.

Custom Folders

Diese Folder sind normale Zope Folder Objekte, welche beliebige Unterobjekte enthalten dürfen.

Wie fange ich an?

Das Skin Tool befindet sich direkt unterhalb Ihres icoya OpenContent Ordners im Zope Manager (*"url/manage"*). Nach dem Klick auf `portal_skins` sieht man eine Liste aller verfügbaren Layer (s.o.). Standardmäßig gibt es nur einen Custom Folder namens custom.

Indem Sie auf das Properties Tab des `portal_skins` Objektes klicken, sehen Sie die aktuellen Skin-Definitionen. In der Layers Spalte befindet sich eine durch Komma getrennte Liste aus Foldern, welche das entsprechende Skin definieren. Nach der Reihenfolge wie die Verzeichnisse hier eingetragen sind, funktioniert auch die Suche nach den entsprechenden Objekten.

Aus welchen Objekten kann ein Skin bestehen?

Damit Zope auch erkennt, welche Art von Code die Datei auf dem Filesystem enthält, findet eine Zuordnung von Dateiendung zu Objekttyp statt.

Die wichtigsten Zuordnungen sind:

- 📄 .pt -> PageTemplate
- 📄 .html -> DTML Document
- 📄 .dtml -> DTML Document
- 📄 .py -> Python Script
- 📄 .gif -> Image
- 📄 .png -> Image
- 📄 .prop -> Stylesheet Property File

Darüberhinaus kann in den Skin Objekten natürlich auf alle Zope-Features wie Akquisition (*"acquisition"*) u.ä. verfügt werden. Auch Unterordner sind in Skin-Foldern erlaubt.

Der beste und schnellste Weg ein eigenes Skin zu entwickeln ist sicherlich, sich an den vorhandenen Skins zu orientieren und möglichst viele Dinge zu übernehmen. Als Beispiel hierfür dient das demo1 Skin im `lib/python/Products/icoyaCMFSkins/skins` Ordner, welches zusammen mit icoya OpenContent geliefert wird.

Ein neues Skin erstellen

Rein praktisch gesehen müssen Sie einfach im `portal_skins` Ordner weitere Unterordner mit Objekten anlegen und diese neuen Layer in eine Custom Skin Layerdefinition eintragen. Dies könnten Sie entweder komplett in Zope mit Hilfe der Custom Folders (s.o.) realisieren oder gleich auf dem Dateisystem. Hier wird nur auf letzteres eingegangen.

- 📄 Wechseln Sie auf dem Dateisystem in das `lib/python/Products/icoyaCMFSkins/skins` Verzeichnis Ihrer icoya OpenContent Installation.
- 📄 Dieses Verzeichnis ist als Skin Folder in Zope bekannt und wird beim Neustart des icoya OpenContent Servers neu eingelesen.
- 📄 Erstellen Sie ein Verzeichnis mit dem Namen `new_common`.
- 📄 Starten sie den icoya OpenContent Server neu.
- 📄 Wechseln Sie in den Zope manager (*"/manage"*) und öffnen sie das Portal Skins Tool.
- 📄 Legen sie ein neues Objekt vom Typ Filesystem Directory View an und wählen sie das soeben angelegte Verzeichnis `new_common`. Jetzt wird im Skins Tool ein Folder (Layer) mit dem gleichen Namen angelegt. Darin finden Sie in Zope alle Objekte, die Sie auf dem Dateisystem in diesem Folder erstellen.
- 📄 Erstellen sie im Properties Tab eine neue Skin Layerdefinition namens `new`, und definieren Sie die Layer (am besten genau gleich wie das Basic Skin). Nach dem custom Layer fügen sie noch ihren eigenen Layer `new_common` und `icoya_common` ein.
- 📄 Stellen Sie dann noch `new` als Default Skin ein.

- ☞ Wenn Sie nun per Webbrowser auf Ihr icoya OpenContent Projekt zugreifen, werden Sie ihr neues Skin betrachten können. Da sich bislang keine Dateien in Ihrem eigenen neuen Skin Folder befinden, sieht Ihr Skin genau wie das Basic Skin aus.

Die neue Skin anpassen

Die neue Skin wurde angelegt und es funktioniert. Nun fehlt allerdings noch der entscheidende Teil. Die eigenen Änderungen. Zunächst sei gesagt, daß es am sinnvollsten ist, zuerst mit einem HTML Editor Ihrer Wahl das Layout der Webseite komplett statisch aufzusetzen. Wenn das geschehen ist, muß dieses Layout noch mit PageTemplate Code erweitert werden. Dabei kommt der große Vorteil von PageTemplates zum tragen. Die PageTemplate Syntax ist komplett XML konform und sprengt damit nicht das Layout. Das heißt, auch nach dem Einfügen der PageTemplate Tags können Sie die Seite wie gewohnt in ihrem HTML Editor verändern. Die PageTemplate Tags stören nicht und bleiben erhalten.

Es hat sich als praktikabel erwiesen, möglichst viele, in sich abgeschlossene Teile des Layouts wie Header, Footer, JavaScript Block oder CSS nicht in einem einzigen großen PageTemplate zu halten, sondern für jeden Baustein ein eigenes Template zu erstellen. Aus diesen Templates wird dann die komplette Webseite generiert. Verwenden Sie METAL-Makros für die Integration der einzelnen Templates.

Das erste Makro, das jedes icoya OpenContent Objekt verwendet, ist das Metal-Makro `main_template/macros/main`. Ohne dieses PageTemplate `main_template.pt` mit einem definierten Makro `main` ist jedes Skin nutzlos. Dieses Makro benötigt mindestens einen Fill-Slot mit dem Namen `main`. In diesen Slot wird der Inhalt eines Content Objektes gerendert. Zum Rendern dieses Inhaltes liefert jedes Content Objekt eine eigene Methode (meist ein PageTemplate) mit. Damit dieses Objekt auch gefunden werden kann, muß das entsprechende Skin-Verzeichnis des Objektes ebenfalls in Ihre Skin-Layer Definition aufgenommen werden. Wir empfehlen Ihnen bei einer neuen Skin immer von einer vorhandenen Skin auszugehen; das heißt, zumindest die Layerdefinition des Basic Skins und den `icoya_common` Layer in Ihre neue Definition mit aufzunehmen.

Schritt für Schritt zum eigenen Layout

- ☞ Erstellen Sie das komplette Layout statisch in einer Datei mit einem beliebigen HTML Editor
- ☞ Trennen Sie das Layout in mehrere Einzelblöcke auf (Core, Header, Footer, JavaScript)
- ☞ Benennen Sie die einzelnen Dateien entsprechend Ihrer Funktionen:
 - Core -> `main_template.pt`
 - Header -> `header.pt`
 - Footer -> `footer.pt`
 - JavaScript -> `jsheader.pt`
- ☞ Definieren Sie Makros und Fill Slots in den einzelnen Dateien.
Wichtige Makros:
 - `Main_template.pt/macros/main` mit Fillslot `main`
 - `Header.pt/macros/portal_header`
 - `Footer.pt/macros/portal_footer`
- ☞ Verwenden Sie diese Makros in `main_template.pt` an den entsprechenden Stellen.

- ☞ Für Bilder ist es empfehlenswert ein eigenes Verzeichnis `new_images` anzulegen und dies genau wie den `new_common` Folder, in Ihrer Layerdefinition mit aufzunehmen. Danach können Sie - so als wären die Bilder im gleichen Verzeichnis - darauf verlinken.
Beispiel: `` (ohne Pfadangabe).