django Organice Documentation

Release 0.3

Peter Bittner <django@bittner.it>

16.01.2016

Die Kollaborationslösung, die alles kann. Für Non-Profit-Organisationen, Sportvereine, kleine bis mittlere Unternehmen.

Mit django Organice gehen Sie den angenehmen Weg, eine Kollaborationsplatform, ein Intranet und Websites für Ihr Geschäft zu betreiben. Es ist eine Zusammenstellung der besten Django-Pakete, vorkonfiguriert, damit Sie sofort durchstarten können. Mit dem bekannten Django Web framework unter der Motorhaube treffen Sie eine sichere Wahl für Ihr IT-Investment, das Ihnen eine leicht erweiterbare Architektur bietet.

Auch in der Cloud verfügbar. Nice, very nice, http://organice.io

KAPITEL 1

Inhalt

1.1 django Organice

Die Kollaborationslösung, die alles kann. Für Non-Profit-Organisationen, Sportvereine, kleine bis mittlere Unternehmen.

Mit django Organice gehen Sie den angenehmen Weg, eine Kollaborationsplatform, ein Intranet und Websites für Ihr Geschäft zu betreiben. Es ist eine Zusammenstellung der besten Django-Pakete, vorkonfiguriert, damit Sie sofort durchstarten können. Mit dem bekannten Django Web framework unter der Motorhaube haben Sie sich für eine sichere Wahl in Ihrem IT-Investment entschieden, das über eine leicht erweiterbare Architektur verfügt.

1.1.1 Grundsätzliche Features

- Befähigt jeden, zu Ihren Inhalten etwas beizutragen – während Sie die vollständige Kontrolle bewahren.
- Unterstützt alle vier gängigen Kollaborations- und Support-Muster auf mehrere Wege:
  1. *Aktive Mitarbeit* (Inhalte erstellen, Korrekturen beisteuern)
  2. *Passive Mitarbeit* (Probleme melden, Aufgaben zuweisen, Feedback geben)
  3. *Aktiver Konsum* (Suchen und Durchstöbern von Inhalten)
  4. *Passiver Konsum* (Abonnieren von Benachrichtigungen)
- Involviert nachhaltig, hilft und unterstützt Sie bei der Zusammenarbeit optimal durch Aussenden von Erinnerungsmails, Bearbeitungs- und Einreichungshinweisen, damit Ihre Benutzer nicht Motivation und Interesse verlieren.

1.1.2 Kernkomponenten

Wesentliche Komponenten von django Organice sind:

- Django Web Framework
- leistungsstarkes Content Management (django CMS)
- leistungsstarker, vielseitiger Blog (Zinnia, cmsplugin)
- Veranstaltungskalender (via Zinnia)

- flexibles Kontaktformular (cmsplugin)
- Seiten für Linksammlungen
- leistungsstarker Newsletter (Emencia)
- Todo-Listen, Aufgabenlisten
- Social Login und Benutzerprofile (allauth)
- Generischer Analytics-Support (analytical)
- Web Analytics, die Ihre Privatsphäre respektiert (Piwik-Integration)
- Zahlreiche Web-Themes (kostenlos verfügbar unter http://organice.io/themes)
- Mehrsprachenfähigkeit
- Multi-Websitefähigkeit
- Datenbankunterstützung für PostgreSQL, MySQL, Oracle u.v.m.

1.1.3 Roadmap

Wir können es kaum erwarten, Ihnen diese Features zu liefern:

- Gruppenfunktionalität mit automatischem Mailverteiler (groupname@example.com)
- Dokumentationsbereich / Wiki (möglicherweise mittels CMS-Funktionalität)
- Diskussionsforum / Q&A-Board (Askbot, Misago)
- Herunterladen von Inhalten zum Generieren von Print-Magazinen
- unaufdringliches, integriertes, transparentes Dokumenten- und Asset-Management
- DropBox, ownCloud, Windows OneDrive, Google Drive, Apple iCloud, etc.-Integration
- voll integrierte Suche (Haystack)
- Live-Chat oder Chat-Integration
- Umfragen (django-crowdsourcing, ntusurvey)

1.1.4 Wer benutzt django Organice?

Beispiele von Websites, die auf django Organice laufen:

- Organice.io Website
- Organice Demo-Website
- Rudolf Steiner Schule Kreuzlingen

1.1.5 Downloads und Mitarbeit

Offizielle Repositories: (synchronisiert)

1. Bitbucket: https://bitbucket.org/organice/django-organice
2. GitHub: https://github.com/organice/django-organice

1.1.6 Hilfe holen

- Dokumentation ist verfügbar unter http://docs.organice.io
- Fragen? Bitte verwenden Sie StackOverflow. Markieren Sie Ihre Fragen mit `django-organice`.
- Haben Sie einen Bug gefunden? Verwenden Sie bitte entweder den Bitbucket oder GitHub Issue Tracker (freie Wahl)
- Brauchen Sie Unterstützung? Unser Gitter chat room steht Ihnen gerne zur Verfügung.

1.2 Installation

Dieses Dokument setzt voraus, dass Sie grundsätzlich mit der Entwicklung und den tools von Python und Django vertraut sind. Falls nein, lesen Sie bitte zuerst pip, virtualenv und virtualenvwrapper. Ein einfaches Verständnis davon ist ausreichend.

1.2.1 Voraussetzungen

- Python 2.7, 3.3 oder 3.4

Alle anderen Abhängigkeiten werden durch den django Organice Installer aufgelöst. Die meisten dieser Abängigkeiten sind bewusst nicht auf ihre Versionsnummer gepinned, um einen liberalen Upgrade-Pfad zu erlauben. Bestätigte, funktionierende Abhängigkeiten sind im CHANGELOG für jedes Release dokumentiert.

Für die Installation empfohlen

- pip
- virtualenv
- virtualenvwrapper

1.2.2 django Organice installieren

1. Wir empfehlen, ein Virtual Environment für den Betrieb von django Organice anzulegen:

```
$ mkvirtualenv example
$ workon example
```

   Die Eingabeaufforderung verändert sich in etwas wie `(example)~$`, um anzuzeigen, dass Ihr neues Virtual Environment aktiv ist.

2. Am einfachsten klappt die Installation mit `pip`:

```
$ pip install django-organice
```

   Damit wird das neueste django Organice-Paket aus dem Internet heruntergeladen und alle Abhängigkeiten automatisch installiert.

Wenn Sie Entwickler sind, möchten Sie vielleicht den letzten Stand von django Organice als Quellcode ziehen: (tun Sie das nicht als einfacher Benutzer)

```
$ git clone https://github.com/Organice/django-organice.git
$ cd django-organice
$ python setup.py install
```

oder alternativ mit pip:

```
$ pip install git+https://github.com/Organice/django-organice.git#egg=django-organi
```

3. Installieren Sie den Datenbankadapter, der zu Ihrer Datenbank passt (PostgreSQL `psycopg2`, MySQL `MySQL-python`, Oracle `cx_Oracle`, etc.), z.B.

```
$ pip install psycopg2
```

Das Django-Projekt empfiehlt PostgreSQL.

Bemerkung: Sie können diesen Schritt überspringen, wenn Sie SQLite verwenden möchten, z.B. zur Evaluierung von django Organice.

---

4. Führen Sie den Organice Setup-Befehl zum Erstellen Ihres neuen Projektes aus: (z.B. *example*)

```
$ organice-setup example
```

5. Bearbeiten Sie Ihre settings in `example/settings/common.py`, `example/settings/develop.py`, etc. Ziehen Sie die Django documentation zu Rate, wenn Sie mit bestimmten settings nicht vertraut sind. Die `develop` settings werden von Ihrem Projekt standardmäßig benutzt (lokale Entwicklung), `common` wird in alle Profile eingebunden.

6. Initialisieren der Datenbank:

```
$ python manage.py organice bootstrap
```

Damit wird Ihre Datenbank vorbereitet und ein paar Beispiel-Inhalte eingepflegt. Wenn Sie bevorzugen, mit einer sauberen Datenbank zu starten, führen Sie stattdessen nur `migrate`:

```
$ python manage.py migrate
```

7. Ihr Django-Projekt starten:

```
$ python manage.py runserver
```

Sie können mit Ihrem Web-Browser jetzt die Adresse http://127.0.0.1:8000/ öffnen und beginnen, Ihr Projekt lokal zu entwickeln.

Bemerkung: Wenn Sie vorhaben, Ihre Inhalte lokal zu erstellen, stellen Sie sicher, dass Sie die gleiche Datenbank-Engine für lokalen Entwicklung und Produktion verwenden. Ihr Plan, den gesamten Datenbankinhalt von der Entwicklungsumgebung in die Produktionsumgebung zu bewegen wird Ihnen sonst größere Kopfschmerzen bereiten. Und, verwenden Sie Sqlite nur zum Evaluieren!

1.2.3 Konfiguration zu Beginn

1. Folgen Sie den Anweisungen des django Organice Installers `organice-setup`, sobald das Setup abgeschlossen ist. Sie müssen ein paar Werte in Ihren Projekt-settings anpassen!
2. Wenn Sie auf Ihrer Website eine andere Sprache als Englisch verwenden möchten oder wenn Sie mehrere Sprachen verwenden wollen: Passen Sie die Werte von `LANGUAGE_CODE` und `LANGUAGES` an und setzen Sie `USE_I18N = True` in Ihren Projekt-settings.
3. Nach der Installation ist django Organice konfiguriert, wenn Sie aber nicht den `bootstrap` Management Command ausgeführt haben, ist die Datenbank ohne Inhalte für Ihre Website. Sie können etwas an Beispiel-Inhalten und andere Daten hinzufügen, indem Sie einen oder alle der folgenden Befehle ausführen:

```
$ python manage.py organice initauth  # prepare social auth provider configuration
$ python manage.py organice initcms   # add pages for your website
$ python manage.py organice initblog  # add blog categories and posts
```

4. Alternativ legen Sie Ihre ersten Seiten, Blog-Einträge und Newsletter-Daten händisch an:
   - Fügen Sie einige Seiten in der Django-Administration unter Cms > Pages hinzu und veröffentlichen Sie Ihre Änderungen.
   - Gehen Sie auf Ihre neue Website und befüllen Sie Ihre neuen Seiten mit Inhalt, indem Sie das Front-end Editing-Feature verwenden.
   - Gehen Sie zu `/blog/` auf Ihrer Website und legen Sie Blog-Einträge an.
   - Fügen Sie in der Django Administration unter Newsletter > Contacts einen Benutzer hinzu.
   - Fügen Sie `localhost` (oder geeigneten Server) zu Newsletter > SMTP servers hinzu.
   - Um das Subscribe-Feature auf der Website zu aktivieren (von `/newsletter/subscribe`), fügen Sie eine Liste zu Newsletter > Mailing lists hinzu.
   - Erstellen Sie nun Ihren ersten Newsletter unter Newsletter > Newsletters.
   - Um Vorlagen zu Emencia Newsletter hinzuzufügen, ziehen Sie bitte den diesbezüglichen Abschnitt in der TinyMCE 3.x documentation zu Rate.
5. Damit das Versenden von Newslettern funktioniert, müssen Sie einen cronjob konfigurieren, der auf `python manage.py send_newsletter` jede halbe Stunde pollt. Wenn das für Sie gerade Spanisch war, fragen Sie bitte Ihren Server Admin um Hilfe. Sie weiß garantiert Bescheid!

1.2.4 Deployment in Produktion

Während der Installation hat `organice-setup` 3 unterschiedliche Umgebungen vorbereitet, die Ihnen beim Deployment helfen:

```
example
-- settings
|   -- __init__.py
|   -- common.py
|   -- develop.py
|   -- staging.py
|   -- production.py
```

Dieses modularisierte Setup ist in Lösung 2 von Tommy Jarnac's Blog-Eintrag zu Django settings best practices [1] beschrieben. Die `develop` settings sind standardmäßig aktiv (für lokale Entwicklung), `common` wird in alle Profile eingebunden.

Für ein Deployment in alle Umgebungen ungleich `develop` muss das verwendete settings-Modul durch das Setzen der Django Umgebungsvariablen `DJANGO_SETTINGS_MODULE` festgelegt werden. Zum Beispiel, wenn Sie Apache als Ihren Django Webserver verwenden, passen Sie Ihre Apache-Konfiguration für `example` an mit:

```
SetEnv DJANGO_SETTINGS_MODULE example.settings.production
```

Bemerkung: Um unterschiedliche settings lokal zu testen, können Sie den Django Webserver mit der `--settings` Option starten:

```
$ python manage.py runserver --settings=example.settings.staging
```

Zu guter Letzt, stellen Sie sicher, dass Sie die deployment checklist des Django-Projekts konsultiert haben und den beschriebenen Best Practices folgen.

1.3 Konfiguration

django Organice kommt mit vernünftigen Vorgabewerten für so gut wie Alles. Darüber hinaus können Sie sein Verhalten über Django settings anpassen, die hier aufgelistet sind.

1.3.1 Settings

Sie können jede der folgenden Optionen in den `settings` Ihres Projekts definieren, um den Vorgabewert zu übersteuern.

ORGANICE_URL_PATH_ADMIN

Default `admin`

Der URL-Pfad zum Erreichen der Django Administrationsoberfläche (z.B. `www.example.com/admin`). Muss nicht-leer sein. Benutzen Sie einen durchgehenden Bezeichner, kein Abstand, kein Schrägstrich am Anfang oder Ende.

ORGANICE_URL_PATH_BLOG

Default `blog`

Der URL-Pfad zum Erreichen der Blog-Startseite (z.B. `www.example.com/blog`). Muss nicht-leer sein. Benutzen Sie einen durchgehenden Bezeichner, kein Abstand, kein Schrägstrich am Anfang oder Ende.

[1] David Cramer von DISQUS hat eine ähnliche Lösung unter http://justcramer.com/2011/01/13/settings-in-django/ beschrieben

ORGANICE_URL_PATH_NEWSLETTER

Default `newsletter`

Der URL-Pfad zum Erreichen der Newsletter-Funktionalität am Front-end (z.B. `www.example.com/newsletter`). Muss nicht-leer sein. Benutzen Sie einen durchgehenden Bezeichner, kein Abstand, kein Schrägstrich am Anfang oder Ende.

ORGANICE_URL_PATH_TODO

Default `todo`

Der URL-Pfad zum Erreichen der Todo-Listen-Funktionalität am Front-end (z.B. `www.example.com/todo`). Muss nicht-leer sein. Benutzen Sie einen durchgehenden Bezeichner, kein Abstand, kein Schrägstrich am Anfang oder Ende.

1.3.2 Settings von Dritt-Apps

Analytics-Provider

Die von `django-analytical` unterstützten Analytics Services werden durch das Setzen unterschiedlicher Service-Eigenschaften in Ihrer settings-Datei aktiviert. Die Eigenschaften sind in their documentation dokumentiert. Aus Sicherheitsgründen sollten Sie diese nicht zu Ihrer common Settings-Datei hinzufügen, sondern zu `settings.production`.

1.4 Themes

One of the nice things of django Organice is that themes are handled as separate projects. In fact, they are pluggable Django apps composed of assets and templates that you can simply install and activate.

1.4.1 Official Themes

Here is a list of django Organice themes officially supported by us:

1. RSSK Theme: django-organice-theme-rssk
2. Fullpage Theme: django-organice-theme-fullpage

If you have a nice theme and would like to include it in this list let us know by e-mail or make a pull request on this page of the documentation.

Mother Theme

`django-organice-theme` is the mother of all themes for django Organice. This theme is installed automatically when you install django Organice. From the development perspective all themes are derived from the mother theme, which contains a collection of static files (assets) and templates, as well as a `Makefile` for asset management. The mother theme is composed of:

- bootstrap-sass (Sass version of Twitter Bootstrap v3)
- Compass (CSS authoring framework using Sass)

- UglifyJS v2 (JavaScript minifier)

The `Makefile` also supports you with updating those components on your development system.

1.4.2 Rolling Your Own Theme

Preparations:

- Visit http://organice.io/themes and find a theme that is as close as it gets of what you want.
- Go to that theme's repository page, make a copy of the whole project, and rename it (e.g. to `mytheme`).

Loop until you're happy:

- Add or adapt the style sheet (`.scss`), JavaScript (`.js`), and other files in `mytheme/static/`.
- Run `make assets` in order to compile the Sass files to CSS, and combine and minify both CSS and JavaScript.
- Adapt the template files in `mytheme/templates/`, and test the results on your development system.

1.5 Benutzerhandbuch

django Organice besteht aus den folgenden Hauptkomponenten:

1. *Content Management* (Cms)
2. *Blog* (Zinnia)
3. *Newsletter*

1.5.1 Content Management

Das Bearbeiten Ihrer Website unterscheidet sich nicht wesentlich vom Surfen im Internet. Wenn Sie auf Ihrer Website eingeloggt sind, haben Sie am oberen Rand der Seite die CMS-Werkzeugleiste zur Verfügung. Klicken Sie auf die Schaltfläche "Bearbeiten", um in den Bearbeiten-Modus zu wechseln. Wenn Sie nun über die Website-Elemente fahren, bemerken Sie Tooltips ("Doppelklicken zum Bearbeiten"). Doppelklicken Sie auf diesen Bereichen und ein Editor-Fenster öffnet sich, um den Inhalt zu bearbeiten.

Sie können seelenruhig Inhalte der Seite verändern, ohne dass sich die Website im Internet verändert. Nur Sie können die Änderungen sehen, und nur so lange Sie im Bearbeitungsmodus sind. Wenn Sie mit den Änderungen glücklich sind, drücken Sie auf die blaue "Änderungen veröffentlichen"-Schaltfläche auf der django CMS Werkzeugleiste und Ihre Version wird auch für andere sichtbar.

Für technischere Aufgaben, wie z.B. das Erstellen einer Navigationsstruktur, das Hinzufügen von Seiten, usw., leitet Sie die django CMS Werkzeugleiste auf die Django Administrationsoberfläche weiter. Brauchen Sie ein paar Eindrücke vorab, bevor Sie das System zum ersten Mal benutzen? Sehen Sie sich das (etwas in die Jahre gekommene) django CMS video on frontend-editing an.

1.5.2 Blog

Interessante Artikel hie und da zu schreiben und zu veröffentlichen nennt man "Blogging". Es ist dem üblichen Bearbeiten der Website sehr ähnlich, dennoch unterscheidet es sich aufgrund der zahlreichen Features, die man quasi automatisch dazubekommt: Kategorien, Etiketten, Kommentare, Veröffentlichungsdatum, Archive (jährlich, monatlich, täglich), verwandte Einträge, RSS-Feeds, usw.

Auf Ihrer Website gehen Sie einfach auf den Blog-Bereich und klicken dann auf die "Hinzufügen"-Schaltfläche, die verfügbar ist, wenn Sie eingeloggt sind. Im Blog-Eintrag-Editor haben Sie die gesamte Funktionsvielfalt zur Verfügung, die Sie bereits aus dem Content Management-Bereich kennen. Bestehende Blog-Einträge bearbeiten funktioniert genauso wie das übliche Überarbeiten Ihrer Website mit der Ausnahme des Veröffentlichens, mit dem ansonsten ein Entwurf als Zwischenschritt zur Verfügung steht: Alle gespeicherten Änderungen sind sofort online sichtbar.

Merke: In django Organice wird die Blog-Funktionalität auch für Anwendungsfälle genutzt, die dem üblichen Bloggen sehr ähnlich sind, z.B. Veranstaltungskalender, Stellenausschreibungen, Pressemeldungen, Download-Listen, etc.

1.5.3 Newsletter

Für das Aussenden von Neuigkeiten und Updates an einen Freundes- oder Kundenkreis steht die Newsletter-Komponente zur Verfügung, die in der Django Administrationsoberfläche bedient werden kann, wenn Sie auf der Website eingeloggt sind. Kicken Sie einfach auf "Admin > Seiten-Administration" in der django CMS Werkzeugleiste und navigieren Sie zum Newsletter-Bereich im Django-Admin.

Diese Komponente hat einige sehr mächtige Features. Am besten kann diese das interessante französische overview video auf der Emencia-Website erklären.

1.6 Bei django Organice mitarbeiten

Offizielle Repositories: (synchronisiert)

1. Bitbucket: https://bitbucket.org/organice/django-organice
2. GitHub: https://github.com/Organice/django-organice

Forken Sie eines der Repositories, machen Sie Ihre Änderungen oder fügen Sie neuen Code hinzu, und eröffnen Sie einen Pull Request.

1.6.1 Wie fange ich an?

Nach dem Klonen Ihres eigenen Forks von Bitbucket oder GitHub stellen Sie sicher, dass Sie ein Virtual Environment für die Entwicklung erstellt und aktiviert haben, dann führen Sie `make develop` aus, um die Packages zu installieren, die Ihnen bei der Entwicklung helfen (Testwerkzeuge, Übersetzungstools, Dokumentationsgeneratoren).

Richtlinien

Der primäre Ziel-Interpreter für die Entwicklung ist **Python 3**. Am besten verwenden Sie die höchste Versionsnummer, die mit dem in das Projekt integrierten Django package kompatibel ist (momentan

3.4). Alle anderen unterstützten Python-Versionen werden vom integration server getestet, sobald Sie einen Pull Request eröffnen. Sie können Tests mittels `tox` oder `setup.py test` lokal ausführen, bevor Sie pushen, z.B.

```
$ tox                # run all tests against all supported Python versions
$ tox -e py34,py27   # run all tests against Python 3.4 and 2.7 only
$ ./setup.py -q test -a tests/management  # only run management tests against default py
```

Der Quellcode soll die `flake8` Standardregeln erfüllen (mit Ausnahme der Zeilenlänge, die bis zu 120 Zeichen betragen darf). Ein pre-commit hook für Git wird der Einfachheit halber automatisch installiert, wenn Sie `make develop` ausführen, somit sollte es nicht einmal möglich sein zu committen, wenn `flake8` nicht erfolgreich durchläuft. Zusätzliche statische Analyse wird am QA server durchgeführt und Sie sollten sicherstellen, dass dort der Gesundheitsindikator für den Quellcode mit jedem Beitrag nach oben geht (oder gleichbleibt).

1.6.2 Unterstützung gesucht

- Schreiben von Tests (Unit Tests, BDD-Tests)
- Neue Features auf der Roadmap (siehe README)
- Übersetzungen (Blog-Einträge, Dokumentation, Benutzeroberfläche)

Übersetzung

Die Übersetzung wird sowohl für die Texte im Quellcode als auch die Dokumentation mit Transifex durchgeführt.

Die mehrsprachige Dokumentation ist in reStructuredText Syntax geschrieben und wird mit Sphinx gebaut. Als Übersetzer können Sie einfach auf Transifex gehen und anfangen, sich für Ihre Sprache ins Zeug zu legen. Einige hilfreiche Hintergrundinformationen gibt es auf Read the Docs und im Transifex support zu lesen.

1.6.3 Versionen erstellen, Packaging, Releases

Hinsichtlich der Versionsnummern von django Organice verwenden wir Semantic Versioning. Änderungen von einem zum nächsten Release werden im CHANGELOG dokumentiert.

1.7 Alternativen zu django Organice

Mit django Organice versuchen wir Menschen glücklich zu machen, die keine Technik- oder Prozessexperten sind. Menschen, die einfach und unkompliziert zusammenarbeiten wollen, ohne vorher an umfangreichen Schulungen teilgenommen zu haben, nur um das Arbeitsinstrument zu verstehen. Einfachheit ist der Schlüssel, alles soll selbsterklärend sein, damit die Arbeit leicht von der Hand geht. Dabei sollen gelegentliche Nutzer genauso glücklich werden wie Experten, die die Plattform täglich nutzen.

Natürlich kann django Organice nicht für alles die Lösung sein. Wenn Sie das Gefühl haben, Ihre speziellen Bedürfnisse kann es nicht befriedigen, könnte es sich auszahlen, einen Blick auf die uns bekannten Alternativen zu werfen. Falls Sie weitere Angebote kennen, drop us a note, damit wir diese Liste sinnvoll ergänzen können.

1.7.1 Software-Alternativen

In alphabetischer Reihenfolge.

- Bitrix Intranet (PHP, kommerziell, wird mit Quellcode geliefert)
- Coyo (Java/Play, kommerziell, wird mit Quellcode geliefert)
- Fairgate (PHP, kommerziell, nur in der Schweiz)
- Microsoft SharePoint + Yammer [2] (.NET + Rails, kommerziell)
- Odoo (Python, Open Source)
- Open Atrium (Drupal, Open Source)
- teamraum Enterprise (Django, kommerziell)
- tendenci (Django, Open Source)

1.7.2 Cloud-Mitbewerber

In alphabetischer Reihenfolge.

- Bitrix24.com / Bitrix24.de [1]
- huddle
- Igloo
- Odoo
- teamraum

1.8 CHANGELOG

1.8.1 0.3

- Mediatree-Komponente deaktiviert (muss auf Python 3 upgegraded werden) – *geplant für v0.4*
- Newsletter-Komponente deaktiviert (muss auf Python 3 upgegraded werden) – *geplant für v0.4*
- Python 3 Unterstützung hinzugefügt
- Umstellung von develop/master Modell auf feature-Branching Modell (Standard-Branch: `master`)
- Migration des Projekts auf Django 1.8.8 und django CMS 3.2
- Beispiel-Inhalte zum Laden hinzugefügt (zuerst via Fixtures, später umgestellt auf Management Commands)
- Integration von Piwik open source analytics (optional mit django-analytical)
- Testabdeckung bereitgestellt, endlich!
- Start von Continuous Builds (Travis-CI, Shippable)

[2] http://intranetsdoneright.blogspot.com/2013/11/what-is-social-intranet.html

Changes https://github.com/Organice/django-organice/compare/v0.2...v0.3

Dependencies https://github.com/Organice/django-organice/blob/v0.3/requirements.txt

1.8.2 0.2

- Makefile auf Projektebene
- Automatisierung des Übersetzungsprozesses mit Transifex (nur für die Dokumentation)
- Neue Einstellungsoptionen (`ORGANICE_URL_PATH_...`)
- Newsletter Editor-Konfiguration, Newsletter Vorlage-Beispiel
- Social Login und Benutzerprofile hinzugefügt (django-allauth)
- Web Assets Pipeline hinzugefügt (bootstrap-sass, Compass, UglifyJS v2)
- jQuery auf v1.11.0 aktualisiert, Vorlagen mit Bootstrap neu überholt
- Sprachauswahl Dropdown-Menü hinzugefügt
- Daten von Website-Themes (Vorlagen, Styles und JavaScript) und Web Assets Pipeline in separate Projekte ausgelagert
- Generieren der Serverkonfiguration (lighttpd) und weiterer Optionen in organice-setup
- Medien-Management hinzugefügt (django-media-tree)
- Todo-Listen hinzugefügt (django-todo)
- Generische Analytics-Integration hinzugefügt (django-analytical)

  Changes https://github.com/Organice/django-organice/compare/v0.1...v0.2

  Dependencies https://github.com/Organice/django-organice/blob/v0.2/docs/requirements.txt

1.8.3 0.1

- Erstausgabe
- Basierend auf Django 1.5.5, django CMS 2.4.3, Zinnia blog 0.13, Emencia newsletter 0.3.dev
- Natürlicherer i18n-Mechanismus als der in Django eingebaute, ohne Sprachen-Präfix für die Standardsprache
- Setup-Skript mit Projektgenerierung, Deployment-Einstellungen, individuelle Vorlagen, Bootstrap 3

  Dependencies https://github.com/Organice/django-organice/blob/v0.1/docs/requirements.txt

KAPITEL 2

Verzeichnisse und Tabellen

- genindex
- search

Symbols

A

B

C

D

G

I

N

R

T

V