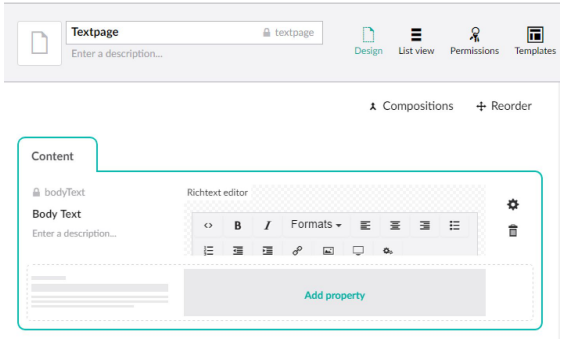
Documenttype and Field naming

A document type defines the Model of a page, a template uses the model to mix page data with html from the template - so a Model tells Umbraco what kind of data is available for this specific type of page.

Generated model conventions

The document type alias, and the alias for each field on the document type is used to generate a strongly typed object

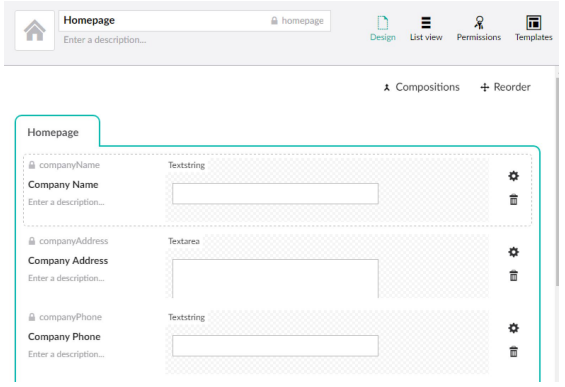



```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage<ContentModels.Textpage>
@using ContentModels = Umbraco.Web.PublishedContentModels;


<h1>@Model.Content.Name</h1>
```

Document properties and models

Each property you add to a document type, is exposed on the Model, so when Umbraco knows the model, it knows what data is available - in this case, `CompanyName` and `CompanyAddress`:




```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage<ContentModels.Homepage>
@using ContentModels = Umbraco.Web.PublishedContentModels;


<h1>@Model.Content.Name</h1>

<p>@Model.Content.CompanyName</p>

<p>@Model.Content.CompanyAddress</p>
```

Anatomy of a Umbraco Template

A Umbraco template uses a Model to interact with the data of the page being rendered. This tells Umbraco what kind of data it can display and how it should display it.

The template declares that it is used with pages of the type “**Textpage**”

```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage<ContentModels.Textpage>
@using ContentModels = Umbraco.Web.PublishedContentModels;
```

The template declares that it inherits html from the master template file “**Master.cshtml**”

```
@{
    Layout = "Master.cshtml";
}
```

Template displays standard values from the Model, properties such as `Name, Id, CreateDate` and `Url` is on all types of pages.

```
<a href="@Model.Content.Url">
    <h1>
        @Model.Content.Name
    </h1>
</a>
```

The template displays simple values from custom properties defined on the document type.

```
<p>@Model.Content.CompanyName</p>
<p>@Model.Content.CompanyAddress</p>
```

The template displays complex values from custom properties

Here a collection of picked content is iterated over from the property “pickedContent”.

```
@*Iterating through a collection of picked content from the property
“pickedContent” *@

<ul>
      @foreach(var item in Model.Content.PickedContent){
            <li>
                  <a href="@item.Url">@item.Name</a>
            </li>
      }
</ul>
```

A cropped image Url is displayed from the property “headerImage”

```
@*Getting the url of a cropped image stored as “headerImage” *@
<img src="@Model.Content.GetCropUrl("headerImage", "header")" />
```

Model typing - <Type> syntax

To have an efficient and concise template, you can in many cases benefit by telling Umbraco what type of content it is processing. Razor has full support for checking types and filtering content by type, the most common scenarios are below:

Specify that the type of Model on the page is of type **Textpage**.

```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage<ContentModels.Textpage>
@using ContentModels = Umbraco.Web.PublishedContentModels;

@Model.Content.BodyText
```

Get the first child of type **NewsArea** below the current page

```
var newsArea = Model.Content.FirstChild<NewsArea>();

<a href="@newsArea.Url">@newsArea.Name</a>
```

Get all Children of type **Textpage** below the current page

```
var textpages = Model.Content.Children<Textpage>();
```

Get a media item of type Image from a mediapicker

```
var img = Model.Content.HeaderImage.OfType<Image>();

<img src="@img.Url" />
```

Specify that the picked content in a contentpicker is of type **NewsArticle**.

```
@var articles = Model.Content.PickedContent.OfType<NewsArticle>();

@foreach(var art in articles){
     <h1>@art.Summary</h1>
}
```

Check if the Current Page uses a **Header** composition - notice the naming convention **IHeader** used here.

```
@if(Model.Content is IHeader){
    <h1>I have header content</h1>
}
```

Composition types can be shared between several pages. You can cast the model of a partial view to a Composition type - valid on all pages using the Header composition type.

```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage<IHeader>

@Model.Content.HeaderImage
```

Razor Query Axes

A query axis is a collection of selected content, relative to the current page, use these to query for sets of content of lists, navigations, breadcrumps and so on.

`Model.Content.Children()`

Returns the pages immediately below the current page.

Usually used for basic lists and sub-navigation

`Model.Content.Ancestors()`

Returns all the pages above the current page

Usually used for breadcrumbs

`@Model.Content.Descendants()`

Returns all the pages below the current page

Very rarely used, as it returns the entire structure of a site in one big collection.

`Model.Content.Site()`

Returns the single page above the current page at level 1

Usually used to find the root for static navigation and lists such as a top navigation

`Model.Content.Parent`

Returns the single page immediately above the current page