uPersonal

uPersonal is an on-site realtime personalization package for Umbraco 7 and Umbraco 8.

Your website visitors build a user profile by simply browsing your website. The package provides tools to adapt the content based on their user profile.

Features

- Content recommendation
- Section in Umbraco backoffice for editors

Requirements

- Umbraco 7.1.1+ & Umbraco 8.1
- Entity Framework 6
- SQL Server database

Getting started

- Get your uPersonal Licence at https://upersonal.nl/licentie-aanvraag/ Licenses are free for local deployments that have hostname localhost or *.local.
- Setup Umbraco
    - Make sure you select SQL Server as database backend
- Install *Entity Framework* using NuGet
- Launch Umbraco (CTRL+F5) and install *uPersonal-*.zip* using the Umbraco package manager. In the Umbraco backoffice, go to Developer -> Packages and press Install local.

Now we have the basic setup for uPersonal. Let's create a document type for our demo.

- Go to Settings -> Document Types -> Create Document Type
    - Name: Demo Page
- Go to Settings -> Templates -> Demo Page. Replace the Razor template with the following code:

```
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage<ContentModels.DemoPage>
@using ContentModels = Umbraco.Web.PublishedContentModels;
@{
    Layout = null;
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
}
@if (up.IsInSegment("Developer"))
{
    <p>Hello World!</p>
} else {
    <p>Refresh</p>
}
```

We created a template that will show the text *Hello World!* if you are a developer. Next we will configure uPersonal to recognize you when you are a developer.

- Go to uPersonal -> Model -> Segments -> Create segment
    - Segment name: Developer
    - Segment lifetime: 30 hours
- Go to uPersonal -> Logic -> Actions -> New Actions
    - Action name: Set segment - Developer
    - Action type: uPersonal -> Set Segment
    - Segment: Developer
- Go to uPersonal -> Logic -> Triggers -> New Trigger
    - Trigger name: Trigger - Developer
    - Add trigger rule
        - Type: Page visit -> Page of document type
        - Content Type: Demo Page
        - Minimum hits: 3
    - Add trigger action
        - Set segment - Developer

We created the Developer segment and have it trigger after visiting a Demo Page document type 3 times. Let's try it out!

- Go to Content -> Create -> Demo Page
    - Name: Demo
- Click the link for the Demo page

You should see some content. Refresh the page 2 more times and you should see the text *Hello World!* Congratulations, you triggered the rule to put yourself in the developer segment.

Basic concepts

uPersonal is an on-site realtime personalization package for Umbraco. uPersonal evaluates page requests and collects information of each user in a profile. This enables you to setup elaborate personalization strategies covering multiple visits and customized triggers.

Content personalization decisions are made using the profile elements: segments and attributes.

Segments

Segments are elements of a user profile you turn on or off. Use segments to target specific content to users.

You have to create segments before you can use them. Segments have the following properties:

- **Segment name:** Reference for the segment throughout the code and the backoffice.
- **Segment lifetime:** Number of hours the segment is active after it is enabled in the user profile.

Show or hide content in code directly using segments with the helper method `IsInSegment(string segmentName)`. For example:

```
var up = uPersonal.Umbraco.uPersonalHelper.Current;
@if (up.IsInSegment("Developers"))
{
    <p>Hello world!</p>
}
```

Or use *content selectors* to enable editors to link content to segments dynamically. In code you call the content selector using `SelectSingleNode(string selectorName)`. The result is an IPublishedContent determined by the the editor that linked a node to the segment of the current user.

Render the IPublishedContent anyway you like:

```
var up = uPersonal.Umbraco.uPersonalHelper.Current;
var personalisedContent = up.SelectSingleNode("uniqueSellingPoint");
@Html.RenderPartial("~/Views/Partials/_banner.cshtml", personalisedContent);
```

Create *triggers* to set and remove segments in a profile or manually manage segments using the helper methods `PutInSegment(string segmentName)` and `RemoveFromSegment(string segmentName)`. This allows you to create custom triggers.

Attributes

Attributes are elements of a profile with a score. The score is determined by page hits and other triggers. Use attributes to rank content based on relevance to the current user profile.

Create profile attributes in the backoffice in uPersonal -> Model -> Profile attributes. They have the following properties:

- **Attribute name:** Reference for the attribute
- **Components:** One or more named components

Ranking content

uPersonal sorts content by relevance to the current user. To determine relevance profile attributes must be defined for the document type. Add the property editor `uPersonal.AttributeComponentsEditor` to the document type and select what attributes are relevant to the document. For each content item the attribute components must be set.

Each page hit uPersonal adjusts the users' profile attributes to be slightly more similar to the page's attributes. The users' profile attributes accumulate over multiple page requests and multiple sessions in order to approximate the attributes relevant to the current user.

Call `GetNodeScore(INode node)` to get a normalized score indicating relevance of the provided node to the current user.

Use *content selectors* to list nodes in order of relevance to the current user. The editor determines the nodes to be included in the list. Call `SelectNodes(string contentSelectorAlias)` to get a sorted list of content.

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
    var shows = up.SelectNodes("recommendedShows");
}

<h1>Your recommended shows</h1>
<ul>
foreach (var show in shows)
{
    <li>@show.Url</li>
}
</ul>
```

Attributes on content nodes serve two purposes. They influence the profile of the user visiting the content element and they allow uPersonal to sort content based on relevance.

Content selectors

Content selectors allow an editor in the backoffice to determine what nodes to show .

- **Selector name:** Display name
- **Selector alias:** Selector reference to use in code
- **Selector type:** Based on presence in segment or based on profile compatibility

Content selectors **based on presence in segment** have the additional properties:

- **Selector rules:** Link segments with content nodes
- **Default node:** Node returned in absence matching segment

Content selectors **based on profile compatibility** have the additional properties:

- **Parent node:** Parent node of nodes to be sorted for compatibility
- **XPath query:** Select nodes to be sorted for compatibility, required in absence of parent node
- **Profile attribute weight:** Relative weight of attributes
- **Max. nodes to return:** Maximum nodes the `SelectNodes` for this content selector returns

Use `SelectNodes(string contentSelectorAlias)` for content selector based on profile compatibility and `SelectSingleNode(string contentSelectorAlias)` for content selectors based on presence in segment or based on profile compatibility to fetch the top result.

Triggers

The following triggers are available.

- Page visit
    - Visitor returning
    - Specific page
    - Page of document type
    - Page with property
- Browser and platform
    - Platform
    - Browser
    - Regular expressions
    - Browser language preference

- Visit source
    - Referring site
- Url and Querystring
    - Url
    - Querystring

Configuration

License

Local deployments that have hostname `localhost` or `*.local` do not require a paid license. A paid license is required for production deployments or tracking will be disabled. Place your uPersonal.lic file in the root of the website.

The status of the license can be checked in backoffice under uPersonal -> Settings -> License configuration.

Separate database

Optionally specify a separate database for uPersonal. Add the connection string *uPersonalDbDSN* to your Web.config. By default uPersonal uses the Umbraco database.

```
<configuration>
  <connectionStrings>
    ...
    <add name="uPersonalDbDSN" connectionString="myconnectionstring"
providerName="System.Data.SqlClient" />
  </connectionStrings>
  ...
</configuration>
```

routes starting with /umbraco/ are ignored

API reference

uPersonal Helper

CurrentVisitorProfile() : XmlDocument

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
}
@up.CurrentVisitorProfile().innerXml;
```

```
<?xml version="1.0" encoding="utf-16"?>
<VisitorProfile Id="3" HasProfile="true">
  <Segment ID="1" Name="Foo" IsInSegment="False"></Segment>
  <ProfileAttribute ID="1" Name="New Profile Attribute">
    <Component ID="1" Name="Component A" Score="0" Percentage="0.0"></Component>
    <Component ID="2" Name="Component B" Score="0" Percentage="0.0"></Component>
  </ProfileAttribute>
</VisitorProfile>
```

CurrentVisitorSegments() : List

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
    var segments = up.CurrentVisitorSegments();
}
<ul>
@foreach (var segment in segments)
{
    <li>@segment</li>
}
</ul>
```

IsInSegment(string segmentName) : bool

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
}
@if (up.IsInSegment("Developers"))
{
    <p>Hello world!</p>
}
```

PutInSegment(string segmentName) : void

RemoveFromSegment(string segmentName) : void

```
class MyController : Umbraco.Web.Mvc.SurfaceController
{
    public ActionResult DoSomething()
    {
        var up = uPersonal.Umbraco.uPersonalHelper.Current;
        up.PutInSegment("Developers");
        // or up.RemoveFromSegment("Developers");

        return Content("OK!");
    }
}
```

AddComponentScore(int componentId, int score) : void

```
class MyController : Umbraco.Web.Mvc.SurfaceController
{
    public ActionResult DoSomething()
    {
        var up = uPersonal.Umbraco.uPersonalHelper.Current;
        up.AddComponentScore(1, 50);

        return Content("OK!");
    }
}
```

GetNodeScore(INode node) : float

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
    var currentNode = Umbraco.TypedContent(Node.getCurrentNodeId());
}

Score for this node: @(up.GetNodeScore(currentNode))
```

SelectNodes(string contentSelectorAlias) : IEnumerable

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
    var shows = up.SelectNodes("recommendedShows");
}
<h1>Your recommended shows</h1>
<ul>
foreach (var show in shows)
{
    <li>@show.Url</li>
}
</ul>
```

SelectSingleNode(string contentSelectorAlias) : IPublishedContent

```
@{
    var up = uPersonal.Umbraco.uPersonalHelper.Current;
    var uniqueSellingPoint = up.SelectSingleNode("uniqueSellingPoint");
}
@Html.Partial("~/Views/Partials/_banner.cshtml", uniqueSellingPoint);
```

License

uPersonal is available under a commercial license. Please inquire uPersonal for more information.

Changelog

0.2.0 compatible with Umbraco 8.1

0.1.0 initial release