

機能開発ハンズオン

2016/12/14 株式会社ウェブチップス

講師: 中野

ハンズオン資料

- 今回のハンズオンで使用するリポジトリ
 - <https://github.com/shirasagi/ss-handson>
- 今回のハンズオンで使用するコマンド集
 - <https://raw.githubusercontent.com/shirasagi/ss-handson/master/Commandlist.md>
 - <https://github.com/shirasagi/ss-handson/blob/master/Commandlist.md>
- シラサギ開発環境を納めたVagrant Box
 - <https://github.com/shirasagi/ss-vagrant>

目次

1. 開発環境の準備

- シラサギのインストール、デモデータの登録ができるようになります

2. シラサギのプログラム構成について

- 他のRailsアプリケーションと違う、特徴的な構成の箇所を紹介

3. ハンズオン1: 既存アドオンの拡張

- 実際にコードを書きながら連絡先アドオンを拡張してみます。

4. ハンズオン2: 新しいアドオンの作成

- 実際にコードを書きながらお天気アドオンを作成してみます。

5. ハンズオン3: Herokuへデプロイ

- ここまでで開発した機能拡張をHerokuへデプロイしてみます。

中級

6. ハンズオン4: 一覧の拡張

- ハンズオン2で開発した天気を一覧に表示してみます。

7. ハンズオン5: 検索フォルダーの開発

- ハンズオン2で開発した天気を検索できるようにしてみます。

8. ハンズオン6: 検索パーツの開発

- ハンズオン5で開発した検索フォルダー用のパーツを作成し、検索フォームのレイアウトに組み込まれるようにしてみます。

9. ハンズオン7: ページの開発

- 標準アドオンや独自に開発したアドオンを組み合わせることでページを作成してみましよう。

開発環境の準備

Vagrant Box

- シラサギ開発環境を納めたVagrant Box
 - <https://github.com/shirasagi/ss-vagrant>

コンソール

sshなどで開発環境へ接続しターミナルを開いてください。
開発環境内のシラサギが起動している場合、停止します。

```
$ cd $HOME/shirasagi  
$ rake unicorn:stop
```

注意: 先頭の\$は入力しません。

ソースコード取得

GitHubからソースコードを取得します。

```
$ cd $HOME  
$ git clone https://github.com/shirasagi/ss-handson sample  
$ cd sample
```


設定ファイル

必要な設定ファイル(各種ymlとunicorn.rb)をコピーします。

```
$ cp -n config/samples/*.{yml,rb} config/
```

ハンズオン用DB

ハンズオン用DBを設定します。

```
$ vi config/mongoid.yml
```

```
# mongod configuration
production:
  clients:
    default:
      database: ss_sample
      hosts:
        - localhost:27017

development:
  clients:
    default:
      database: ss_sample
      hosts:
        - localhost:27017
```

Railsを開発モードへ変更

Railsを開発モードへ変更します。

```
$ cp config/defaults/environment.yml config/  
$ vi config/environment.yml
```

```
# default environment  
RAILS_ENV: development
```

外部依存モジュールのインストール

外部依存モジュールをインストールします。

```
$ bundle install
```

初期データ投入

MongoDBのインデックス作成

```
$ rake db:drop  
$ rake db:create_indexes
```

サイト作成

```
$ rake ss:create_site data='{ name: "サイト名", host: "www", domains:  
"localhost:3000" }'
```

環境にあわせて変更

自治体サンプルデータ投入

```
$ rake db:seed name=demo site=www
```

Unicorn起動

Unicorn起動

```
$ rake unicorn:start
```

ブラウザでhttp://ドメイン:3000/にアクセス

The screenshot shows the homepage of Shirasagi City. At the top, there is a navigation bar with the city logo and name 'シラサギ市'. Below this is a main menu with categories like '暮らし・手続き', '子育て・教育', '健康・福祉', '観光・文化・スポーツ', '産業・仕事', and '市政情報'. The main content area features a large image of pink cherry blossoms. To the right, there is a '注目情報' (Attention Information) section with several news items, including '市内の微小粒子状物質 (PM2.5) の測定データ (速報値) を公開しています。'. Below this, there is an 'お知らせ' (Notice) section with dates and titles of notices, such as 'ふれあいフェスティバル' and '平成26年4月より国民健康保険税率が改正されます'. At the bottom, there are sections for 'くらしのガイド' (Living Guide) and '安心安全情報' (Safety Information).

シラサギのプログラム構成

シラサギで使っている技術

- Ruby
- Ruby on Rails
- Javascript & CoffeeScript
- CSS & SCSS
- Git & GitHub
- MongoDB & Mongoid (データベースドライバー)

シラサギのディレクトリ構成

- 参照: <http://shirasagi.github.io/devel/directories.html>

ページとノード

- ページ
 - 固定ページ、記事ページ、FAQページ、イベントページと様々な種類がある。
 - 定期的にHTMLファイルに書き出される。
 - 設定次第では書き出さないようにすることもできる。
- ノード（フォルダーともいう）
 - ページのコンテナ
 - ノード自体がコンテンツを提供する。
 - 記事/記事リスト: 記事一覧 → 静的なので書き出される
 - カテゴリー/カテゴリーリスト: カテゴリー一覧 → 静的なので書き出される
 - メールフォーム/フォーム: お問い合わせフォーム → 静的なので書き出される
 - 施設/施設検索: 施設検索フォーム → 動的なので書き出されない
- ページとノードは、コンテンツを提供する。
- ページは、静的なコンテンツ（書き出し可能なコンテンツ）を提供する。
- ノードは、一覧や検索などページの集合を扱ったり、動的なコンテンツを提供する。

パーツとレイアウト

- パーツ
 - 共通して利用できる部品
- レイアウト
 - コンテンツの全体的な構成を定義する
 - グローバルナビゲーション、ローカルナビゲーション、サイドバー、フッターなど
 - 複数のパーツを組み合わせることができる。
 - ページやノードに割りあてることで有効になる。
 - 普段はリッチなUI/UXを提供するレイアウトを、災害時には情報表示をメインとした簡素のレイアウトを切り替えることも可能。
- パーツとレイアウトは、コンテンツ作成をアシストする。

記事/記事リストが提供するコンテンツ

全体構成を「記事レイアウト」に定義

パーツ「navi」

パーツ「breadcrumb」

パーツ「sns」

記事/記事リストが提供するコンテンツ

The screenshot shows the Shirasagi City website's article list page. At the top, there is a navigation bar with links for '本文へ', 'ご利用案内', 'ふりがなをつける', '読み上げる', and background/color options. Below this is the city logo and a main navigation menu with categories like '暮らし・手続き', '子育て・教育', '健康・福祉', '観光・文化・スポーツ', '産業・仕事', and '市政情報'. The article list is displayed in a table format with columns for date and title. A '月別ページ一覧' (Monthly Page List) sidebar is on the right, showing a calendar view of articles by month. Social media sharing buttons (Like, Share, Tweet, Bookmark, G+) and a 'CLIP' button are located above the article list. A 'カレンダー' (Calendar) button is at the bottom right of the article list area.

記事	月別ページ一覧
2016年10月28日 ふれあいフェスティバル	2016年12月(0)
2016年10月28日 転居届	2016年11月(0)
2016年10月28日 自動交付機・コンビニ交付サービスについて	2016年10月(22)
2016年10月28日 住民票コードの変更	2016年9月(0)
2016年10月28日 住民票コードとは	2016年8月(0)
2016年10月28日 住所変更の証明書について	2016年7月(0)
2016年10月28日 住民票記載事項証明書様式	2016年6月(0)
2016年10月28日 証明書発行窓口	2016年5月(0)
	2016年4月(0)
	2016年3月(0)
	2016年2月(0)
	2016年1月(0)

パーツ「tool」

パーツ「head」

パーツ「docs/archive/month」

カテゴリー/カテゴリーリストが提供するコンテンツ

The screenshot shows the Shirasagi City website with several callouts pointing to specific features:

- 全体構成を「FAQトップ」に定義** (Define the overall structure as 'FAQ Top') points to the top navigation bar.
- パーツ「navi」** (Part 'navi') points to the main navigation menu.
- パーツ「breadcrumb」** (Part 'breadcrumb') points to the breadcrumb trail 'HOME > よくある質問'.
- パーツ「faq/faq-search/search」** (Part 'faq/faq-search/search') points to the search input field.
- カテゴリー/カテゴリーリストが提供するコンテンツ** (Content provided by category/category list) points to the category list on the right.
- パーツ「tool」** (Part 'tool') points to the utility links at the top right.
- パーツ「head」** (Part 'head') points to the main navigation menu.
- パーツ「faq/category-list」** (Part 'faq/category-list') points to the category list on the right.
- パーツ「links-life」** (Part 'links-life') points to the related links section on the right.

The website content includes:

- Utility links: 本文, ご利用案内, ふりがなをつける, 読み上げる, 背景色 (白, 青, 黒), 文字サイズ (小さく, 標準, 大きく).
- Mobile site link: スマホ・携帯サイト.
- Contact link: お問い合わせ.
- Site map link: サイトマップ.
- Search bar: サイト内検索.
- Main navigation: [暮らし・手続き](#), [子育て・教育](#), [健康・福祉](#), [観光・文化・スポーツ](#), [産業・仕事](#), [市政情報](#).
- Breadcrumb: HOME > よくある質問.
- Search section: よくある質問, キーワード , 検索, リセット.
- Category list (right): **カテゴリー一覧**, [暮らし・手続き](#), [子育て・教育](#), [健康・福祉](#), [観光・文化・スポーツ](#), [産業・仕事](#), [市政情報](#).
- Related links (right): **関連リンク**, [電子申請・届出](#), [申請書ダウンロード](#), [施設予約](#), [水道仕様開始・停止受付](#).

メールフォーム/フォームが提供するコンテンツ

The image shows a screenshot of the Shirasagi City website's contact form. The page layout is as follows:

- Header:** Shirasagi City logo and navigation menu (Home, City Information, etc.).
- Navigation:** A horizontal menu with categories like '暮らし・手続' (Living/Procedures), '子育て・教育' (Child-rearing/Education), '健康・福祉' (Health/Welfare), '観光・文化・スポーツ' (Tourism/Culture/Sports), '産業・仕事' (Industry/Work), and '市政情報' (Municipal Information).
- Breadcrumbs:** 'HOME > 市へのお問い合わせ' (Home > Contact Us).
- Form Title:** '市へのお問い合わせ' (Contact Us).
- Form Fields:** A series of input fields for 'お名前' (Name), '企業・団体名' (Company/Organization Name), 'メールアドレス' (Email Address), '性別' (Gender), '年齢' (Age), 'お問い合わせ区分' (Inquiry Category), and 'お問い合わせ内容' (Inquiry Content).
- Footer:** Contact information for Shirasagi City, including address, phone numbers, and opening hours.

Callouts on the left side of the image identify specific parts of the form:

- パーツ「navi」:** Points to the navigation menu.
- パーツ「breadcrumb」:** Points to the breadcrumb trail.
- メールフォーム/フォームが提供するコンテンツ:** Points to the entire contact form area.

Callouts on the right side of the image describe the overall structure:

- パーツ「tool」:** Points to the search bar and utility links.
- パーツ「head」:** Points to the header area.
- 全体構成を「1カラム」に定義:** A blue callout indicating the page is designed as a single column.

施設/施設検索が提供するコンテンツ

全体構成を「施設ガイド」に定義

パーツ「navi」

パーツ「breadcrumb」

施設/施設検索が提供するコンテンツ

パーツ「tool」

パーツ「head」

パーツ「map-side」

本文

ご利用案内 ふりがなをつける 読み上げる 背景色 白 青 黒 文字サイズ 小さく 標準 大きく

スマホ・携帯サイト お問い合わせ サイトマップ

サイト内検索 検索

シラサギ市

暮らし・手続き 子育て・教育 健康・福祉 観光・文化・スポーツ 産業・仕事 市政情報

HOME > 施設ガイド

施設ガイド

キーワード

キーワード

施設の種類を選択

文化施設 運動施設 小学校 公園・公共施設

施設の用途を選択

遊ぶ 学ぶ 相談する

施設の地域を選択

東区 西区 南区 北区

この条件で検索する 検索条件をリセットする

施設一覧

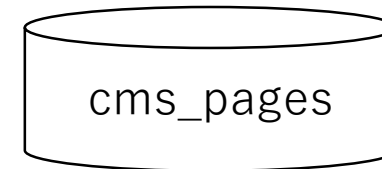
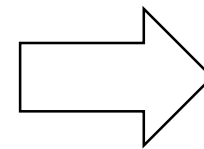
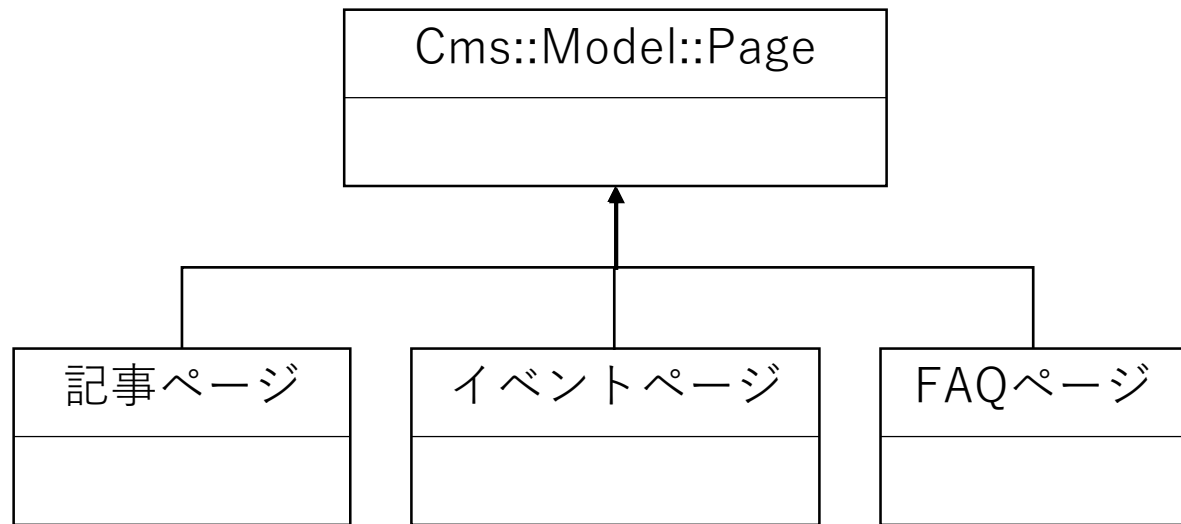
文化施設

運動施設

学校

公共施設

リモートフィック・モデリング



ページは、内部的にはCms::Model::Pageを基底とするクラス構造を持つ

シラサギはドキュメント指向データベースのMongoDBを採用しているので、cms_pagesという単一のコレクションに格納する

ポリモーフィック・モデル

- route属性

- どのレコードがどのモデルに対応するのかを表す属性。
- 全てのページがcms_pagesにごちゃ混ぜになって格納されている。
- route属性で、データベースからレコードを取り出したときに、どの種類のページなのかを判定する。

管理画面と公開画面

- 管理画面
 - コンテンツを構築する画面。
- 公開画面
 - 構築したコンテンツを公開する。
- モデル
 - 管理画面と公開画面で共通
- コントローラー / ビュー
 - 管理画面と公開画面で別々

記事ページのModel / View / Controller

	管理画面	公開画面
Model	app/models/article/page	←
Controller	app/controllers/article/pages_controller	app/controllers/article/agents/pages/page_controller
View	app/views/article/pages/*	app/views/article/agents/pages/page/*

• 管理画面

- 権限のあるページだけが閲覧できる。
- 非公開のページでも権限があれば見える。

• 公開画面

- 権限は無関係。
- 公開されているページのみ閲覧できる。
- 非公開のページは閲覧できない。アクセスしても404。

記事ノードのModel / View / Controller

	管理画面	公開画面
Model	app/models/article/node/page	←
Controller	app/controllers/article/pages_controller	app/controllers/article/agents/nodes/page_controller
View	app/views/article/pages/*	app/views/article/agents/nodes/page/*

- 管理画面

- なんと記事ページと同じ!

- 公開画面

- 権限は無関係。
- 公開されているノードのみ閲覧できる。
- 非公開のノードは閲覧できない。アクセスしても404。

記事リストパーツのModel / View / Controller

	管理画面	公開画面
Model	app/models/article/part/page	←
Controller	n/a	app/controllers/article/agents/parts/page_controller
View	n/a	app/views/article/agents/parts/page/*

• 管理画面

- なんと管理画面のコントローラとビューが存在しない!
- アドオンで管理機能を提供する。

• 公開画面

- 公開されているパーツのみ閲覧できる。
- 非公開のパーツは閲覧できない。アクセスしても404。

アドオン

- 以下をカプセル化
 - DBフィールド
 - データ検証や保存ロジック
 - 編集画面 / 詳細画面 / 公開画面
- ノード、ページ、パーツ、レイアウト、…シラサギ内のすべてのモデルは複数のアドオンが組み合わさっています。

厳密にはモデルアドオンといますが、開発者は単に「アドオン」ということが多いです。

Cms::Addon::BodyのModel / View / Controller

	管理画面	公開画面
Model	app/models/concerns/cms/addon/body	←
Controller	n/a	n/a
View	app/views/cms/agents/addons/body/*	app/views/cms/agents/addons/body/view/index.html.erb

• 管理画面

- アドオンの場合、コントローラーは存在しない
- 編集画面ではapp/views/cms/agents/addons/body/_form.html.erbが、詳細画面ではapp/views/cms/agents/addons/body/_show.html.erbがレンダリングされる

• 公開画面

- アドオンの場合、コントローラーは存在しない
- 公開画面用のviewが存在しないことも多い
- 公開画面用のviewが存在する場合、viewサブディレクトリのindex.html.erbがレンダリングされる

ハンズオン1 既存アドオンの拡張

連絡先アドオンの拡張

- 記事ページに連絡先アドオンが組み込まれています。
- 夜間窓口、申込先、第2電話番号なんでもいいので追加してみましょう。

連絡先

表示設定 ?

所属 ?

グループ名
シラサギ市/企画政策部/政策課 <input type="button" value="削除"/>

担当 ?

電話番号 ?

ファックス番号 ?

メールアドレス ?

連絡先アドオンの検索

- 記事ページに組み込まれているアドオンを見てください。
- `app/models/article/page.rb`を開いてみてください。
- 管理画面のアドオンの順と、`app/models/article/page.rb`のincludeの順は同じです。
- 連絡先アドオンを探してみてください。関連記事アドオンの下あたりにはあるはずです。

フィールドをモデルへ追加

- 連絡先アドオンは、Contact::Addon::Pageです。
- `app/models/concerns/contact/addon/page.rb`を開いてみます。
- 私は「夜間窓口」を追加したいと思いますので、文字列型の `contact_night_window` フィールドを追加します。
- 忘れずに `permit_params` も追加してください。

フィールドをモデルへ追加

```
module Contact::Addon
  module Page
    extend ActiveSupport::Concern
    extend SS::Addon

    included do
      field :contact_state, type: String
      field :contact_charge, type: String
      field :contact_tel, type: String
      field :contact_fax, type: String
      field :contact_email, type: String
      field :contact_night_window, type: String
      belongs_to :contact_group, class_name: "SS::Group"
      permit_params :contact_state, :contact_group_id, :contact_charge
      permit_params :contact_tel, :contact_fax, :contact_email
      permit_params :contact_night_window
    end

    .....
  end
end
```

閑話休題: permit_paramsとは？

- Rails4からStrongParametersという仕組みが追加されました。
- 簡単に言うと、悪意あるユーザーから変なパラメータが送られてきた場合に防御する機能です。
- シラサギでは、モデルにpermit_paramsを書いてやることで、安全にパラメータを受け取れるようにしています。

- 野呂先生の解説記事:

<https://www.transnet.ne.jp/2016/05/18/rails%E5%88%9D%E5%AD%A6%E8%80%85%E3%82%B9%E3%83%88%E3%83%AD%E3%83%B3%E3%82%B0%E3%83%91%E3%83%A9%E3%83%A1%E3%83%BC%E3%82%BF%E3%83%BCcolnr/>

夜間窓口の入力画面の開発

- 夜間窓口を入力できるようにするため、ビューを修正します。
- `app/views/contact/agents/addons/page/_form.html.erb`を開き、テキスト型の入力欄を追加します。

夜間窓口の入力画面の開発

app/views/contact/agents/addons/page/_form.html.erb

```
<dt><%= @model.t :contact_email %><%= @model.tt :contact_email %></dt>
<dd><%= f.text_field :contact_email, value: (params[:action] =~ /new/) ?
@cur_group[:contact_email] : @item.contact_email %></dd>

<dt><%= @model.t :contact_night_window %><%= @model.tt :contact_night_window %></dt>
<dd><%= f.text_field :contact_night_window %></dd>
</dl>
```

夜間窓口の表示画面の開発

- 入力した夜間窓口を表示してみます。
- `app/views/contact/agents/addons/page/_show.html.erb`を開き、夜間窓口の表示を追加します。

夜間窓口の表示画面の開発

app/views/contact/agents/addons/page/_show.html.erb

```
<dt><%= @model.t :contact_email %></dt>
<dd class="contact-email"><%= @item.contact_email %></dd>

<dt><%= @model.t :contact_night_window %></dt>
<dd class="contact-night-window"><%= @item.contact_night_window %></dd>
</dl>
```

確認

- ここまで修正できたら、一旦ブラウザで表示を確認してみましょう。
- 英語で表示されていますが、気にせず、夜間窓口を入力し保存して見ます。
- 夜間窓口が保存されれば成功です。

日本語ロケールの作成

- 日本語化しツールチップも作成してみます。
- `config/locales/contact/ja.yml`を開きます。
- “`contact_`”で検索してみます。

日本語ロケールの作成

config/locales/contact/ja.yml

```
mongoid:
  attributes:
    cms/model/page: &cmspage
    contact_state: 表示設定
    contact_group_id: 所属
    contact_group: 所属
    contact_charge: 担当
    contact_tel: 電話番号
    contact_fax: ファックス番号
    contact_email: メールアドレス
    contact_night_window: 夜間窓口
```

ymlファイルは非常に脆いファイルです。タブは使用せず必ずスペースを使用し、インデントを適切に設定してください。

contact_email:

- 連絡先に表示する部署または担当者のメールアドレスを記入します。

contact_night_window:

- 連絡先に表示する夜間窓口を記入します。

日本語ロケールの確認

- 一旦ブラウザで表示を確認してみましょう。
- 日本語で表示されているか、ツールチップが表示されているかを確認してみます。
- 以上で管理側の対応が終わりました。

公開側の表示画面の開発

- ここからは公開側の表示を実装していきます。
- 記事ページのプレビューを表示させましょう。
- 何の実装していないので、夜間窓口が表示されていません。
- 今から夜間窓口が公開側に表示されるように実装していきます。

公開側の表示画面の開発

- `app/views/contact/agents/addons/page/view/index.html.erb`を開きます。
- ついでに`config/locales/contact/ja.yml`に日本語を追加します。

公開側の表示画面の開発

app/views/contact/agents/addons/page/view/index.html.erb

```
<% if @cur_page.contact_night_window.present? %>
  <dl class="night-charge">
    <dt><%= "#{t("contact.view.contact_night_window")}:" %></dt>
    <dd><%= @cur_page.contact_night_window %></dd>
  </dl>
<% end %>
</footer>
<% end %>
```

config/locales/contact/ja.yml

```
view:
  title: お問い合わせ
  tel: 電話
  fax: Fax
  email: E-Mail
  contact_night_window: 夜間窓口
```


公開側の表示画面の開発

- 先ほど表示させたプレビューをリロードしてみましょう。
- 夜間窓口が表示されれば成功です。

まとめ

- 次のファイルを修正しました。
 - app/models/concerns/contact/addon/page.rb
 - app/views/contact/agents/addons/page/_form.html.erb
 - app/views/contact/agents/addons/page/_show.html.erb
 - app/views/contact/agents/addons/page/view/index.html.erb
 - config/locales/contact/ja.yml
- 次のことを学びました。
 - アドオンのモデルとビューの関係
 - 日本語化
 - ツールチップ
 - Rails4のStrongParameter対応

演習

- 次の課題に取り組んでみてください。
 - 連絡先アドオンに別のフィールドを更に追加し、入力画面、表示画面、公開画面を開発。
 - 別のアドオンに独自のフィールドを追加し、入力画面、表示画面、公開画面を開発。

ハンズオンの成果物

- 本ハンズオンの成果物は<https://github.com/shirasagi/ss-handson>のhandson/contact-group-extブランチにあります。

ハンズオン2 新しいアドオンの作成

天気アドオンの作成

- 先ほどの理解をより確実なものとするため、新しいアドオンを作成してみましょう。
- 天気アドオンを作成してみます。

モデルの作成

- 天気を保持するモデルを作成します。
- `app/models/concerns/cms/addon/weather.rb`を作成します。

モデルの作成

app/models/concerns/cms/addon/weather.rb

```
module Cms::Addon
  module Weather
    extend ActiveSupport::Concern
    extend SS::Addon

    included do
      field :weather, type: String
      permit_params :weather
    end

    def weather_options
      [ ["晴れ", "sunny"], ["曇り", "cloudy"],
        ["雨", "rain"], ["雪", "snow"],
      ]
    end
  end
end
```


アドオンを記事ページへ組み込み

- 天気アドオンを記事ページに組み込みます。
- 連絡先アドオンの上に組み込んでみましょう。

- 記事ページはapp/models/article/page.rbでしたね。
- このファイルをテキストエディタで開きます。

アドオンを記事ページへ組み込み

app/models/article/page.rb

```
class Article::Page
  include Cms::Model::Page
  include Cms::Page::SequencedFilename
  include Cms::Addon::EditLock
  include Workflow::Addon::Branch
  include Workflow::Addon::Approver
  include Cms::Addon::Meta
  include Gravatar::Addon::Gravatar
  include Cms::Addon::Body
  include Cms::Addon::BodyPart
  include Cms::Addon::File
  include Category::Addon::Category
  include Cms::Addon::ParentCrumb
  include Event::Addon::Date
  include Map::Addon::Page
  include Cms::Addon::RelatedPage
  include Cms::Addon::Weather
  include Contact::Addon::Page
end
```

入力画面、表示画面、公開画面の作成

- 入力画面
 - `app/views/cms/agents/addons/weather/_form.html.erb`
- 表示画面
 - `app/views/cms/agents/addons/weather/_show.html.erb`
- 公開画面
 - `app/views/cms/agents/addons/weather/view/index.html.erb`

- これらを一気に作っていきます。

入力画面と表示画面の作成

app/views/cms/agents/addons/weather/_form.html.erb

```
<dl class="see mod-cms-weather">  
  <dt><%= @model.t :weather %><%= @model.tt :weather %></dt>  
  <dd><%= f.select :weather, @item.weather_options, include_blank: true %></dd>  
</dl>
```

app/views/cms/agents/addons/weather/_show.html.erb

```
<dl class="see mod-cms-weather">  
  <dt><%= @model.t :weather %></dt>  
  <dd><%= @item.label :weather %></dd>  
</dl>
```

公開画面の作成

app/views/cms/agents/addons/weather/view/index.html.erb

```
<% if @cur_page.weather.present? %>
  <span class="weather <%= @cur_page.weather %>">
    <%= @cur_page.label :weather %>
  </span>
<% end %>
```

閑話休題: labelメソッド

- オプションから選択するというのは頻出イディオムで、シラサギではこれをサポートする機能があります。
- モデル側に、コードとその名称のペアの配列を返す `xxxx_options` というメソッドを定義しておけば、ビュー側で `item.label :xxxx` や `@cur_page.label :xxxx` で、ラベルの名称を簡単に表示できます。

アドオンの確認

- ここまでできたらブラウザで表示を確認してみましょう。

日本語化

- 一部、英語で表示されているので日本語化します。
- `config/locales/cms/ja.yml`を開きます。

日本語化

config/locales/cms/ja.yml:アドオンの名称を登録

```
modules:  
  cms: 標準機能  
  addons:  
    cms/role: ロール  
    cms/group_permission: 権限  
    .....  
    cms/archive_view_switcher: アーカイブ用表示設定  
    cms/weather: 天気
```

日本語化

config/locales/cms/ja.yml:アドオンのフィールドを登録

```
cms/import_page: 取り込みページ
```

```
  attributes:
```

```
    cms/content:
```

```
      released: 公開日時
```

```
      .....
```

```
    cms/addon/archive_view_switcher:
```

```
      archive_view: 表示設定
```

```
cms/addon/weather:
```

```
  weather: 天気
```

日本語化

config/locales/cms/ja.yml:アドオンのツールチップを登録

tooltip:

cms/model/page:

destination_filename:

- ページやフォルダーを移動します。

.....

cms/addon/weather:

weather:

- 天気を選択します。

アドオンの確認

- ブラウザで表示を確認してみましょう。
- 正しく日本語化できていますか？

まとめ

- 次のファイルを修正または新規作成しました。
 - app/models/article/page.rb
 - app/models/concerns/cms/addon/weather.rb
 - app/views/cms/agents/addons/weather/_form.html.erb
 - app/views/cms/agents/addons/weather/_show.html.erb
 - app/views/cms/agents/addons/weather/view/index.html.erb
 - config/locales/cms/ja.yml
- 次のことを学びました。
 - アドオンのモデルとビューの関係
 - 日本語化
 - ツールチップ

演習

- 次の課題に取り組んでみてください。
 - モデルに日本語を直接書いています。日本語の部分は config/locales/cms/ja.yml に定義し、Ruby プログラム内からはロケールを参照するように修正してみてください。
 - 参考: app/models/concerns/cms/addon/release.rb
 - 参考: app/models/concerns/cms/addon/release.rb
 - 「晴れ」「曇り」「雨」「雪」の4種類の天気を拡張してみてください。
 - 雷注意報、にわか雨、お花見日和

ハンズオンの成果物

- 本ハンズオンの成果物は<https://github.com/shirasagi/ss-handson>のhandson/weather-addonブランチにあります。

ハンズオン3 Herokuへデプロイ

Herokuアカウント

- Herokuのアカウントをお持ちでない方？
 - ブラウザで <https://www.heroku.com/> にアクセスし、アカウントを作成してください。
- Herokuアカウントにクレジットカード番号を登録していない方？
 - 本ハンズオンで実施する作業は全て無料枠の中で実施されますが、MongoDBのHerokuプラグインを利用するには、クレジットカード番号の登録が必要です。

Heroku CLIのインストール

- ここまでで開発したものをHerokuへデプロイしてみます。
- Herokuにアプリをデプロイする方法は色々ありますが、もっとも一般的なコマンドラインからデプロイする方法でデプロイしてみます。
- まずは、コマンドラインからHerokuを操作するためのツールをインストールします。

```
$ wget -qO- https://toolbelt.heroku.com/install.sh | sh
$ echo 'PATH="/usr/local/heroku/bin:$PATH"' >> ~/.profile
$ source ~/.profile
$ heroku version
heroku-cli: Installing CLI... 23.42MB/23.42MB
heroku-toolbelt/3.43.16 (x86_64-linux) ruby/2.3.1
heroku-cli/5.5.8-df05424 (linux-amd64) go1.7.4
You have no installed plugins.
```

Heroku CLIでHerokuへログイン

- 次のコマンドを実行しHerokuへログインします。

```
$ heroku login
Enter your Heroku credentials.
Email: xxxx@xxxxxx.co.jp
Password (typing will be hidden):
Logged in as xxxx@xxxxxx.co.jp
```

Heroku上にアプリを作成

- 次のコマンドを実行し、Heroku上にアプリを作成します。

```
$ heroku create  
Creating app... done, ● xxxxx-yyyyyyyyy-zzzzz  
https://xxxxx-yyyyyyyyy-zzzzz.herokuapp.com/ | https://git.heroku.com/xxxxx-yyyyyyyyy-  
zzzzz.git
```

MongoDBの作成

- Heroku上にMongoDBを作成します。

```
$ heroku addons:create mongolab
Creating mongolab on ● xxxxx-yyyyyyyyy-zzzzz... free
Welcome to mLab. Your new subscription is being created and will be available shortly.
Please consult the mLab Add-on Admin UI to check on its progress.
Created mongolab-pointy-91072 as MONGODB_URI
Use heroku addons:docs mongolab to view documentation
```

Heroku向けに様々なMongoDBマネージドサービスが提供されていますが、本ハンズオンではmongolabのfreeプランを利用します。

シラサギのMongoDB設定の変更

- Heroku上のMongoDBを利用するように設定します。
- MongoDBのサーバーアドレスを調べるため、次のコマンドを実行します。

```
$ heroku config | grep MONGODB_URI
MONGODB_URI:
mongodb://heroku_hthct6w0:upbvkv5hutu0309qg3ib3hf2h@ds011790.mlab.com:11790/heroku_hthct6w0
```

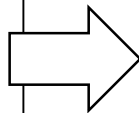
- 赤字の部分がMongoDBサーバーです。これをconfig/mongoid.ymlに設定します。

シラサギのMongoDB設定の変更

config/mongoid.yml

```
# mongodb configuration
production:
  clients:
    default:
      database: ss_sample
      hosts:
        - localhost:27017

development:
  clients:
    default:
      database: ss_sample
      hosts:
        - localhost:27017
```



```
# mongodb configuration
production:
  clients:
    default:
      uri:
mongodb://heroku_hthct6w0:upbvkv5hutu0309qg3ib3hf2h@ds
011790.mlab.com:11790/heroku_hthct6w0

development:
  clients:
    default:
      uri:
mongodb://heroku_hthct6w0:upbvkv5hutu0309qg3ib3hf2h@ds
011790.mlab.com:11790/heroku_hthct6w0
```

資料上は改行されていますが、1続きで入力してください。

シラサギをGrid FSモードへ変更

- Herokuの注意点として、Heroku上に作成したファイルが管理外の場合、アプリを再起動した場合などで削除されてしまいます。
- これを回避するために、シラサギをGrid FSモードへ変更します。
- 次のファイルを編集します。

```
$ vi config/environment.yml
```

```
# Default environment
RAILS_ENV: production

# enable csrf protect
protect_csrf: true

# File storage ('file' or 'grid_fs')
storage: grid_fs
```


閑話休題: Grid FSとは？

- MongoDBにファイルを保存することができる機能。
- MongoDBでは、1ドキュメント（RDBでいう1レコード）あたり16Mバイトという制限があるが、Grid FSを利用することで制限を超えたファイルを保存することが可能となる。
- MongoDBを複数台のサーバーからなるSharded Cluster構成にすることで、負荷が分散され、また、SPOFがなくなり堅牢性が向上する。このようなDB上にファイルを保存することができる。
- 注意点
 - 当社ではGrid FSでの本番運用経験はありません。
 - Grid FSモードで動作させるのは、評価目的に限定してください。

シラサギの設定ファイルをコミット

- シラサギをHeroku上で動作させるために必要な設定ファイルをコミットします。

```
$ git add -f config/environment.yml
$ git add -f config/mongoid.yml
$ git add -f config/secrets.yml
$ git add -f config/unicorn.rb
$ git commit -m "add configuration files"
```

シラサギをHerokuへデプロイ

- Herokuへデプロイします。

```
$ git push heroku HEAD:master
Counting objects: 51905, done.
Compressing objects: 100% (17389/17389), done.
Writing objects: 100% (51905/51905), 26.71 MiB | 5.30 MiB/s, done.
Total 51905 (delta 30152), reused 51905 (delta 30152)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Ruby app detected
remote: -----> Compiling Ruby/Rails
remote: -----> Using Ruby version: ruby-2.3.1
remote: -----> Installing dependencies using bundler 1.13.6
.....
remote: -----> Launching...
remote:          Released v6
remote:          https://xxxxx-yyyyyyyyy-zzzzz.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy.... done.
To https://git.heroku.com/xxxxx-yyyyyyyyy-zzzzz.git
 * [new branch]      master -> master
```

初期データの投入

- MongoDBが空の状態でシラサギが起動しました。
- 初期データを投入するため、つぎの4つのコマンドを上から順に実行します。

```
$ heroku run bundle exec rake db:drop
$ heroku run bundle exec rake db:create_indexes
$ heroku run bundle exec rake ss:create_site data='{ name: "サイト名", host: "www",
domains: "localhost:3000" }'
$ heroku run bundle exec rake db:seed name=demo site=www
```

後で管理画面から変更するので、
ここでは何でもいいです。

ブラウザでのアクセス

- ブラウザでHeroku上のシラサギにアクセスしてみましょう。
- 次のコマンドを実行し、URLを確認します。

```
$ heroku apps:info
=== quiet-mountain-66056
Addons:          heroku-postgresql:hobby-dev
                 mongolab:sandbox
Dynos:          web: 1
Git URL:        https://git.heroku.com/xxxxx-yyyyyyyyy-zzzzz.git
Owner:          xxxx@xxxxxx.co.jp
Region:         us
Repo Size:      28 MB
Slug Size:      81 MB
Stack:          cedar-14
Web URL:        https://xxxxx-yyyyyyyyy-zzzzz.herokuapp.com/
```

- 一番下にWeb URLが表示されています。このURLに.mypageを付けたURLにブラウザでアクセスします。

```
https://xxxxx-yyyyyyyyy-zzzzz.herokuapp.com/.mypage
```

ブラウザでのアクセス



The screenshot shows a web browser window displaying the SHIRASAGI login page. The page has a dark header with the SHIRASAGI logo and name. Below the header is a login form titled "ログイン" (Login) with version "ver. 1.4.1". The form contains two input fields: "ユーザーIDまたはメールアドレス" (User ID or email address) with the value "sys@example.jp" and "パスワード" (Password) with masked characters "....". A red "ログイン" button is positioned below the fields. A blue callout box on the right side of the page contains the text: "システム管理者 (sys@example.jp / pass) でログイン" (Login as system administrator).

サイト情報の修正

- シラサギは複数のサイトを管理することができます。
- 内部的にはドメインとポートをサイトに紐付けることで実現しています。
 - Webサーバーのリバースプロキシと同じ仕組みだと考えると分かりやすいです。
- 初期データ投入時に“localhost:3000”を紐付けてしまったので修正します。

サイト情報の修正

サイト管理画面 -> サイト設定 -> サイト情報 -> 編集

The screenshot displays the SHIRASAGI website management interface. The top navigation bar includes the SHIRASAGI logo, a search icon, and user information (policy department, system administrator). The left sidebar lists various site settings, with 'Site Information' selected. The main content area shows the 'Site Information' editing form, which includes fields for Site Name, Host Name, Domain, and HTTPS status. The 'Domain' field is highlighted with a red border, indicating it is the current focus.

SHIRASAGI

サイト管理 グループ 政策課 システム管理者

サイト名 サイト情報

詳細へ戻る

基本情報

サイト名

ホスト名

ドメイン

HTTPS

ページ設定

公開予約の既定値

モバイル設定

地図設定

かな設定

サイト確認 サイトプレビュー

サイト設定

- サイト情報
- グループ
- ユーザー
- 権限/ロール
- ワークフロー
- メンバー
- お知らせ
- テンプレート
- Theme切り替え
- ソースクリーニング
- 本文レイアウト
- ページ検索
- 郵便番号
- かな辞書
- 組織変更
- LDAP
- リンクチェック
- 読み上げ音声

フォルダー書き出しとページ書き出し

- シラサギには静的コンテンツをHTMLに書き出しておくことで、必要最小限のDBアクセスでコンテンツを応答することができる機構が備わっています。
- 先ほどサイトの情報を更新したので、書き出されているHTMLを更新しなければなりません。
- 左側のナビビューにある「フォルダー書き出し」と「ページ書き出し」をそれぞれ実行してみましょう。

フォルダー書き出し

The screenshot shows the SHIRASAGI website management interface. The top navigation bar includes the SHIRASAGI logo, a search icon, and user information (policy department, system administrator). The left sidebar contains various management options, with 'フォルダー書き出し' (Folder Export) selected. The main content area displays the site name, status (stop), and processing progress (3/0 items). A list of files to be exported is shown, including /404.html, /index.html, /mobile.html, and several calendar files. A red '実行' (Execute) button is visible at the bottom of the file list.

SHIRASAGI サイト管理 グループ 政策課 システム管理者

サイト名

ステータス stop 処理件数 3 / 0 開始日時 2016-12-12 20:56:29 終了日時 2016-12-12 20:58:17

サイト名
/404.html
/index.html
/mobile.html
/attention/index.html
/board/index.html
/calendar/201512.html
/calendar/201601.html
/calendar/201602.html
/calendar/201603.html
/calendar/201604.html
/calendar/201605.html
/calendar/201606.html
/calendar/201607.html
/calendar/201608.html
/calendar/201609.html

実行

ページ書き出し

The screenshot displays the SHIRASAGI website management interface. The top navigation bar includes the SHIRASAGI logo, a gear icon for 'サイト管理' (Site Management), a user icon for 'グループ' (Group), and a dropdown menu for '政策課 システム管理者' (Policy Department System Administrator). The left sidebar contains a search bar and a list of navigation options: 'サイト確認' (Site Confirmation), 'サイトプレビュー' (Site Preview), 'コンテンツ' (Content), 'フォルダー' (Folder), '固定ページ' (Fixed Page), 'パーツ' (Part), 'レイアウト' (Layout), '共有ファイル' (Shared File), 'サイト内検索' (Site Search), 'サイト設定' (Site Settings), 'フォルダー取り込み' (Folder Import), '全コンテンツ一覧出力' (Export All Content List), 'フォルダー書き出し' (Export Folder), and 'ページ書き出し' (Export Page), which is currently selected.

The main content area shows the 'サイト名' (Site Name) section with the following details:

- ステータス: stop
- 処理件数: 61 / 61
- 開始日時: 2016-12-12 21:01:03
- 終了日時: 2016-12-12 21:02:39

A list of files to be exported is displayed in a scrollable area:

- # サイト名
- /404.html
- /ad/page30.html
- /ad/page31.html
- /ad/page32.html
- /ad/page33.html
- /ad/page34.html
- /ad/page35.html
- /anpi-ezine/anpi/anpi37.html
- /anpi-ezine/event/page38.html
- /calendar/page28.html
- /docs/body_layout.html
- /docs/page1.html
- /docs/page10.html
- /docs/page11.html
- /docs/page12.html

A red '実行' (Execute) button is located at the bottom of the file list.

サイトの確認

- 公開画面を確認してみます。
- 「サイト確認」ボタンをクリックしてみてください。

サイトの確認

- 次の画面が表示されれば成功です。

The screenshot shows the homepage of Shirasagi City. At the top, there is a navigation bar with links for 'ご利用案内', 'ふりがなをつける', and '文字サイズ' (with options for '小さく', '標準', '大きく'). Below this is the city logo and name 'シラサギ市', along with links for 'スマホ・携帯サイト', 'お問い合わせ', and 'サイトマップ'. A search bar is also present. A main menu bar contains categories: 'くらし・手続き', '子育て・教育', '健康・福祉', '観光・文化・スポーツ', '産業・仕事', and '市政情報'. The main content area features a large video player showing pink flowers. To the right, there is a '注目情報' section with several news items, and an 'お知らせ' section with more news. On the left, there are sections for 'くらしのガイド' (with icons for marriage, pregnancy, childcare, moving, welfare) and '安心安全情報' (with links for disaster, fire, traffic, and hazard maps). At the bottom left, there is a section for 'オンラインサービス' with links for electronic applications, document downloads, and facility reservations.

Heroku上のアプリを削除

- Heroku上のアプリが起動しっぱなしだと、誤って課金されてしまうかもしれません。
- 最後にHeroku上のアプリを削除します。

```
$ heroku apps:destroy --app xxxxx-yyyyyyyyy-zzzzz
```

まとめ

- 次のことを学びました。
 - Herokuへのデプロイ

ハンズオン4 一覧の拡張

一覧ノードの特長

- 一覧ノードにはリスト表示アドオンが組み込まれています。

■検索条件 (URL)

URLとありますが、フォルダーのパスを指定します。次のような特長があります。

- 指定がない場合、自フォルダー下のフォルダーやページを対象とする。
- 指定がある場合、指定されたフォルダー下のフォルダーやページを対象とする。
- 複数指定することができる。

リスト表示

検索条件(URL) ?

並び順 ?

表示件数 ?

上部HTML ?

ループHTML ?

下部HTML ?

NEWマーク期間 ?

0 日

一覧の並び順と表示件数の指定

■上部HTML、ループHTML、下部HTML

一覧の表示を調整することができ、ループHTMLには、右図にあるように#{date.long}や#{index_name}などの変数を埋め込むことが可能となっています。

「新着」と判定する日数

一覧ノードの特長

- 次のような特長があります。
 - 他のフォルダー内のフォルダーやページを一覧表示できる。
 - 一覧表示を上部HTML、ループHTML、下部HTMLに分割し、それぞれカスタマイズできる。
 - ループHTMLには、変数を記述することができる。

天気を一覧に表示

- 本ハンズオンでは、ループHTMLに`{weather}`と書くことで天気を、`{weather_code}`と書くことで天気コードを表示できるように拡張してみます。
- 対応するのは簡単で、アドオン内で`template_variable_handler`を用いて変数のハンドラーを登録するだけです。
- `app/models/concerns/cms/addon/weather.rb`をテキストエディタで開き、ハンドラーを登録してみましょう。

天気を一覧に表示

app/models/concerns/cms/addon/weather.rb

```
module Cms::Addon
  module Weather
    extend ActiveSupport::Concern
    extend SS::Addon

    included do
      field :weather, type: String
      permit_params :weather

      template_variable_handler :weather, :template_variable_handler_weather
      template_variable_handler :weather_code, :template_variable_handler_weather_code
    end

    def weather_options
      [ ["晴れ", "sunny"], ["曇り", "cloudy"],
        ["雨", "rain"], ["雪", "snow"],
      ]
    end

    def template_variable_handler_weather(*args)
      ERB::Util.html_escape(label(:weather))
    end

    def template_variable_handler_weather_code(*args)
      ERB::Util.html_escape(weather)
    end
  end
end
```

確認

- 記事フォルダのループHTMLを次のように設定

```
<article class="item-#{class} #{new}">
  <header>
    <time datetime="#{date.iso}">#{date.long}</time>
    <h2><a href="#{url}">#{index_name}</a></h2>
    <span class="#{weather_code}">#{weather}</span>
  </header>
</article>
```

- PCプレビューをクリック

2016/12/13 13:24 PC 携帯

2016年12月13日 [広報SHIRASAGI3月号を掲載](#)

2016年12月13日 [自動交付機・コンビニ交付サービスについて](#) 雨

2016年12月13日 [転居届](#) 曇り

2016年12月13日 [ふれあいフェスティバル](#) 晴れ

2016年12月13日 [冬の感染症に備えましょう](#)

前の月へ 翌月へ 次の月へ

2016年12月

日	月	火	水	木	金	土
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

- 天気が表示されれば成功です。

まとめ

- 次のファイルを修正しました。
 - `app/models/concerns/cms/addon/weather.rb`
- 次のことを学びました。
 - 一覧表示フォルダの概要
 - ループHTMLの変数の拡張方法

演習

- 次の課題に取り組んでみてください。
 - ページにカテゴリーを設定し、カテゴリーフォルダーのループHTMLを変更してみてください。
 - 「標準機能/ページリスト」フォルダーを新規作成し、検索条件URLに記事フォルダーのパスを、ループHTMLに天気情報を含むHTMLを記述してみてください。
 - ツールチップを修正し、今回開発したテンプレート変数の説明を追記してみてください。

ハンズオンの成果物

- 本ハンズオンの成果物は<https://github.com/shirasagi/ss-handson>のhandson/list-extブランチにあります。

ハンズオン5 検索フォルダーの開発

検索機能の開発

- ハンズオン2で開発した天気を検索できるようにしてみましょう。
- 方法としては2案あります。
 - 案1: 既存の記事フォルダーに検索機能を追加する。
 - 案2: 天気を検索する機能をもったフォルダーを新規開発する。
- 案1のメリット
 - メソッド1つ、公開側のビューを1つ、合計2つ開発するだけなので簡単。
- 案1のデメリットとして:
 - 検索ページのパンくずが記事一覧と同じになる。
 - レイアウトが記事一覧と同じになる。

検索機能の開発

- 案1の場合、レイアウトが変更されないという点がネックとなります。
- なぜかという点と……
 - 記事一覧レイアウトに天気検索パーツを組み込み、右サイドに表示したとします。

記事一覧レイアウトに
転記パーツを組み込み



- 天気検索のレイアウトは記事一覧と同じになるので、天気検索の右サイドにも天気検索パーツが表示されてしまうので、検索フォームが2重に表示されます（下図）。



天気検索を記事一覧ノードの機能として開発すると

- パンくずが「記事」となる。
- レイアウトが変更されないため、検索フォームが2重に表示される。

- 本ハンズオンでは案2の方法で実装します。

モデルの作成

- フォルダー「記事/天気検索」（属性: article/weather_search）を開発してみます。
- app/models/article/node.rbをテキストエディタで開き、末尾辺りにノードを追加します。
- ここでは、雛形としてフォルダー「標準機能/ページリスト」（英: cms/page）をコピーし、修正します。

モデルの作成

app/models/article/node.rb

```
class WeatherSearch
  include Cms::Model::Node
  include Cms::Addon::NodeSetting
  include Cms::Addon::Meta
  include Event::Addon::PageList
  include Cms::Addon::Release
  include Cms::Addon::DefaultReleasePlan
  include Cms::Addon::GroupPermission
  include History::Addon::Backup

  default_scope ->{ where(route: "article/weather_search") }
end
```

フォルダー属性

赤字の箇所がcms/pageからの変更点



ノードの登録

- 作成したノードをリポジトリに登録します。
- リポジトリに登録することで、シラサギでフォルダー作成時に、一覧に表示されるようになります。
- リポジトリの登録はapp/models/article/initializer.rbに実装しますので、 app/models/article/initializer.rbをテキストエディタで開きます。



ノードの登録

app/models/article/initializer.rb

```
module Article
  class Initializer
    Cms::Node.plugin "article/page"
    Cms::Node.plugin "article/weather_search"
    Cms::Part.plugin "article/page"
  end
end
```

Cms::Node.pluginにフォルダ属性を引数で渡すと、リポジトリに登録されます。

確認

- ウェブブラウザをリロードし、フォルダーを新規作成してみます。
- 次のようにWeather Searchが表示されれば成功です。



確認

- 下の図を参考に天気検索フォルダーを作成します。
 - 記事フォルダーを検索条件URLに設定してください。

The screenshot displays the configuration page for a folder named '天気検索' (Weather Search). The interface is organized into several sections:

- 基本情報 (Basic Information):** Contains fields for 'フォルダー属性' (Folder Attribute) set to '記事/Article/Weather Search', 'タイトル' (Title) set to '天気検索', '一覧用タイトル' (List Title), 'フォルダー名' (Folder Name) set to 'weather_search', and 'レイアウト' (Layout) set to '1カラム'.
- フォルダー設定 (Folder Settings):** This section is currently empty.
- メタ情報 (Meta Information):** This section is currently empty.
- リスト表示 (List Display):** Contains a '検索条件(URL)' (Search Condition (URL)) field set to 'docs' and a '並び順' (Sort Order) dropdown menu.

The sidebar on the left provides navigation for various site management tasks, including folder creation and content management.

フォルダー属性の日本語化

- フォルダー属性が英語で表示されていますので日本語化してみましょう。
- config/locales/article/ja.ymlをテキストエディタで開きます。

config/locales/article/ja.yml

```
cms:  
  nodes:  
    article/page: 記事リスト  
    article/weather_search: 天気検索  
  parts:  
    article/page: 記事リスト
```

確認

- 下のようによりフォルダ属性が日本語化されれば成功です。

The screenshot shows a settings interface for a folder. On the left is a sidebar with navigation options: フォルダ, 固定ページ, パーツ, レイアウト, 共有ファイル, サイト内検索, サイト設定, フォルダ取り込み, 全コンテンツ一覧出力, フォルダ書き出し, ページ書き出し. The main content area is divided into sections: 基本情報, フォルダ設定, メタ情報, and リスト表示. In the 基本情報 section, the 'フォルダ属性' field is circled in orange and contains the text '記事/天気検索' next to a '変更する' button. Other fields in this section include 'タイトル' (天気検索), '一覧用タイトル', 'フォルダ名' (weather_search), and 'レイアウト' (1カラム). The リスト表示 section contains '検索条件(URL)' (docs) and '並び順'.

Routing Error

- 作成したフォルダーをクリックしてみてください。エラーが表示されます。

☐ 天気検索

#219 2016/12/13 20:52 weather_search 記事 公開中



クリック

Routing Error

No route matches [GET] "/s1/article219/weather_searches"

Rails.root: /home/vagrant/sample

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

Routes

Routes match in priority from top to bottom

Helper	HTTP Verb	Path
Path / Url		<input type="text" value="Path Match"/>
sns_mypage_path	GET	/mypage(.:format)
sns_logout_path	GET	/mypage/logout(.:format)
sns_login_path	GET POST	/mypage/login(.:format)
sns_remote_login_path	GET POST	/mypage/remote_login(.:format)
sns_login_status_path	GET	/mypage/status(.:format)
sns_auth_token_path	GET	/mypage/auth_token(.:format)
sns cms_path	GET	/mypage/cms(.:format)
sns_gws_path	GET	/mypage/gws(.:format)
sns_login_saml_path	GET	/mypage/login/saml/:id/init(.:format)
	POST	/mypage/login/saml/:id/consume(.:format)
sns_login_saml_metadata_path	GET	/mypage/login/saml/:id/metadata(.:format)
sns_login_open_id_connect_path	GET	/mypage/login/oid/:id/init(.:format)
sns_login_open_id_connect_callback_path	GET POST	/mypage/login/oid/:id/callback(.:format)



Routing Error

- ルーティングが作成されていないため発生しているエラーです
- ルーティングを作成しましょう。
- `config/routes/article/routes.rb`をテキストエディタで開きます。

- ルーティングを作成する前に、シラサギの特徴的なルーティングを説明します。

シラサギrouting

- 管理画面

- Railsの仕組みを用いて普通にルーティングを定義
- ただし、毎度同じ処理を記述するのは面倒なのでDSLのサポートあり
- 逆にDSLを知らないと少し分かりにくいかも

- 公開画面

- サイト構築者が自由にURL（サイト構造）を設計できるようになっているため、開発時に公開画面のURLは決まらないためルーティングできない
- Railsの構文を拡張し、シラサギ独自方式でルーティングを定義
- DSLのサポートあり



記事ページ / ノードの管理画面のルーティング

```
content "article" do
  get "/" => redirect { |p, req| "#{req.path}/pages" }, as: :main
  get "generate" => "generate#index"
  post "generate" => "generate#run"
  resources :pages, concerns: [:deletion, :copy, :move, :lock, :download, :import, :opendata_ref]
end
```

- “content”というDSLを用いてルーティングを定義
- 1行目は“/article/”にアクセスすると“/article/pages”へリダイレクトするルーティング
- 2行目と3行目は書き出し処理（割愛）
- 4行目が記事ページと記事ノードの管理画面のルーティング
 - article/pageという属性のノードにアクセスされたらという意味です。
- ※記事ページと記事ノードの管理画面のコントローラは同じためルーティングは1つ

記事ノードの公開画面のルーティング

```
node "article" do
  get "page/(index.:format)" => "public#index", cell: "nodes/page"
end
```

- “node”というDSLを用いる
- 1+2: article/page という route 属性を持つノードがアクセスされたら
- 1+4: article/agents/nodes/page_controller.rb というコントローラーの
- 3: indexというアクションを実行する
- ※グレーの箇所は固定で決まっています。

ルーティングの作成

- 作成したノードの属性はarticle/weather_searchでした。
- このノードのルーティングを定義するので、次のようにすれば良さそうです。

管理側のルーティング

```
content "article" do
  get "/" => redirect { |p, req| "#{req.path}/pages" }, as: :main
  get "generate" => "generate#index"
  post "generate" => "generate#run"
  resources :pages, concerns: [:deletion, :copy, :move, :lock, :download, :import, :opendata_ref]
  resources :weather_searches, concerns: [:deletion]
end
```

公開側のルーティング

```
node "article" do
  get "page/(index.:format)" => "public#index", cell: "nodes/page"
  get "page/rss.xml" => "public#rss", cell: "nodes/page", format: "xml"
  get "weather_search/(index.:format)" => "public#index", cell: "nodes/weather_search"
end
```

ルーティングの作成と確認

- 全ページのルーティングをconfig/routes/article/routes.rbに記述したら、ブラウザをリロードしてみましょう。
- エラーメッセージが変わりました。今度はコントローラーが存在しないというエラーです。

Routing Error

uninitialized constant Article::WeatherSearchesController

Rails.root : /home/vagrant/sample

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

Routes

Routes match in priority from top to bottom

Helper	HTTP Verb	Path
Path / Url		<input type="text" value="Path Match"/>
sns_mypage_path	GET	/mypage(.:format)
sns_logout_path	GET	/mypage/logout(.:format)
sns_login_path	GET POST	/mypage/login(.:format)
sns_remote_login_path	GET POST	/mypage/remote_login(.:format)
sns_login_status_path	GET	/mypage/status(.:format)
sns_auth_token_path	GET	/mypage/auth_token(.:format)
sns_cms_path	GET	/mypage/cms(.:format)
sns_gws_path	GET	/mypage/gws(.:format)
sns_login_saml_path	GET	/mypage/login/saml/:id/init(.:format)
	POST	/mypage/login/saml/:id/consume(.:format)
sns_login_saml_metadata_path	GET	/mypage/login/saml/:id/metadata(.:format)
sns_login_open_id_connect_path	GET	/mypage/login/oid/:id/init(.:format)
sns_login_open_id_connect_callback_path	GET POST	/mypage/login/oid/:id/callback(.:format)

コントローラーの作成（管理側）

- app/controllers/article/weather_searches_controller.rbを作成しましょう。

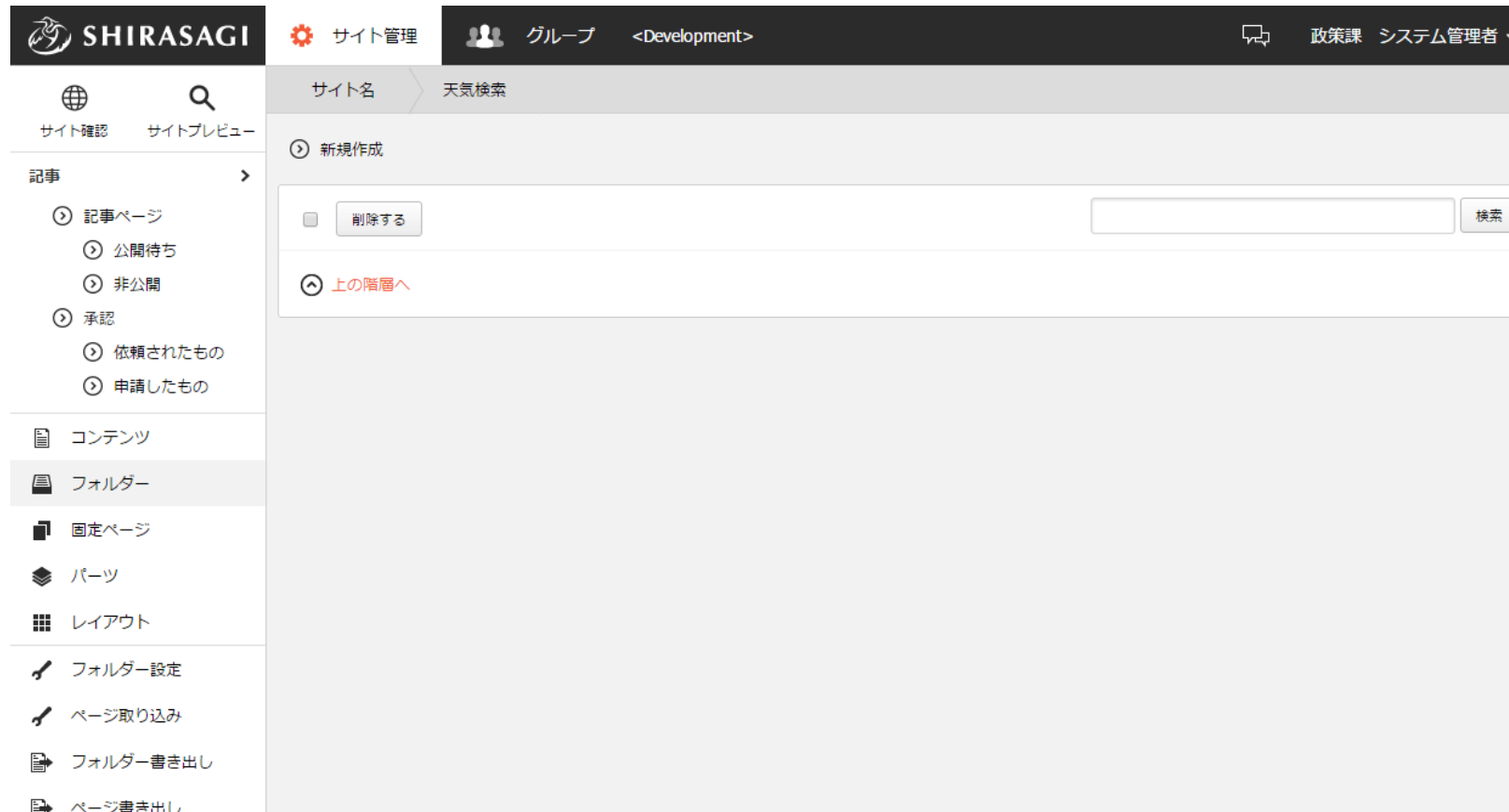
app/controllers/article/weather_searches_controller.rb

```
class Article::WeatherSearchesController < ApplicationController
  def index
    redirect_to node_nodes_path
  end
end
```

- 非常に簡素なコントローラーで、標準機能/フォルダーリストにリダイレクトするだけのコントローラーを作成します。
- 作成したらブラウザをリロードしてみましょう。

コントローラーの確認

- ようやくフォルダーの中身が表示されました。
- 天気検索ノードの管理側の実装は以上です。



公開側の実装

- 公開側の実装に進みます。
- 天気検索フォルダーの「フォルダー設定」から「PCプレビュー」をクリックします。
- まだコントローラーを作成していないので、エラーが表示されます。

NameError in Cms::PreviewController#index

uninitialized constant Article::Agents::Nodes::WeatherSearchController

Extracted source (around line #6):

```
4 def initialize(controller)
5   if controller.is_a?(String)
6     controller = "#{controller}_controller".camelize.constantize
7   end
8   @controller = controller.new
9   @controller.params = ActionController::Parameters.new
```

Rails.root: /home/vagrant/sample

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

```
lib/ss/agent.rb:6:in `initialize'
app/controllers/application_controller.rb:16:in `new'
app/controllers/application_controller.rb:16:in `new_agent'
app/controllers/concerns/cms/public_filter/node.rb:27:in `render_node'
app/controllers/concerns/cms/public_filter.rb:37:in `index'
```

Request

Parameters:

```
{"site"=>"1",
 "path"=>"weather_search"}
```

[Toggle session dump](#)

公開側のコントローラーの作成

- `app/controllers/article/agents/nodes/weather_search_controller.rb`を作成します。
- 次のような空のコントローラーを作成します。

```
class Article::Agents::Nodes::WeatherSearchController < ApplicationController
  include Cms::NodeFilter::View
  helper Cms::ListHelper

  def index
  end
end
```

閑話休題: Cms::NodeFilter::View

- 便利なメソッドや処理がつまったモジュールで、次のメンバー変数が設定されています。
 - @cur_site: 処理対象となるサイトが設定されている。
 - @cur_node: 処理対象となるノードが設定されている。
 - @cur_date: 処理対象となる日時が設定されている。
 - プレビュー時、未来日で処理する場合があります。
 - ページネーション付きのビューを表示するための便利なメソッド `render_with_pagination` が提供されている。



公開側コントローラーの確認

- コントローラーを作成したらブラウザをリロードしてみましょう。
- 今度はビューがないというエラーになりました。

Template is missing

Missing template article/agents/nodes/weather_search/index, application/index with {:locale=>[:ja, :en], :formats=>[:html], :variants=>[], :handlers=>[:erb, :builder, :raw, :ruby, :coffee, :builder]}. Searched in: *"/home/vagrant/sample/app/views" *
"/usr/local/rvm/gems/ruby-2.3.1/gems/kaminari-0.16.3/app/views"

Extracted source (around line #27):

```
25   def render_with_pagination(items)
26     raise "404" if params[:page].to_i > 1 && items.empty?
27     render
28   end
29 end
```

Rails.root: /home/vagrant/sample

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

```
app/controllers/concerns/cms/node_filter/view.rb:27:in `render_with_pagination'
app/controllers/article/agents/nodes/weather_search_controller.rb:18:in `index'
lib/ss/agent.rb:29:in `render'
app/controllers/concerns/cms/public_filter/node.rb:29:in `render_node'
app/controllers/concerns/cms/public_filter.rb:37:in `index'
```

Request

Parameters:

```
{"site"=>"1",
 "path"=>"weather_search"}
```

[Toggle session dump](#)

公開側のビューの作成

- 公開側のビューを作成してみます。

```
$ mkdir app/views/article/agents/nodes/weather_search
$ vi app/views/article/agents/nodes/weather_search/index.html
```

```
<%= form_tag @cur_node.url, method: "get" do |f| %>
  <h2>天気</h2>
  <div class="weather">
    <label><%= check_box_tag 'weathers[]', 'sunny' %>晴れ</label>
    <label><%= check_box_tag 'weathers[]', 'cloudy' %>曇り</label>
    <label><%= check_box_tag 'weathers[]', 'rain' %>雨</label>
    <label><%= check_box_tag 'weathers[]', 'snow' %>雪</label>
  </div>
  <footer class="send">
    <%= submit_tag '検索', name: nil %>
  </footer>
<% end %>
```

公開側ビューの確認

- 次のような画面が表示された成功です。

2016/12/13 21:48 PC 携帯

[ご利用案内](#) [ふりがなをつける](#) [読み上げる](#) 背景色 [白](#) [青](#) [黒](#) 文字サイズ [小さく](#) [標準](#) [大きく](#)

 **シラサギ市** [スマホ・携帯サイト](#) [お問い合わせ](#) [サイトマップ](#)

サイト内検索

[くらし・手続き](#) [子育て・教育](#) [健康・福祉](#) [観光・文化・スポーツ](#) [産業・仕事](#) [市政情報](#)

[HOME](#) > [天気検索](#)

天気検索

天気
 晴れ 曇り 雨 雪

シラサギ市役所 〒000-0000 大鷲県シラサギ市小鷲町1丁目1番地1号 [市役所のご案内](#)
電話番号：00-000-0000 ファクシミリ：00-000-0000

開庁時間 月曜日から金曜日の午前8時30分から午後5時(祝日・休日、12月29日から1月3日を除く)
(注)部署、施設によっては、開庁・開館の日・時間が異なる場合があります。

[個人情報の取り扱い](#) [リンク](#) [著作権・免責事項](#) [webアクセシビリティ](#) [RSSについて](#)

Copyright © City of Shirasagi All rights Reserved.

検索機能の実装

- 公開側のコントローラーとビューに検索機能を実装していきます。

- 検索条件パラメータの取得

- 晴れと曇りを検索すると次のようなURLになります。

`http://localhost:3000/weather_search/?utf8=%E2%9C%93&weathers%5B%5D=sunny&weathers%5B%5D=cloudy`

- コントローラー側では次のように取得できます。

`params[:weathers]`

- `weathers`と複数形になっていることから分かるように文字列の配列として検索パラメータを取得することができます。

- コントローラーとビューを一気に実装していきます。

検索機能の実装: コントローラー

app/controllers/article/agents/nodes/weather_search_controller.rb

```
class Article::Agents::Nodes::WeatherSearchController < ApplicationController
  include Cms::NodeFilter::View
  helper Cms::ListHelper

  def index
    # 検索条件が設定されていない場合、検索画面を表示
    @weathers = params[:weathers]
    return if @weathers.blank?

    # 検索
    @items = Article::Page.site(@cur_site).and_public(@cur_date).
      where(@cur_node.condition_hash).
      in(weather: @weathers).
      order_by(@cur_node.sort_hash).
      page(params[:page]).
      per(@cur_node.limit)

    render_with_pagination @items
  end
end
```

検索結果の表示: ビュー

app/views/article/agents/nodes/weather_search/index.html.erb

```
<%= form_tag @cur_node.url, method: "get" do |f| %>
  <h2>天気</h2>
  <div class="weather">
    <label><%= check_box_tag 'weathers[]', 'sunny' %>晴れ</label>
    <label><%= check_box_tag 'weathers[]', 'cloudy' %>曇り</label>
    <label><%= check_box_tag 'weathers[]', 'rain' %>雨</label>
    <label><%= check_box_tag 'weathers[]', 'snow' %>雪</label>
  </div>
  <footer class="send">
    <%= submit_tag '検索', name: nil %>
  </footer>
<% end %>

<% if @items.present? %>
  <div class="pages">
    <%= render_page_list %>
  </div>

  <%= paginate @items if @items.try(:current_page) %>
<% end %>
```

閑話休題: render_page_list

- 上部HTML、ループHTML、下部HTMLを用いて一覧をレンダリングするヘルパーメソッドです。
- このメソッドを呼ぶだけで、上部HTML、ループHTML、下部HTMLをサポートすることができるようになります。



確認

- ブラウザをリロードし次のように表示されれば成功です。

▼ 本文へ | [ご利用案内](#) | [ふりがなをつける](#) | [読み上げる](#) | 背景色 | 文字サイズ

 **シラスギ市** [スマホ・携帯サイト](#) [お問い合わせ](#) [サイトマップ](#)

サイト内検索

[くらし・手続き](#) [子育て・教育](#) [健康・福祉](#) [観光・文化・スポーツ](#) [産業・仕事](#) [市政情報](#)

[HOME](#) > [天気検索](#)

天気検索

天気
 晴れ 曇り 雨 雪

2016年12月13日
[自動交付機・コンビニ交付サービスについて](#)
2016年12月13日
[ふれあいフェスティバル](#)

シラスギ市役所 〒000-0000 大鷲県シラスギ市小鷲町1丁目1番地1号 [市役所のご案内](#)
電話番号：00-000-0000 ファクシミリ：00-000-0000

開庁時間 月曜日から金曜日の午前8時30分から午後5時(祝日・休日、12月29日から1月3日を除く)
(注)部署、施設によっては、開庁・開館の日・時間が異なる場合があります。

[個人情報の取り扱い](#) [リンク](#) [著作権・免責事項](#) [webアクセシビリティ](#) [RSSについて](#)

Copyright © City of Shirasagi All rights Reserved.

まとめ

- 次のファイルを新規作成または修正しました。
 - app/controllers/article/agents/nodes/weather_search_controller.rb
 - app/controllers/article/weather_searches_controller.rb
 - app/models/article/initializer.rb
 - app/models/article/node.rb
 - app/views/article/agents/nodes/weather_search/index.html.erb
 - config/locales/article/ja.yml
 - config/routes/article/routes.rb
- 次のことを学びました。
 - フォルダの作成方法
 - 検索処理の作成方法
 - 一覧画面の作成方法

演習

- 次の課題に取り組んでみてください。
 - ビューに日本語を直接書きました。これらをロケールに移動してみてください。
 - 検索ボタンをクリックすると設定した検索条件がクリアされてしまいます。クリアされないようにしてみてください。
 - 検索条件をクリアする「リセット」を作成してみてください。

ハンズオンの成果物

- 本ハンズオンの成果物は<https://github.com/shirasagi/ss-handson>のhandson/weather-search-nodeブランチにあります。



ハンズオン6 検索パーツの開発



パーツの開発

- 現在のシラサギでは、ノードの機能をトップページとすることはできません。
 - トップページは固定ページでなければなりません。
- また、記事一覧ページに記事検索のフォームをナビゲーションの下やサイドバーに表示したいでしょう。
- このようなことをするにはパーツが必要です。
- 本ハンズオンでは前章で開発した検索ノードのパーツを開発してみます。

検索パーツの仕様

- 検索パーツは、検索フォームを表示するのみとする。
 - 検索機能は検索ノードが提供する。
 - 検索パーツの検索ボタンがクリックされたら、検索ノードへ遷移する。
- 検索パーツは、検索ノードと連携して検索機能を提供します。

検索パーツの仕様

- ここで問題が一つあります。検索パーツは検索ノードの位置をどのようにして知るのでしょうか？
- シラサギはCMSです。好きな階層に好きな名前で作成することができます。
- 検索ノードは一つとは限らず、複数個作成される可能性があります。

検索パーツの仕様

- 「検索パーツは、検索ノードと連携して検索機能を提供」という仕様でした。
- この仕様の裏を返せば、検索パーツは単独で存在することはできず、必ずペアになる検索ノードが必要です。
- ここから、検索パーツの妥当な仕様制限として、次の仕様制限が導かれます：
 1. 検索パーツは検索ノードの下に作成する。
 2. 検索パーツの親ノードを検索ノードとみなす。
- 本ハンズオンでは、このような仕様で検索パーツを開発します。

モデルの作成

- パーツ「記事/天気検索」（属性: article/weather_search）を開発してみます。
- app/models/article/part.rbをテキストエディタで開き、末尾辺りにパーツを追加します。

モデルの作成

app/models/article/part.rb

```
class WeatherSearch
  include Cms::Model::Part
  include Cms::Addon::Release
  include Cms::Addon::GroupPermission
  include History::Addon::Backup

  default_scope ->{ where(route: "article/weather_search") }
end
```

パーツ属性

パーツの登録

- ノードと同じように、作成したパーツをリポジトリに登録します。
- リポジトリに登録することで、シラサギでフォルダー作成時に、一覧に表示されるようになります。
- リポジトリの登録はapp/models/article/initializer.rbに実装しますので、 app/models/article/initializer.rbをテキストエディタで開きます。



パーツの登録

app/models/article/initializer.rb

```
module Article
  class Initializer
    Cms::Node.plugin "article/page"
    Cms::Node.plugin "article/weather_search"
    Cms::Part.plugin "article/page"
    Cms::Part.plugin "article/weather_search"
```

Cms::Part.pluginにパーツ属性を引数で渡すと、リポジトリに登録されます。

確認

- ウェブブラウザをリロードし、パーツを新規作成してみます。
- 次のようにWeather Searchが表示されれば成功です。



確認

- 下の図を参考に天気検索パーツを天気フォルダーの下に作成します。

The screenshot shows the SHIRASAGI website management interface. The top navigation bar includes 'SHIRASAGI', 'サイト管理' (Site Management), 'グループ' (Group) set to '<Development>', and '政策課 システム管理者' (Policy Department System Administrator). The left sidebar contains navigation options such as '記事' (Articles), 'コンテンツ' (Contents), 'フォルダー' (Folders), '固定ページ' (Fixed Pages), 'パーツ' (Parts), 'レイアウト' (Layouts), 'フォルダー設定' (Folder Settings), 'ページ取り込み' (Page Import), 'フォルダー書き出し' (Folder Export), and 'ページ書き出し' (Page Export). The main content area is titled '天気検索' (Weather Search) and is highlighted with an orange circle. A callout box points to this area with the text '天気検索フォルダーの下に作成' (Create under the Weather Search folder). The '基本情報' (Basic Information) section shows the part name '天気検索' (circled in orange) and the file name 'weather_search'. Other settings include '記事/Article/Weather Search' (with an '変更する' button), '携帯向け表示' (Mobile display) set to '表示' (Display), and '動的表示' (Dynamic display) set to '無効' (Ineffective). At the bottom, there are '保存' (Save) and 'キャンセル' (Cancel) buttons.

パーツ属性の日本語化

- パーツ属性が英語で表示されていますので日本語化してみましょう。
- `config/locales/article/ja.yml`をテキストエディタで開きます。

`config/locales/article/ja.yml`

```
cms:  
  nodes:  
    article/page: 記事リスト  
    article/weather_search: 天気検索  
  parts:  
    article/page: 記事リスト  
    article/weather_search: 天気検索
```

確認

- 下のようによりフォルダー属性が日本語化されれば成功です。

The screenshot shows the SHIRASAGI website management interface. The top navigation bar includes the SHIRASAGI logo, 'サイト管理' (Site Management), 'グループ <Development>' (Group <Development>), and '政策課 システム管理者' (Policy Section System Administrator). The main content area is titled '天気検索' (Weather Search) and includes a search bar and navigation links for '詳細へ戻る' (Return to details) and '一覧へ戻る' (Return to list). The '基本情報' (Basic Information) section contains the following settings:

- パーツ属性 (Part Attribute): 記事/天気検索 (Article/Weather Search) - This text is circled in orange, and a '変更する' (Change) button is next to it.
- パーツ名 (Part Name): 天気検索 (Weather Search)
- ファイル名 (File Name): weather_search.part.html
- 携帯向け表示 (Mobile Display): 表示 (Display)
- 動的表示 (Dynamic Display): 無効 (Ineffective)

The '公開設定' (Publication Settings) and '権限' (Permissions) sections are also visible. At the bottom, there are '保存' (Save) and 'キャンセル' (Cancel) buttons.

公開側の開発のその前に

- パーツの管理側にコントローラーとビューは存在しないので、以上でパーツの管理側の作成は完了です。
 - 2章でパーツの管理側にはコントローラーとビューが存在しないことを説明しました。
- パーツの公開側の実装に進みたいと思いますが、ちょっとその前に、シラサギの特徴的なルーティングがパーツにもありますので、ルーティングを説明します。

例: 記事リストパーツの公開画面のルーティング

```
part "article" do
  get "page" => "public#index", cell: "parts/page"
end
```

- “part”というDSLを用いる
- 1+2: article/page という route 属性を持つパーツがアクセスされたら
- 1+4: article/agents/parts/page_controller というコントローラーの
- 3: indexというアクションを実行する

ルーティングの作成

- 今回はarticle/weather_pageという属性をもつパーツを開発したので、次のようなルーティングを作成します。

config/routes/article/routes.rb

```
part "article" do
  get "page" => "public#index", cell: "parts/page"
  get "weather_search" => "public#index", cell: "parts/weather_search"
end
```

- 作成したルーティングはarticle/agents/parts/weather_search_controllerというコントローラーのindexアクションを実行します。

公開側コントローラーの作成

- app/controllers/article/agents/parts/weather_search_controller.rb をテキストエディタで開き、次のような空のアクションをもつコントローラーを作成します。

app/controllers/article/agents/parts/weather_search_controller.rb

```
class Article::Agents::Parts::WeatherSearchController < ApplicationController
  include Cms::PartFilter::View

  def index
  end
end
```

公開側のビューの作成

- 次のような内容をもつビューを `app/views/article/agents/parts/weather_search/index.html.erb` に作成します。

`app/views/article/agents/parts/weather_search/index.html.erb`

```
<%= form_tag @cur_part.parent.url, method: "get" do |f| %>
  <h2>天気</h2>
  <div class="weather">
    <label><%= check_box_tag 'weathers[]', 'sunny' %>晴れ</label>
    <label><%= check_box_tag 'weathers[]', 'cloudy' %>曇り</label>
    <label><%= check_box_tag 'weathers[]', 'rain' %>雨</label>
    <label><%= check_box_tag 'weathers[]', 'snow' %>雪</label>
  </div>
  <footer class="send">
    <%= submit_tag '検索', name: nil %>
  </footer>
<% end %>
```

検索ボタンがクリックされたらパーツの親フォルダーに遷移する

パーツをレイアウトへ組み込み

- 作成したパーツをレイアウトを組み込んでみます。
- 記事一覧のレイアウトは「記事レイアウト」ですので、このレイアウトに作成したパーツを組み込んでみます。



パーツをレイアウトへ組み込み

- 作成したパーツのパスはweather_search/weather_searchでした。
- 以下のように記事レイアウトのサイドバーの先頭に作成したパーツを組み込みます。

記事レイアウト

```
<div id="side">
  <section id="weather">
    <header class="title"><h2>天気検索</h2></header>
    {{ part "weather_search/weather_search" }}
  </section>
  <section id="month">
    <header class="title"><h2>月別ページ一覧</h2></header>
    {{ part "docs/archive/month" }}
  </section>
  <section id="calendar">
    <header class="title"><h2>カレンダー</h2></header>
    {{ part "docs/archive/calendar" }}
  </section>
  {{ part "links-life" }}
</div>
```

確認

- ブラウザをリロードし、記事一覧のプレビューを表示してみましょう。
- 次のように表示されれば完成です。

The screenshot shows the Shirasagi City website interface. At the top, there is a navigation bar with various utility links like 'ご利用案内', 'ふりがなをつける', '読み上げる', and background/color options. Below this is the city logo and a search bar. A main navigation menu contains categories like '暮らし・手続き', '子育て・教育', '健康・福祉', '観光・文化・スポーツ', '産業・仕事', and '市政情報'. The main content area is titled 'HOME > 記事' and features a '記事' (Articles) section with social media sharing buttons and a list of news items. On the right side, there is a '天気検索' (Weather Search) widget with a search box and a '検索' button. An orange callout box labeled '検索パーツ' (Search Part) points to this widget. Below the weather widget is a '月別ページ一覧' (Monthly Page List) showing the current month (December 2016) and previous months with article counts.

まとめ

- 次のファイルを新規作成または修正しました。
 - app/controllers/article/agents/parts/weather_search_controller.rb
 - app/models/article/initializer.rb
 - app/models/article/part.rb
 - app/views/article/agents/parts/weather_search/index.html.erb
 - config/locales/article/ja.yml
 - config/routes/article/routes.rb

- 次のことを学びました。
 - パーツの作成方法
 - パーツとノードの連携

演習

- 次の課題に取り組んでみてください。
 - ビューに日本語を直接書きました。これらをロケールに移動してみてください。
 - 検索条件をクリアする「リセット」を作成してみてください。
 - FAQ/FAQ記事検索パーツでは、検索ノードを指定できるようになっています。FAQ/FAQ記事検索パーツを参考に、天気検索パーツで天気検索ノードを指定できるようにしてみてください。

ハンズオンの成果物

- 本ハンズオンの成果物は<https://github.com/shirasagi/ss-handson>のhandson/weather-search-partブランチにあります。

ハンズオン7 ページの開発




ページの開発


- 本ハンズオンでは独自のページを開発してみます。
- ちょっとその前に、シラサギが標準的に提供している便利な再利用可能なアドオンを紹介します。


アドオンカタログ

メタ情報: Cms::Addon::Meta

メタ情報

キーワード 


概要 

サマリー 


- キーワードと概要は、HTMLの<meta name="keywords">、<meta name="description">を設定
- サマリーはループHTMLの#{summary}を設定

本文: Cms::Addon::Meta

本文



[サンプルファイル \(PDF 783KB\)](#)



- 本文を設定

アドオンカタログ

ファイル: Cms::Addon::File



- 添付ファイルを設定

カテゴリー: Category::Addon::Category



- ページのカテゴリーを設定

アドオンカタログ

ファイル: Cms::Addon::File



- 添付ファイルを設定

カテゴリー: Category::Addon::Category



- ページのカテゴリーを設定

アドオンカタログ

イベント: Event::Addon::Date

イベント

イベントタイトル ?

イベント日 ? -

- イベント日時を設定

Map::Addon::Page

地図



マーカー設定 ?

座標

マーカー名

説明

- 地図上のマーカーを設定
- 設定したマーカーはポップアップします。

アドオンカタログ

関連記事: Cms::Addon::RelatedPage

関連記事

関連記事 ?

関連記事を選択する

タイトル	
世帯または世帯主を変更するとき	削除
住民票記載事項証明書様式	削除
自動交付機・コンビニ交付サービスについて	削除

- 関連ページを設定

連絡先: Contact::Addon::Page

連絡先

表示設定 ?

表示 ▼

所属 ?

連絡先グループを選択する

グループ名

シラサギ市/企画政策部/政策課

削除

担当 ?

電話番号 ?

000-000-0000

ファックス番号 ?

000-000-0000

メールアドレス ?

kikakuseisaku@example.jp

- 連絡先を設定

アドオンカタログ

公開設定: Cms::Addon::Release

承認アドオンを組み込んでいる時

公開設定	
公開日時 ?	2016/12/14 19:41

承認アドオンを組み込んでいない時

公開設定	
ステータス	公開 ▼
公開日時 ?	2016/12/15 05:25

- 公開日時を設定
- 公開状態を設定

公開予約: Cms::Addon::ReleasePlan

公開予約	
公開開始日時(予約) ?	<input type="text"/>
公開終了日時(予約) ?	<input type="text"/>

- 公開開始日時（予約）と公開終了日時（予約）を設定

アドオンカタログ

権限: Cms::Addon::GroupPermission

権限

管理グループ ?

グループを選択する

グループ名	
シラサギ市/企画政策部/政策課	削除
シラサギ市/企画政策部/広報課	削除

権限レベル ?

1 ▼

- 管理グループと権限レベルを設定

承認: Workflow::Addon::Approver

承認

承認申請 ?

自所属 ▼ 選択

- 承認経路を選択し、ページを承認に回す

アドオンカタログ（ノード用）

リスト表示: Event::Addon::PageList

リスト表示

検索条件(URL) ?

並び順 ?

表示件数 ? 100

上部HTML ?

ループHTML ?

```
1 <article class="item-#{class} #{new}">
2 <header>
3 <time datetime="#{date.iso}"#{date.long}</time>
4 <h2><a href="#{url}">#{index_name}</a></h2>
5 </header>
```

下部HTML ?

- リスト表示型のノードに組み込まれる。
- 詳しくは6章を参照

カテゴリー（設定）： Category::Addon::Setting

カテゴリー

カテゴリー設定 ?

カテゴリーを選択する

カテゴリー名

くらしのガイド

削除

- ノードに組み込まれるアドオン
- 配下のページで選択できるカテゴリーを、ここで指定したカテゴリーに限定することが可能となる。
- 無指定（既定値）の場合、サイトに定義した全カテゴリー内から選択可能となる。

ページの開発

- ハンズオンに戻りまして、新規にページを作成してみます。
- 新規ページの仕様
 - 記事/ビデオ（英: article/video）
 - 次のアドオンの組み込み
 - 既存のアドオンから
 - メタ情報
 - 本文
 - 公開設定
 - 公開予約
 - 権限
 - 新規のアドオン
 - YouTubeのビデオIDからYouTube埋め込みプレイヤーを表示する



ページの開発

- ハンズオンに戻りまして、新規にページを作成してみます。
- 新規ページの仕様
 - 記事/ビデオ（属性: article/video_page）
 - 次のアドオンの組み込み
 - 既存のアドオンから
 - メタ情報
 - 本文
 - 公開設定
 - 公開予約
 - 権限
 - 新規のアドオン
 - ビデオアドオン（Article::Addon::Video）：YouTubeのビデオIDからYouTube埋め込みプレイヤーを表示する



ビデオアドオンの開発

- モデルとビューを開発します。

app/models/concerns/article/addon/video.rb

```
module Article::Addon
  module Video
    extend ActiveSupport::Concern
    extend SS::Addon

    included do
      field :video_id, type: String
      permit_params :video_id
    end
  end
end
```

ビデオアドオンの開発

app/views/article/agents/addons/video/_form.html.erb

```
<dl class="see mod-article-video">
  <dt><%= @model.t :video_id %><%= @model.tt :video_id %></dt>
  <dd><%= f.text_field :video_id %></dd>
</dl>
```

app/views/article/agents/addons/video/_show.html.erb

```
<dl class="see mod-article-video">
  <dt><%= @model.t :video_id %></dt>
  <dd><%= @item.video_id %></dd>
</dl>
```

app/views/article/agents/addons/video/view/index.html.erb

```
<% if @cur_page.video_id.present? %>
  <iframe id="ytplayer" type="text/html" width="640" height="360"
    src="http://www.youtube.com/embed/<%= @cur_page.video_id %>"
    frameborder="0"/>
<% end %>
```


ビデオアドオンの開発

config/locales/article/ja.yml

```
modules:  
  article: 記事  
  addons:  
    article/video: ビデオ
```

```
mongoid:  
  models:  
    article/page: 記事ページ  
    article/node/page: 記事リスト  
    article/part/node: 記事リスト  
  attributes:  
    article/addon/video:  
      video_id: ビデオID
```



ビデオページのモデル作成

- ビデオアドオンが開発できたので、ビデオページのアドオンを作成してみましょう。
- 作成するファイルはapp/models/article/video_page.rbです。

app/models/article/video_page.rb

```
class Article::VideoPage
  include Cms::Model::Page
  include Cms::Page::SequencedFilename
  include Cms::Addon::Meta
  include Cms::Addon::Body
  include Article::Addon::Video
  include Cms::Addon::Release
  include Cms::Addon::ReleasePlan
  include Cms::Addon::GroupPermission

  set_permission_name "article_video_pages"

  default_scope ->{ where(route: "article/video_page") }
end
```

Cms::Model::Pageと
Cms::Page::SequencedFilenameは必須

権限の設定 (後述)

ページ属性

ビデオページモデルの権限

- ビデオページモデルのリソース名をset_permission_nameメソッドを利用して“article_video_pages”に設定しました。
- シラサギでは、通常、このリソースに対してread, edit, delete, releaseの4つの権限を管理します。
- そして、シラサギでは一般的に自グループ（private）にread, edit, delete, releaseの権限を与えるかどうかと、他グループ（other）にread, edit, delete, releaseの権限を与えるかどうかを管理します。



ビデオページモデルの権限

- ビデオページの権限テーブルは次のとおりとなります。

リソース名	権限	グループ	権限名
article_video_pages	read	private	read_private_article_video_pages
		other	read_other_article_video_pages
	edit	private	edit_private_article_video_pages
		other	edit_other_article_video_pages
	delete	private	delete_private_article_video_pages
		other	delete_other_article_video_pages
	release	private	release_private_article_video_pages
		other	release_private_article_video_pages

ビデオページモデルの権限

- 先のテーブルの権限名をapp/models/article/initializers.rbで設定します。

app/models/article/initializers.rb

```
Cms::Role.permission :read_private_article_video_pages  
Cms::Role.permission :read_other_article_video_pages  
Cms::Role.permission :edit_private_article_video_pages  
Cms::Role.permission :edit_other_article_video_pages  
Cms::Role.permission :delete_private_article_video_pages  
Cms::Role.permission :delete_other_article_video_pages  
Cms::Role.permission :release_private_article_video_pages  
Cms::Role.permission :release_other_article_video_pages
```

ビデオページのルーティング

- シラサギのルーティングについてこれまでになどか掲載してきましたが、ページのルーティングについても説明します。

例: 記事ページ / ノードの管理画面のルーティング

```
content "article" do
  get "/" => redirect { |p, req| "#{req.path}/pages" }, as: :main
  get "generate" => "generate#index"
  post "generate" => "generate#run"
  resources :pages, concerns: [:deletion, :copy, :move, :lock, :download, :import, :opendata_ref]
end
```

- “content”というDSLを用いてルーティングを定義
- 1行目は“/article/”にアクセスすると“/article/pages”へリダイレクトするルーティング
- 2行目と3行目は書き出し処理（割愛）
- 4行目が記事ページと記事ノードの管理画面のルーティング
 - article/pageという属性のノードまたはページにアクセスされたらという意味です。

お気づきの方もいるかもしれませんが、記事ページと記事ノードの管理画面のルーティングは同じです。

例: 記事ページの公開画面のルーティング

```
page "article" do
  get "page/:filename.:format" => "public#index", cell: "pages/page"
end
```

- “page”というDSLを用いる
- 1+2: article/page という route 属性を持つページがアクセスされたら
- 1+4: article/agents/pages/page_controller.rb というコントローラーの
- 3: indexというアクションを実行する
- ※グレーの箇所は固定で決まっています。

ルーティングの作成

- 作成したページの属性はarticle/video_pageでした。
- このページのルーティングを定義するので、次のようにすれば良さそうです。

管理側のルーティング

```
content "article" do
  get "/" => redirect { |p, req| "#{req.path}/pages" }, as: :main
  get "generate" => "generate#index"
  post "generate" => "generate#run"
  resources :pages, concerns: [:deletion, :copy, :move, :lock, :download, :import, :opendata_ref]
  resources :video_pages, concerns: [:deletion]
end
```

公開側のルーティング

```
page "article" do
  get "page/:filename.:format" => "public#index", cell: "pages/page"
  get "video_page/:filename.:format" => "public#index", cell: "pages/video_page"
end
```

記事ノード/ページのナビ修正

- 本ハンズオンでは記事ノードで記事ページを作成できるようにしてみます。



「承認」の下に「ビデオページ」を追加し、記事ノードでビデオページを作成できるようにしてみます。

記事ノード/ページのナビ修正

- ナビを修正するためapp/views/article/main/_navi.html.erbをテキストエディタで開きます。

記事ノード/ページのナビ修正

app/views/article/main/_navi.html.erb

```
<%= node_navi :article do %>

<h3><%= link_to : "article.page", article_pages_path %></h3>

<ul class="narrow-page">
  <li><%= link_to : "workflow.page.ready", article_index_ready_path %></li>
  <li><%= link_to : "workflow.page.closed", article_index_closed_path %></li>
</ul>

<h3><span><%= t "workflow.pages" %></span></h3>

<ul class="narrow-page">
  <li><%= link_to : "workflow.page.approve", article_index_approve_path %></li>
  <li><%= link_to : "workflow.page.request", article_index_request_path %></li>
</ul>

<h3><%= link_to : "article.video_page", article_video_pages_path %></h3>

<% end %>
```

確認

- ブラウザをリロードしてみましょう。次のように表示されればとりあえずはOKです。

The screenshot shows the SHIRASAGI website management interface. The top navigation bar includes the SHIRASAGI logo, a search icon, and user information: "サイト管理" (Site Management), "グループ" (Group), and "<Development>". On the right, it shows "政策課 システム管理者" (Policy Section System Administrator).

The main content area is titled "記事" (Articles) and contains a list of articles. The left sidebar has a "記事" (Articles) section with sub-items: "記事ページ" (Article Page), "公開待ち" (Waiting for publication), "非公開" (Unpublished), "承認" (Approval), "依頼されたもの" (Requested items), "申請したもの" (Applied items), and "Article.video page". Other sidebar items include "コンテンツ" (Content), "フォルダー" (Folder), "固定ページ" (Fixed page), "パーツ" (Parts), "レイアウト" (Layout), "フォルダー設定" (Folder settings), "ページ取り込み" (Page import), and "フォルダー書き出し" (Folder export).

The article list includes the following items:

- 削除する
- 上の階層へ
- ふれあいフェスティバル
#29 2016/12/15 15:47 page27.html 記事 非公開
- 転居届
#21 2016/12/14 19:41 tenkyo.html 記事 公開中
- 自動交付機・コンビニ交付サービスについて
#20 2016/12/14 19:41 page19.html 記事 公開中
- 住民票コードの変更
#19 2016/12/14 19:41 page18.html 記事 公開中
- 住民票コードとは
#18 2016/12/14 19:41 page17.html 記事 公開中
- 住所変更の証明書について
#17 2016/12/14 19:41 page16.html 記事 公開中
- 住民票記載事項証明書様式
#16 2016/12/14 19:41 page15.html 記事 公開中
- 証明書発行窓口
#15 2016/12/14 19:41 page14.html 記事 公開中

確認

- Article.video pageをクリックしてみます。
- 次のようなエラーが表示されるかと思えます。
- コントローラーが未作成のため発生しているエラーですのでコントローラーを作成します。

Routing Error

uninitialized constant Article::VideoPagesController

Rails.root: /home/vagrant/sample

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

Routes

Routes match in priority from top to bottom

Helper	HTTP Verb	Path
Path / Url		<input type="text" value="Path Match"/>
sns_mypage_path	GET	/mypage(.:format)
sns_logout_path	GET	/mypage/logout(.:format)
sns_login_path	GET POST	/mypage/login(.:format)
sns_remote_login_path	GET POST	/mypage/remote_login(.:format)
sns_login_status_path	GET	/mypage/status(.:format)
sns_auth_token_path	GET	/mypage/auth_token(.:format)
sns_cms_path	GET	/mypage/cms(.:format)

コントローラーの作成

- app/controllers/article/video_pages_controller.rbを作成します。

app/views/article/main/_navi.html.erb

```
class Article::VideoPagesController < ApplicationController
  include Cms::BaseFilter
  include Cms::PageFilter

  model Article::VideoPage

  append_view_path "app/views/cms/pages"
  navi_view "article/main/navi"

  private
  def fix_params
    { cur_user: @cur_user, cur_site: @cur_site, cur_node: @cur_node }
  end
end
```

確認

- ブラウザーをリロードすると、次のような画面が表示されます。
- この画面をよく見ると「新規作成」が見つかりません。
- 何故かと言うと、先ほど article_video_pages というリソースに対して、read, edit, delete, release の権限を作成しました。
- この権限が管理者に割り当てられていないから、新規作成が表示されていません。



権限の付与

- サイト設定の権限/ロールから、管理者の権限を編集してみましょう。

サイト確認 サイトレビュー

詳細へ戻る 一覧へ戻る

サイト設定

- サイト情報
- グループ
- ユーザー
- 権限/ロール**
- ワークフロー
- メンバー
- お知らせ
- テンプレート
- Theme切り替え
- ソースクリーニング
- 本文レイアウト
- ページ検索
- 郵便番号
- かな辞書
- 組織変更
- LDAP
- リンクチェック
- 読み上げ音声
- RDF語彙
- ジョブ
- 操作履歴

ロール名 ? 管理者

権限レベル 3 ▼

権限設定

[広告管理]

<input checked="" type="checkbox"/> バナーの削除 (全て)	<input checked="" type="checkbox"/> バナーの編集 (全て)	<input checked="" type="checkbox"/> バナーの閲覧 (全て)
<input checked="" type="checkbox"/> バナーの削除 (所有)	<input checked="" type="checkbox"/> バナーの編集 (所有)	<input checked="" type="checkbox"/> バナーの閲覧 (所有)

[記事]

<input checked="" type="checkbox"/> ページの承認 (全て)	<input checked="" type="checkbox"/> ページの削除 (全て)	<input checked="" type="checkbox"/> ページの編集 (全て)
<input checked="" type="checkbox"/> ページの移動 (全て)	<input checked="" type="checkbox"/> ページの閲覧 (全て)	<input checked="" type="checkbox"/> ページの公開 (全て)
<input checked="" type="checkbox"/> ページのロック解除 (全て)	<input checked="" type="checkbox"/> ページの承認 (所有)	<input checked="" type="checkbox"/> ページの削除 (所有)
<input checked="" type="checkbox"/> ページの編集 (所有)	<input checked="" type="checkbox"/> ページの移動 (所有)	<input checked="" type="checkbox"/> ページの閲覧 (所有)
<input checked="" type="checkbox"/> ページの公開 (所有)		

Cms role.delete other article video pages Cms role.edit other article video pages Cms role.read other article video pages

Cms role.release other article video pages Cms role.delete private article video pages Cms role.edit private article video pages

Cms role.read private article video pages Cms role.release private article video pages

[掲示板]

<input checked="" type="checkbox"/> 安否の削除	<input checked="" type="checkbox"/> 安否の編集	<input checked="" type="checkbox"/> 安否の閲覧
<input checked="" type="checkbox"/> 投稿の削除	<input checked="" type="checkbox"/> 投稿の編集	<input checked="" type="checkbox"/> 投稿の閲覧

ここら辺の権限すべてに
チェックをいれ保存します。

確認

- 権限を割り当ててやると「新規作成」が表示されました。

The screenshot shows the SHIRASAGI CMS interface. At the top, there is a navigation bar with the SHIRASAGI logo, a gear icon for 'サイト管理' (Site Management), a group icon for 'グループ <Development>', and a user profile icon for '政策課 システム管理者'. Below the navigation bar, there is a sidebar on the left with a search icon and a list of article categories: '記事', '記事ページ', '公開待ち', '非公開', '承認', '依頼されたもの', '申請したもの', and 'Article.video page'. The main content area is titled '記事' (Articles) and features a '新規作成' (New Article) button with a plus icon. Below this button, there is a search bar with a '削除する' (Delete) button and a '検索' (Search) button. At the bottom of the main content area, there is a link labeled '上の階層へ' (Go to the previous level).

ビデオページの作成

- 新規作成をクリックし、ビデオページを作成してみます。
- 右のようにタイトル、本文、ビデオIDの3箇所になにか入力し、保存してください。

The screenshot shows the SHIRASAGI CMS interface for creating a video page. The interface is in Japanese and shows various input fields and sections. Three specific areas are circled in orange:

- Title:** The title field contains the text "ウィル・スティーヴン「頭良さそうにTED風プレゼンをする方法」".
- Main Text:** The main text area contains the text "ウィル・スティーヴン「頭良さそうにTED風プレゼンをする方法」です。".
- Video ID:** The video ID field contains the text "ToJD5r2SmwI".

Other visible elements include a sidebar with navigation options like "記事ページ", "公開待ち", "非公開", "承認", "依頼されたもの", "申請したもの", "Article.video page", "コンテンツ", "フォルダー", "固定ページ", "パーツ", "レイアウト", "フォルダー設定", "ページ取り込み", "フォルダー書き出し", "ページ書き出し". The top navigation bar shows "SHIRASAGI", "サイト管理", "グループ", and "<Development>". The bottom right has "保存" (Save) and "キャンセル" (Cancel) buttons.

ビデオページ作成エラー

- 保存をクリックすると次のようなエラーが表示されます。
- 公開側のコントローラーが存在しないため、エラーとなっています。
- ページは保存するとHTMLに書き出されます。

NameError in Article::VideoPagesController#create

uninitialized constant Article::Agents::Pages::VideoPageController

Extracted source (around line #6):

```
4 def initialize(controller)
5   if controller.is_a?(String)
6     controller = "#{controller}_controller".camelize.constantize
7   end
8   @controller = controller.new
9   @controller.params = ActionController::Parameters.new
```

Rails.root: /home/vagrant/sample

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

```
lib/ss/agent.rb:6:in `initialize'
app/controllers/application_controller.rb:16:in `new'
app/controllers/application_controller.rb:16:in `new_agent'
app/controllers/concerns/cms/public_filter/page.rb:20:in `render_page'
app/controllers/concerns/cms/public_filter/page.rb:41:in `generate_page'
app/models/concerns/cms/model/page.rb:41:in `block in generate_file'
app/models/concerns/cms/model/page.rb:40:in `generate_file'
app/controllers/concerns/cms/page_filter.rb:50:in `create'
```



ビデオページ作成エラー

- ページは保存するとHTMLに書き出されます。
- HTMLに書き出す際、公開側のコントローラーが呼ばれます。
- しかし、まだ公開側のコントローラーを作成していないので、エラーが発生しています。

公開側コントローラーの作成

- 公開側のコントローラーを作成してみましょう。
- `app/controllers/article/agents/pages/video_page_controller.rb`を新規作成します。

`app/controllers/article/agents/pages/video_page_controller.rb`

```
class Article::Agents::Pages::VideoPageController < ApplicationController
  include Cms::PageFilter::View
end
```

公開側ビューの作成

- コントローラーに続けてビューも作成してしまいましょう。

app/controllers/article/agents/pages/video_page_controller.rb

```
<header class="released">
  <time datetime="<%= I18n.l @cur_page.date.to_date, format: :iso %>">
    <%= I18n.l @cur_page.date.to_date, format: :long %>
  </time>
</header>

<%= render file: "cms/agents/pages/page/index" %>
```

確認

- ブラウザをリロードすると、今度は保存に成功しました。
- 公開側の画面を確認するため、PCプレビューをクリックしてみましょう。次のような画面が表示されれば成功です。

The screenshot shows the Shirasagi City website interface. At the top left is the Shirasagi City logo and name. To the right are links for 'スマホ・携帯サイト', 'お問い合わせ', and 'サイトマップ'. Below these is a search bar labeled 'サイト内検索' with a '検索' button. A navigation bar contains links for '暮らし・手続き', '子育て・教育', '健康・福祉', '観光・文化・スポーツ', '産業・仕事', and '市政情報'. The main content area shows a breadcrumb trail: 'HOME > 記事 > ウィル・スティーヴン 「頭良さそうにTED風プレゼンをする方法」'. Below this is a blue header for the article: 'ウィル・スティーヴン 「頭良さそうにTED風プレゼンをする方法」'. There are social sharing buttons for 'いいね! 0', 'シェア 0', 'ツイート', 'ブックマーク 0', and 'G+ 0', along with a 'CLIP' button. The date '2016年12月15日' is displayed. A short description follows: 'ウィル・スティーヴン 「頭良さそうにTED風プレゼンをする方法」です。'. The video player shows a man on a stage with a large 'TED New York' sign in the foreground. The video title is 'ウィル・スティーヴン「頭良さそうにTED風プレゼンをする方法」'.

日本語化

- ナビと権限に英語が表示されていました。日本語化してみます。
- `config/locales/article/ja.yml`をテキストエディタで開きます。

ナビの日本語化

```
ja:  
  article:  
    page: 記事ページ  
    video_page: ビデオページ
```

権限の日本語化

```
read_private_article_video_pages: ビデオページの閲覧 (所有)  
read_other_article_video_pages: ビデオページの閲覧 (全て)  
edit_private_article_video_pages: ビデオページの編集 (所有)  
edit_other_article_video_pages: ビデオページの編集 (全て)  
delete_private_article_video_pages: ビデオページの削除 (所有)  
delete_other_article_video_pages: ビデオページの削除 (全て)  
release_private_article_video_pages: ビデオページの公開 (所有)  
release_other_article_video_pages: ビデオページの公開 (全て)
```

日本語化

- 日本語化されているかどうかブラウザをリロードして確認してみましょう。

The screenshot shows the SHIRASAGI website management interface. The top navigation bar includes 'SHIRASAGI', 'サイト管理' (Site Management), 'グループ' (Group), and '<Development>'. The user is identified as '政策課 システム管理者' (Policy Department System Administrator). The main content area is titled '記事' (Articles) and shows a list of actions: '編集する' (Edit), '削除する' (Delete), and '一覧へ戻る' (Return to List). The article being viewed is titled 'ウィル・スティーヴン 「頭良さそうにTED風プレゼンをする方法」' (Will Stevinson 'How to Give a TED-style Presentation that Looks Like You're Smart'). The file name is '64.html'. The interface is in Japanese, indicating successful localization.

まとめ

- 次のファイルを新規作成または修正しました。
 - app/controllers/article/agents/pages/video_page_controller.rb
 - app/controllers/article/video_pages_controller.rb
 - app/models/article/initializer.rb
 - app/models/article/video_page.rb
 - app/models/concerns/article/addon/video.rb
 - app/views/article/agents/addons/video/_form.html.erb
 - app/views/article/agents/addons/video/_show.html.erb
 - app/views/article/agents/addons/video/view/index.html.erb
 - app/views/article/agents/pages/video_page/index.html.erb
 - app/views/article/main/_navi.html.erb
 - config/locales/article/ja.yml
 - config/routes/article/routes.rb
- 次のことを学びました。
 - ページの作成方法
 - ページの権限の作成方法



演習

- 次の課題に取り組んでみてください。
 - ビデオアドオンを拡張し、ビデオの横と縦の大きさを設定できるようにしてみてください。
 - 関連ページアドオンやカテゴリーアドオンをビデオページに組み込んでみましょう。

ハンズオンの成果物

- 本ハンズオンの成果物は<https://github.com/shirasagi/ss-handson>のhandson/video-pageブランチにあります。

