FirstSpirit™
Unlock Your Content

RealtimeTargeting
FirstSpirit Version 5.x

Version	**1.1**
Status	**RELEASED**
Date	**2015-01-20**
Department	Product Management
Copyright	2014 e-Spirit AG

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-spirit.com
www.e-spirit.com

e-Spirit

Table of content

1 Introduction

The FirstSpirit *Real-time Targeting* module provides for the custom delivery of FirstSpirit content on websites to specific target groups and to individual website visitors. FirstSpirit makes it possible for each visitor to have the information corresponding to his or her needs provided at the right time on the suitable output device.

The result is that the information offered is more relevant to the website visitor. This leads to:

- higher customer satisfaction,
- long-term customer loyalty,
- higher conversions, and
- an optimized user experience.

The following conditions form the basis for website content delivery that is specific to the target group(s):

1. **Target group definition**

   When implementing a project, the first thing is to define conceptually the target groups that are relevant to the specific website. This classification is always made on a case-by-case basis in each project and for each customer. The target groups are often also referred to as segments or (visitor) groups.

2. **Target group identification**

   After defining the target groups, it is important to establish how to determine and identify which website visitors fall into which target group. The rules required for this depend significantly on the types of target groups.

   Here are a few examples:

   a. *Mobile user vs. desktop user*

      Since a visitor's browser provides this information directly, the information is very easy to identify.

b. Potential customer (guest) vs. existing customer

This information can be determined in various ways with technology. For instance, it can be determined using a login area on the website.

c. Potential customers interested in a particular topic

The interests of a website visitor for a specific topic can be identified explicitly or implicitly.

In the case of explicit assignment, the visitor is asked directly if a topic interests him or her. This information is stored in the visitor profile.

Implicit assignment usually works with a "scoring" mechanism. This is used to establish what types of articles the visitor most frequently views. From a specific threshold, it is possible to assign the visitor to a target group.

To identify the target groups, a profile for each website visitor is always required in order to have a lot of flexibility when creating target group rules.

3. Maintaining the content for different target groups

To begin with, the editorial maintenance of various content and assignment of this content to the different target groups (known as segments) provide the best experience for the website visitor. Both can be accomplished using the *Real-time Targeting* module directly in the FirstSpirit user interface. In this case, FirstSpirit has to identify the list of defined segments in order to be able to provide the editor with the segments when maintaining the specific content.

4. Delivery of content specific to target groups

Delivering the appropriate content for website visitors always takes place in real time, since segment assignment can change for each page as required. The *Real-time Targeting* module checks the following when delivering content:

a. The segments to which the visitor is currently assigned

b. The page content that is relevant to this segment

In addition to delivering content specific to target groups, the *Real-time Targeting* module supports, for instance, the ability to cater to individuals by accessing individual profile data (volume of business in the last month, last login, etc.).

The technical details of the interplay of the described conditions are explained in the architecture overview presented in the next section.

1.1 Architecture overview

The architecture of the *Real-time Targeting* module comprises four elements that interact with each other (*see Figure 1-1*):

The interaction of the individual elements is covered in more detail in the following sections.

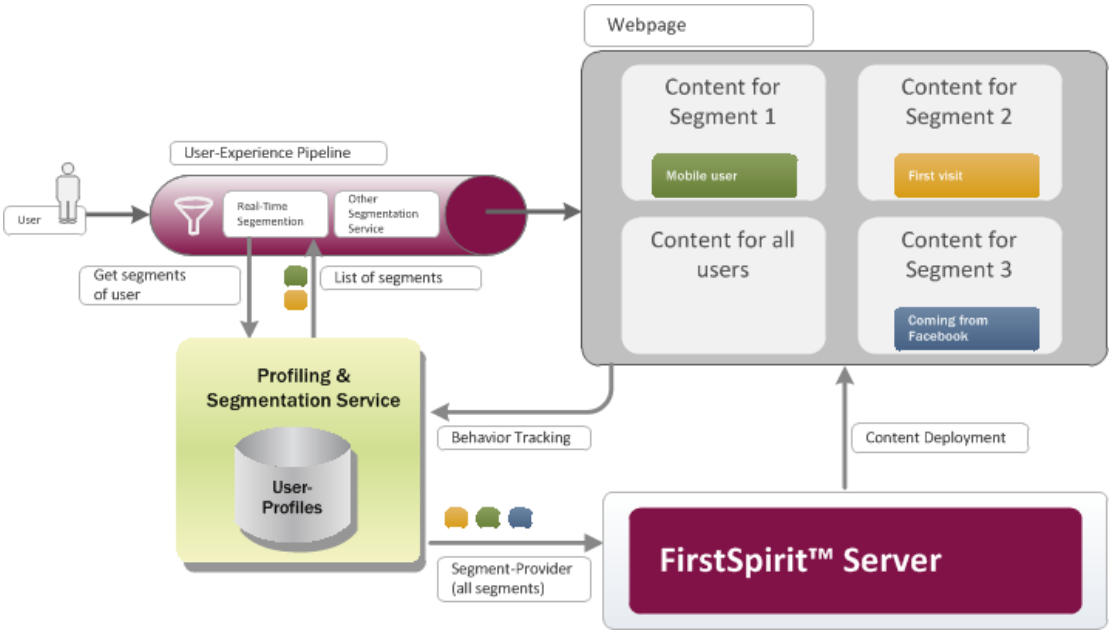

Figure 1-1: Architecture

1.1.1 User Experience Pipeline

The User Experience Pipeline (UXP) runs every time a page is called by a website visitor. Its purpose is to analyze all of the target group information or segments that contain the current visitor. 1 to n Profiling & Segmentation Services are called for this purpose (*see section 1.1.2, page 8*).

By default, the *Real-time Targeting* module uses just such a service (*Woopra™*). If other or additional services are required, these can be added to UXP at any time for the project. The module at this location has a completely open architecture and is not limited to a specific service.

1.1.2 Profiling & Segmentation Service

This service handles three different tasks:

- **Visitor tracking and storing visitor profiles**

  Tracking logic on the website or in a mobile app is used to save all user interactions and to compile them into a profile. This persistent storage is used to provide a website visitor's complete history. The history can in turn be used to define segments (see the point that follows).

  Usually, in addition to standard activities (viewing a particular web page, downloading a specific file), project-specific events are transferred from the web page to the visitor profile (visitor purchased a specific product, order process was canceled by visitor, etc.).

- **Defining rules for segments**

  Based on the visitor profiles, rules can be used to define the segments to which the visitors are assigned. The Profiling & Segmentation Service typically has a graphical user interface for defining the rules. Creation and modification of rules can be done on a one-time basis when the project is set up as well as ad hoc at any time.

  The service must be able to provide the list of all existing segments to FirstSpirit using the API so that it will be available to the editors for allocation of content.

- **Real-time delivery of a visitor's segments**

  A key task of the Profiling & Segmentation Service is to provide the segment list associated with an identified website visitor. This interface must be real-time capable so that there is no delay in the downstream website delivery. The website visitor is usually identified by an ID (e.g. a cookie).

The FirstSpirit *Real-time Targeting* module supports the connection to the Woopra™ service by default (https://www.woopra.com). Woopra™ offers the three interfaces shown and contains turnkey implementations ready for use. Woopra™ also features turnkey CRM system integration (http://salesforce.com), offering additional useful application scenarios.

However, as previously mentioned, other services can also be implemented as an alternative.

1.1.3 Content creation and assignment in FirstSpirit

The task of the editors is to create specific content geared toward communication with target groups. For this purpose, a generally applicable version of content is usually created first and then a specific variant is developed which is built upon this version. This type of variant is then assigned to one or more target groups or segments.

A web page thus contains (from a physical perspective) all content variants. At the time of delivery, however, only the relevant content is provided (*see section 1.1.4, page 9*).

1.1.4 Content delivery

After analyzing the segment list associated with a visitor, the web server run-time environment can filter and output the "relevant" web page content. Only the correct variants as determined by editor parameterization are displayed. Irrelevant content is not delivered.

Technically speaking, a Java Tag Library is used for this purpose. It compares the segments analyzed in the UXP with the editor's page configuration and supplies the relevant content.

1.2 Technical requirements

The *Real-time Targeting* module has the following technical requirements:

- FirstSpirit version: FirstSpirit 5.1 (or higher)
- Run-time environment: Tomcat 7 (or web server with a servlet engine, version 3.0 or higher, and JSTL version 1.0 or higher)
- Use of Profiling & Segmentation Service: Woopra™ (www.woopra.com)
- JavaScript library: jQuery 1.9.1 (or higher)

*The Real-time Targeting module actively supports only Tomcat 7 as the run-time environment at this time. Use of other web servers has **not** been tested and is therefore not guaranteed to work.*

By default, the Real-time Targeting module uses the Woopra™ service previously mentioned. However, if the use of other or additional services is desired, these services can be added to a specific UXP project at any time (see section 1.1.1, page 8). Chapter 6 describes the steps required to add these services.

1.3 Data protection and privacy

As previously described in the introduction, the *Real-time Targeting* module supports delivery of specific website content. The visitors are assigned to predefined target groups based on their interactions with the website.

We expressly inform that for target group identification personal data is collected, used and transmitted to the Profiling & Segmentation Service.

The Woopra™ standard tracking mechanism collects the following data:

- Browser type, domain name and IP address
- End device ID and operating system

- Referrer as well as the visitor's time of access and total time spent on the site
- Web pages visited and links clicked
- All transmitted data

In addition, a cookie is used which is saved on the website visitor's end device. The cookie uses a unique ID to identify the particular user and is valid for two years from the last website interaction.

These conditions are relevant to data protection and privacy laws and must therefore be addressed specifically for each project.

If additional information in the project is sent to Woopra™ or another Profiling & Segmentation Service is used, additions or adjustments must be made to the mentioned list accordingly.

The Real-time Targeting module only allows for the identification, analysis and use of the specific interests of the website visitors. Data protection and privacy must therefore be addressed specifically for each project.

1.4 Topics covered in this documentation

This document describes the functions of the real-time capture of website visitor behavior and covers the components required to do this as well as how to install and configure them:

Chapter 2: This chapter describes the components required.

Chapter 3: This chapter covers how to install and configure the components on the FirstSpirit server.

Chapter 4: The adjustments that need to be made in your FirstSpirit projects are described in this chapter.

Chapter 5: This chapter provides a few application scenarios for using the *Real-time Targeting* module.

Chapter 6: The steps for expanding the module or replacing the Profiling & Segmentation Services used are contained in this chapter.

2 Components

The FirstSpirit *Real-time Targeting* module consists of various components:

- FirstSpirit module see section 2.1, page 12
- FirstSpirit components see section 2.2, page 12
- Persona Simulator see section 2.3, page 13
- Profiling & Segmentation Service see section 2.4, page 14

The individual components are presented in the sections that follow.

2.1 FirstSpirit module

The *Real-time Targeting* module is the component that needs to be installed on the FirstSpirit server.

Using the functions provided by the module on the FirstSpirit server, the segments defined within the Profiling & Segmentation Service (*see section 2.4, page 14*) are transferred to your FirstSpirit project and have to be provided to the editor there in order to allow for target group specific assignment of the project content. This assignment is evaluated automatically during output of the content.

In addition, a function for checking the assignment made is provided to the editors in the form of the Persona Simulator (*see 2.3, page 13*).

2.2 FirstSpirit components

The different FirstSpirit components act like a link between FirstSpirit and the Profiling & Segmentation Service used (*also refer to section 2.4, page 14*).

There are *two project components*: While the first component triggers the template import to your project, the account data of the Profiling & Segmentation Service is managed in the second component. With these, the module establishes a project-specific connection between the FirstSpirit server and the service.

The *web components* control the output of target group specific content generated by the editors in the preview, in ContentCreator and on the website (*also refer to section 2.3, page 13*).

In this way, the functions of the two systems are combined and put to their best use.

2.3 Persona Simulator

The *Persona Simulator* is used to provide the editors a function in the preview and in ContentCreator which allows them to take on a persona whose association with particular target groups is specifically defined (*also refer to section 4.3, page 36*). The editors can thus view a web page as seen by the specific target group(s) and can use this to check the appearance of the content variants they created. Without the *Persona Simulator*, the editors would only be able to view the general content and would only be able to perform the check on the live website.

The *Persona Simulator* is based on the Multi Perspective Preview (*see Roadmap 2013-2017 & the FirstSpirit 5.1 release notes*). This allows it to be seamlessly integrated into ContentCreator (*see Figure 2-1*).

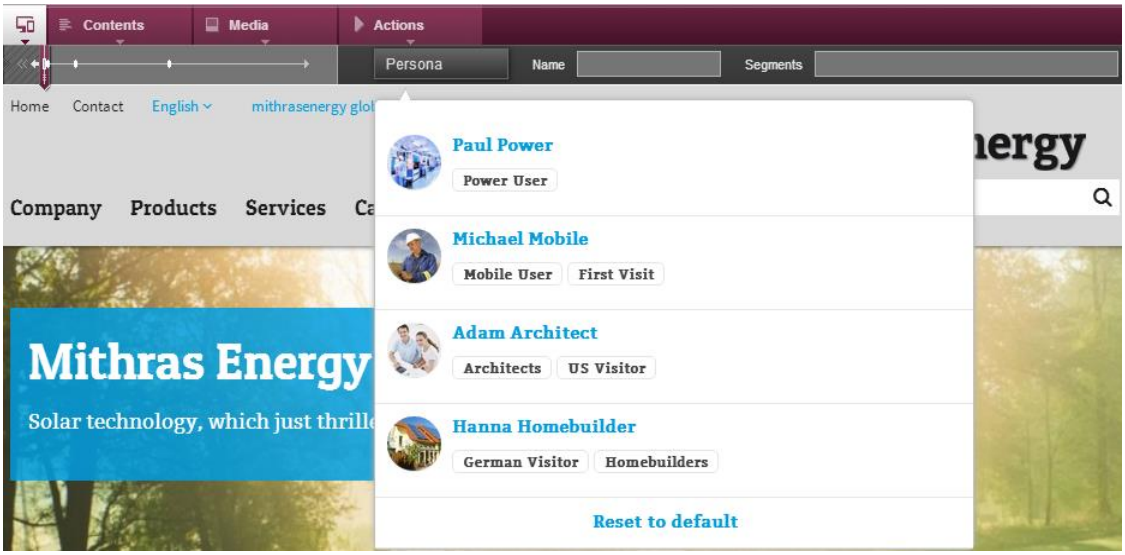

Figure 2-1: Persona Simulator in ContentCreator

In the SiteArchitect preview, a bar associated with the *Persona Simulator* has also been added that looks and functions like the ContentCreator bar (*see Figure 2-2*). The editors can thus use the *Persona Simulator* in SiteArchitect as well.

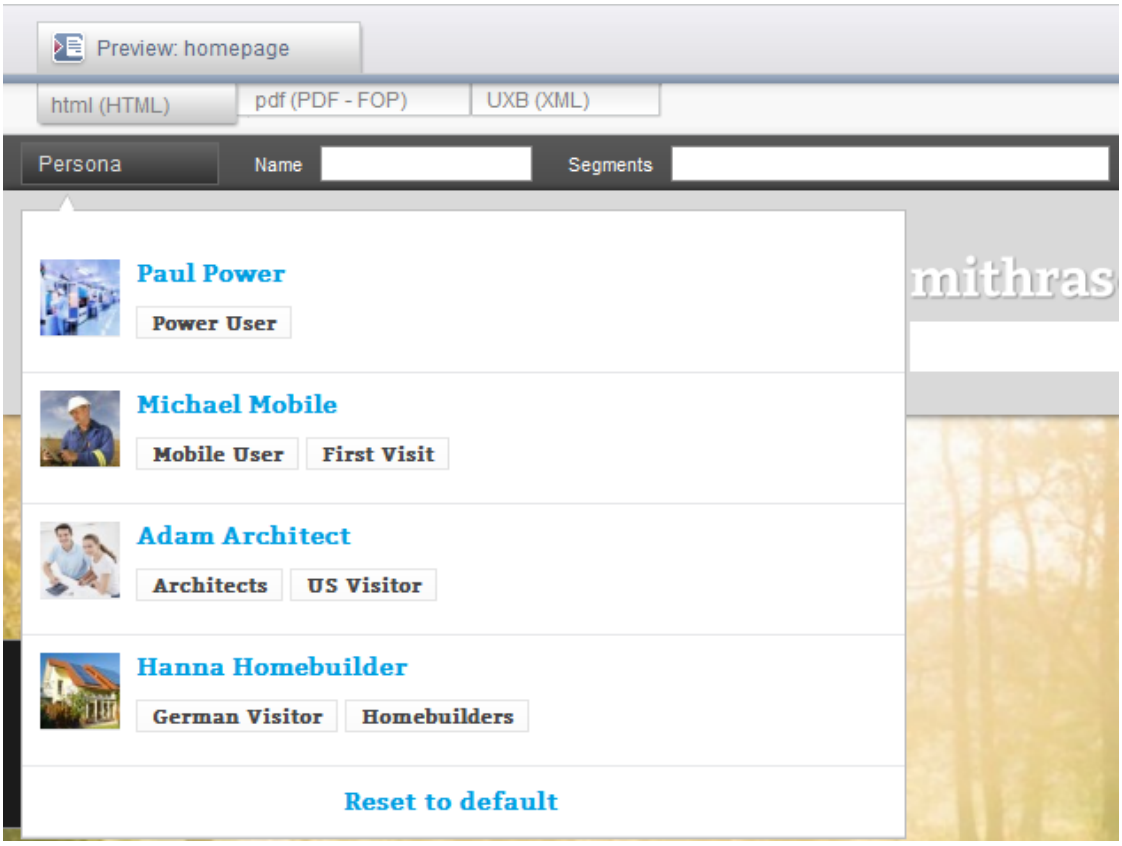

Figure 2-2: Persona Simulator in SiteArchitect

2.4 Profiling & Segmentation Service: Woopra™

Woopra™ is a service for real-time analysis of website visitor behavior. The Woopra user interface provides a variety of information for analyzing this behavior, which can also be expanded as needed.

In this way, different visitors can be detected and identified again upon subsequent visits. This provides the option to freely define and assign different target groups (called segments) to them within the service and to provide them with content specific to the target group(s) via FirstSpirit. The individual visitor receives the information relevant to him or her, which optimizes the personal user experience.

By default, the Real-time Targeting module uses the Woopra™ service previously mentioned. However, if the use of other or additional services is desired, these services can be added to a specific UXP project at any time (see section 1.1.1, page 8). Chapter 6 describes the steps required to add these services.

3 Installation and configuration

Various steps need to be taken before using *Real-time Targeting* functions:

- Registering and configuring Woopra™ see section 3.1, page 15
- Installing the module see section 3.2, page 17
- Configuring the project components see section 3.3, page 18
- Setting up the Persona Simulator see section 3.4, page 20
- Configuring the web components see section 3.5, page 23

Configuring a proxy server is only necessary if it is used to access the Internet (refer to section 3.6, page 29 for more information).

3.1 Registering and configuring Woopra™

Tracking a website visitor is done using a corresponding Profiling & Segmentation Service, which is used by FirstSpirit for the *Real-time Targeting* function, but is not supplied by FirstSpirit.

To use the Woopra™ service, an account registered on the following site is required:

http://www.woopra.com

The website to be analyzed needs to be added to the *My Websites* subpage, which can be accessed using the flyout menu item of the same name under *My Account*. It will then appear in the list of added websites displayed on the page (*see Figure 3-1*).

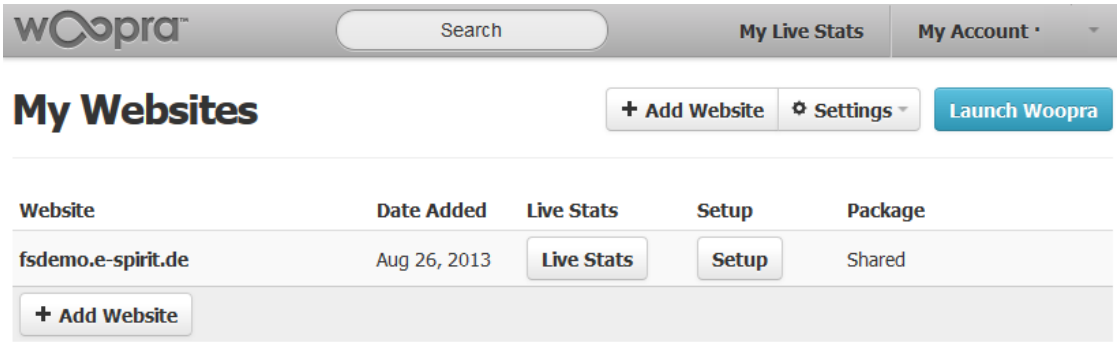

Figure 3-1: Woopra™ - My Websites

Clicking *Live Stats* opens the Dashboard where the current visitor count as well as additional information is compiled in an overview (*see Figure 3-2*).

Figure 3-2: Woopra™ – Dashboard

The website visitors can be assigned to different segments. These segments can be freely defined using the Labels menu item at the bottom. A tracking code, which must be included in the page (*see section 4.4, page 38*), is used to record visitor actions. If a visitor fulfills the conditions defined for the segment(s) based on his or her actions, the visitor is added automatically to the respective segment(s) (*see Figure 3-3*).

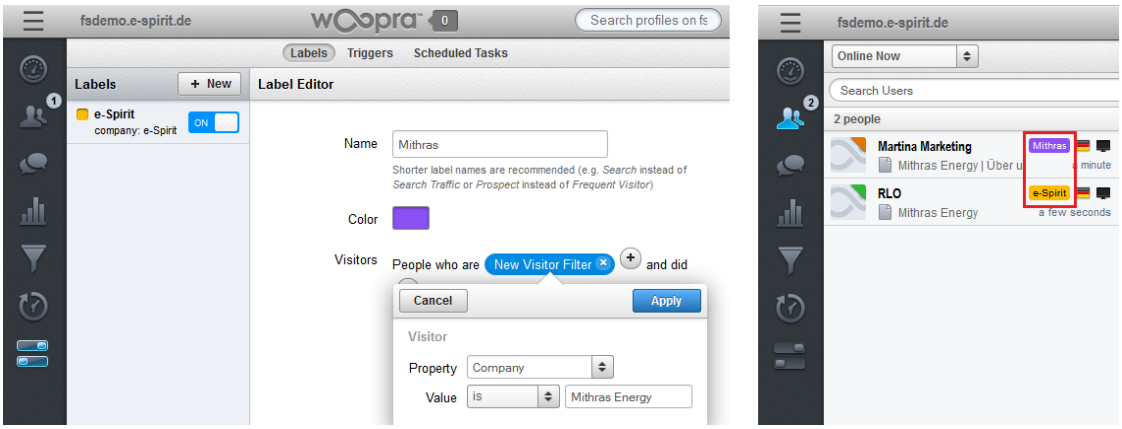

Figure 3-3: Woopra™ – Labels

For more detailed information about Woopra™, please read the Woopra documentation: https://www.woopra.com/docs/

Information on how to use an additional or different Profiling & Segmentation Service is available in chapter 6 starting on page 52 as well as in the particular provider's documentation.

3.2 Installing the module

The module needs to be added to the FirstSpirit Server using the *rtt-fsm-<version number>.fsm* file that was supplied with the software. To install the module, open the *ServerManager* and select *Modules* under *Server properties*.

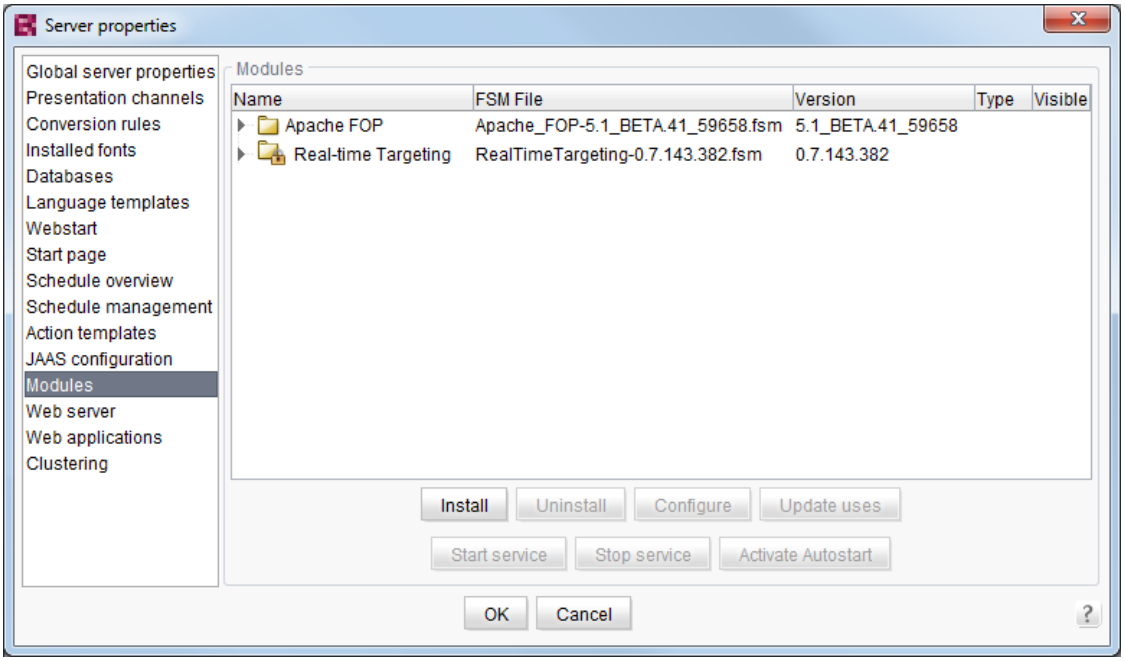

Figure 3-4: Managing modules under "Server properties"

The main panel contains a list of modules installed on the server. After clicking *Install*, select the supplied *rtt-fsm-<version number>.fsm* file and click *Open* to confirm your selection. After successful installation, a *Real-time Targeting* folder is added to the list and must be given *All permissions* (*see Figure 3-4*).

After any module installation or update, the FirstSpirit Server needs to be restarted.

Select *Real-time Targeting Woopra Segment Provider Service* in this folder and open the server-wide Woopra settings by clicking *Configure* (*see Figure 3-5*).

Except for proxy server information, the other information shown in this dialog is already provided initially and just needs to be changed if required. If the module communicates with the Profiling & Segmentation Service via a proxy server, the server's data needs to be entered here (*refer also to section 3.6*).

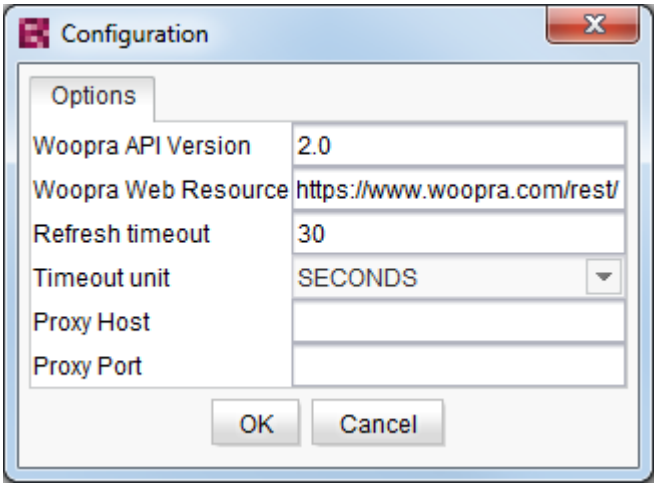

Figure 3-5: Woopra™ settings

Close *Settings* and *Server properties* after configuration by clicking *OK* at each screen.

More information about installing modules is available in the FirstSpirit documentation for administrators.

3.3 Configuring project components

To establish a connection between *Real-time Targeting Woopra Segment Provider Service* contained within the module and Woopra™ and to import the required templates into your FirstSpirit project, two project components need to be added and configured. Open the *ServerManager* and select *Project components* in *Project properties* (*see Figure 3-6).*

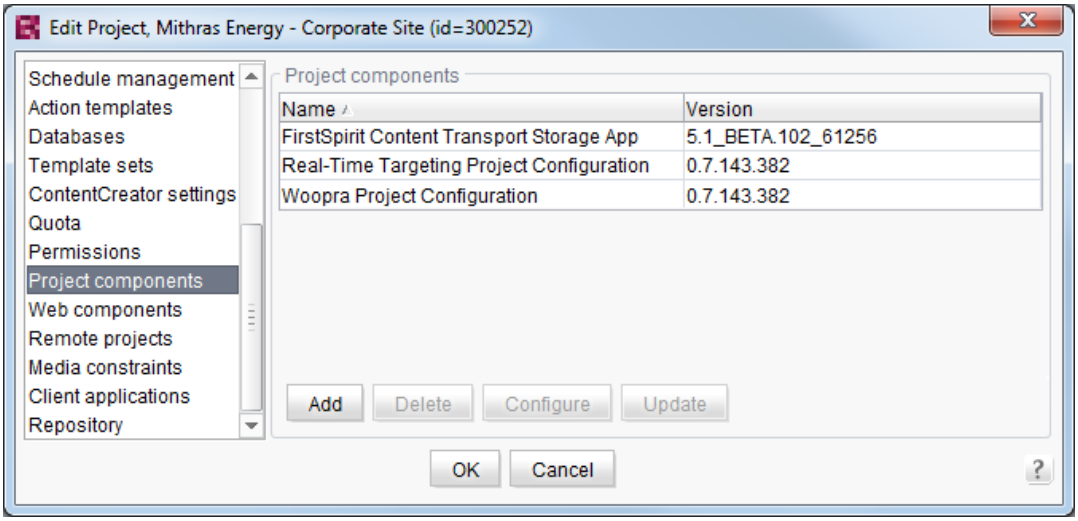

Figure 3-6: Project components

A list of all project components is displayed in the main panel. After clicking *Add*, select *Real-time Targeting Project Configuration* and then *Woopra Project Configuration* and click *OK* to confirm your selection. Both project components will then be added to the list in the main panel and will need to be configured.

Select one component at a time and click *Configure* to open the associated configuration dialog.

Real-time Targeting Project Configuration

In the configuration dialog for *Real-time Targeting Project Configuration*, a schema first needs to be selected before various templates can be imported using the "Import templates" button (*see Figure 3-7*). The schema is required for the data source of the *Personas* (*see section 4.3, page 36*).

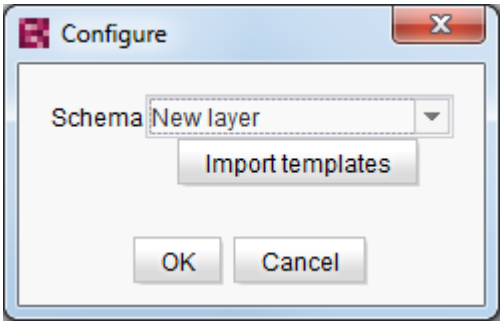

Figure 3-7: Real-time Targeting Project Configuration

Woopra Project Configuration

Access data for Woopra™ and the website to be analyzed needs to be added to the *Woopra Project Configuration* dialog (*see Figure 3-8*). The access data is used by the manufacturer's module service so that a connection can be made between the FirstSpirit Server and Woopra™. The specified website must also be entered in Woopra™ (*see section 3.1, page 15*).

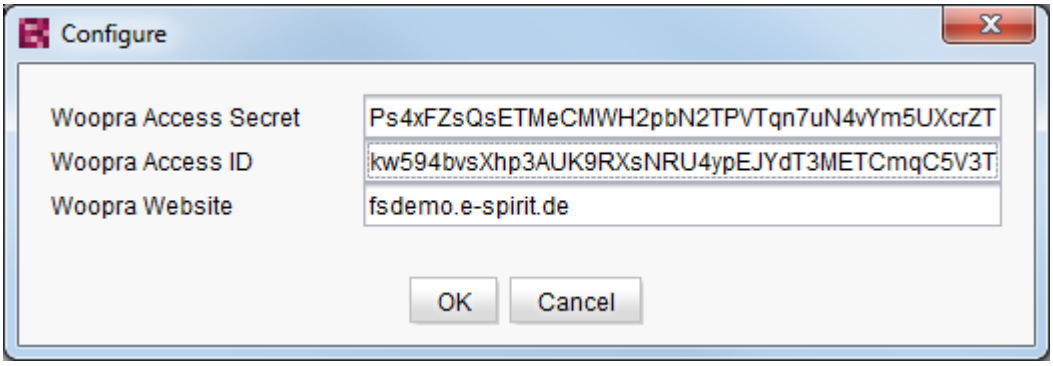

Figure 3-8: Woopra Project Configuration

3.4 Setting up the Persona Simulator

The Persona Simulator requires two modifications to the project settings:

- Selecting the Persona preview parameters see section 3.4.1, page 20
- Adding the persona resolution see section 3.4.2, page 21

3.4.1 Selecting the Persona preview parameters

To use the Persona Simulator in ContentCreator, the multi-perspective preview needs to be activated by selecting *Preview parameters* in the *project options*.

To do this, open the *ServerManager* and select *Options* in the *Project properties (see Figure 3-9).* Under *Preview parameters*, use the corresponding button to select the *Persona Preview Parameter* page template. This page template was already imported to the project automatically during configuration of the *Real-time Targeting Project Configuration* project component (*see section 3.3, page 18*).

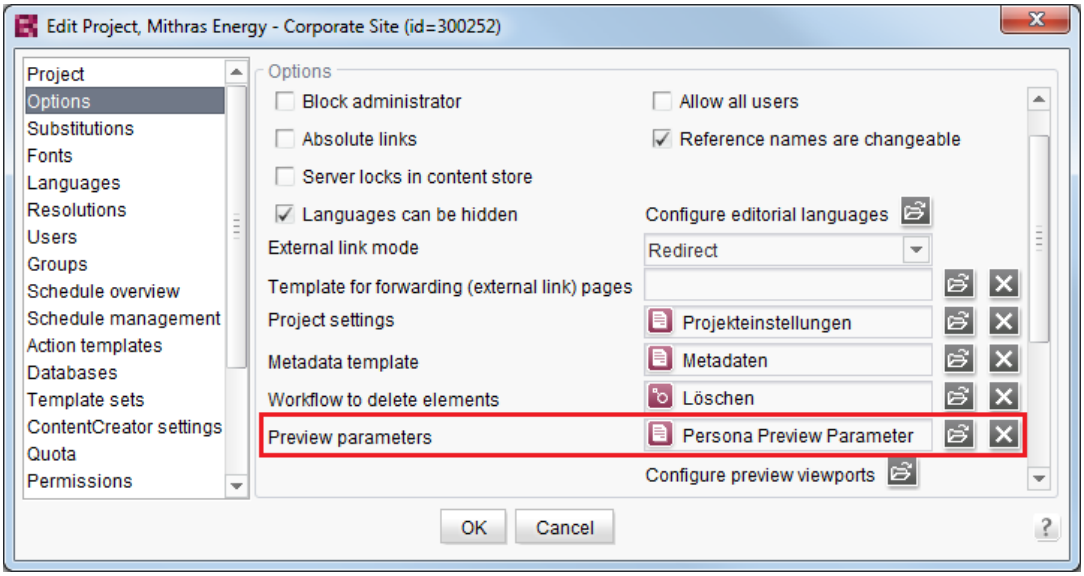

Figure 3-9: Selecting preview parameters

3.4.2 Adding the persona resolution

The individual personas in the Persona Simulator include an image that needs to have its own resolution.

To add the resolution, open the *ServerManager* and select *Resolutions* in the *Project properties*. A list of all existing resolutions is displayed in the main panel.

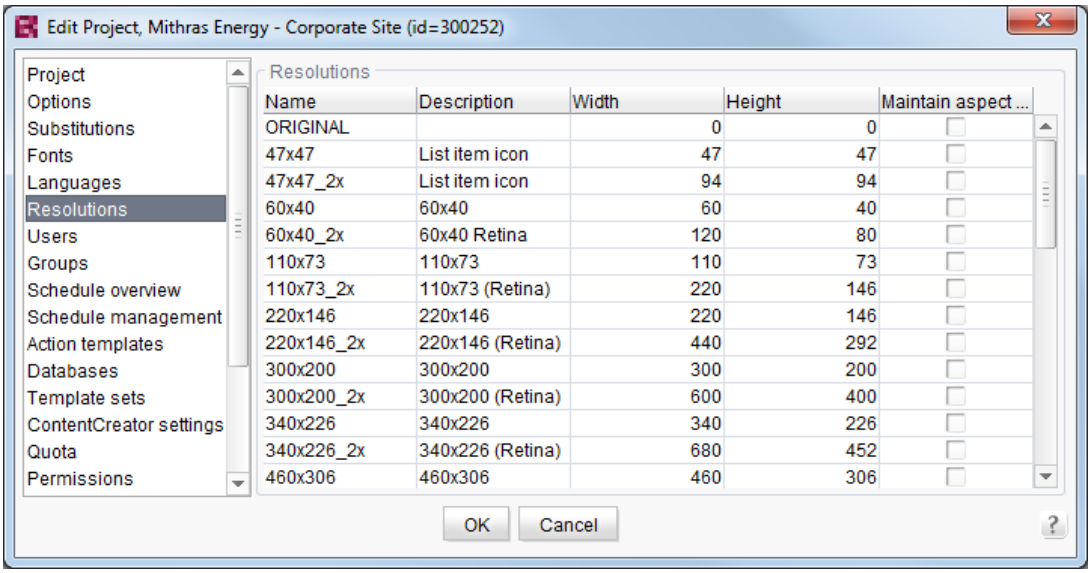

Figure 3-10: Resolutions

Right-click to open the context menu and select *New* to add a new resolution called *persona* with a size of *55x55 pixels*.

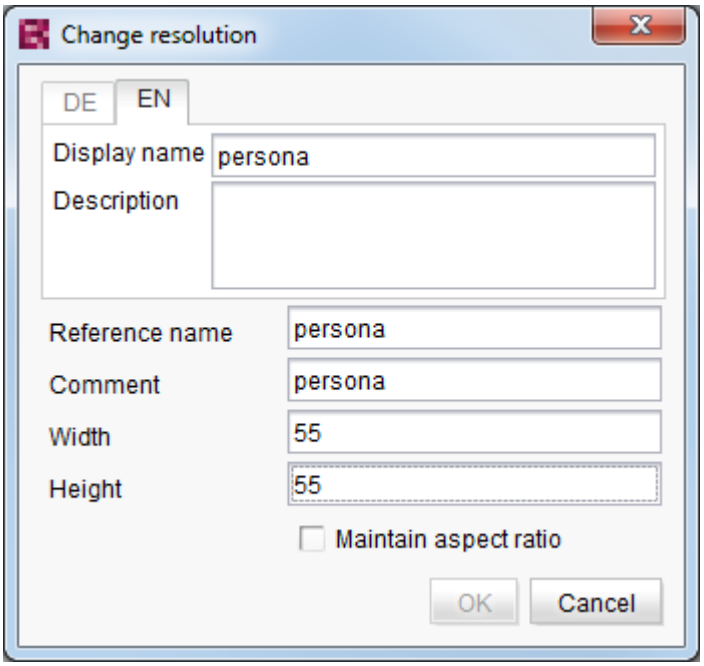

Figure 3-11: Adding a new resolution

3.5 Configuring the web components

The *preview*, *ContentCreator* and *production* each need a web component. To add the component, open the *ServerManager* and select *Web components* in the *Project properties*.

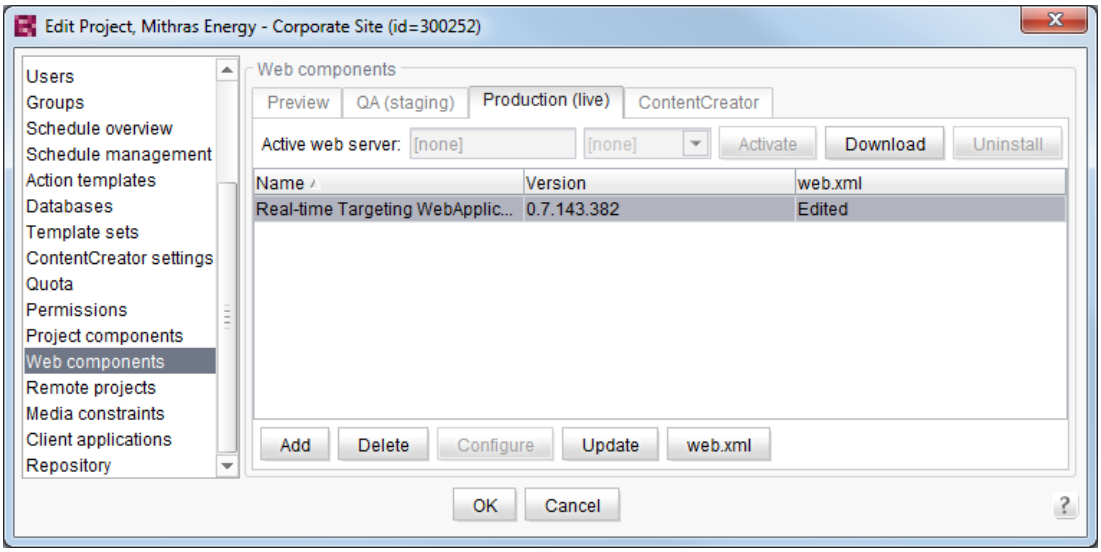

Figure 3-12: Web components

The main panel contains various tab pages (*see Figure 3-12*). Select *Preview*, *Production (Live)* and *ContentCreator* in succession, and after clicking *Add*, select *Real-time Targeting WebApplication* and add it by clicking *OK*.

In the case of *ContentCreator* and *Preview*, the web component must be installed on an *active web server* that is available for selection and must then be activated. The server can be selected using the selection box.

For *Production*, a war file needs to be generated using *Download* and then needs to be installed on a Java web application server (such as Tomcat). However, the associated *web.xml* file needs to be configured before doing this. This is described in section 3.5.1 that follows.

A servlet engine is required which implements the version 3.0 (or higher) servlet API.

More detailed information on how to add web components is available in the FirstSpirit documentation for administrators.

3.5.1 web.xml

The three web components mentioned all initially have the same *web.xml* file. For the *production* web component, however, this initial web.xml file will need to be modified.

Open the *ServerManager* and select *Web components* within the properties of your project. In the main panel, switch to the *Production (Live)* tab page and select *Real-time Targeting WebApplication* there (*see Figure 3-13*).

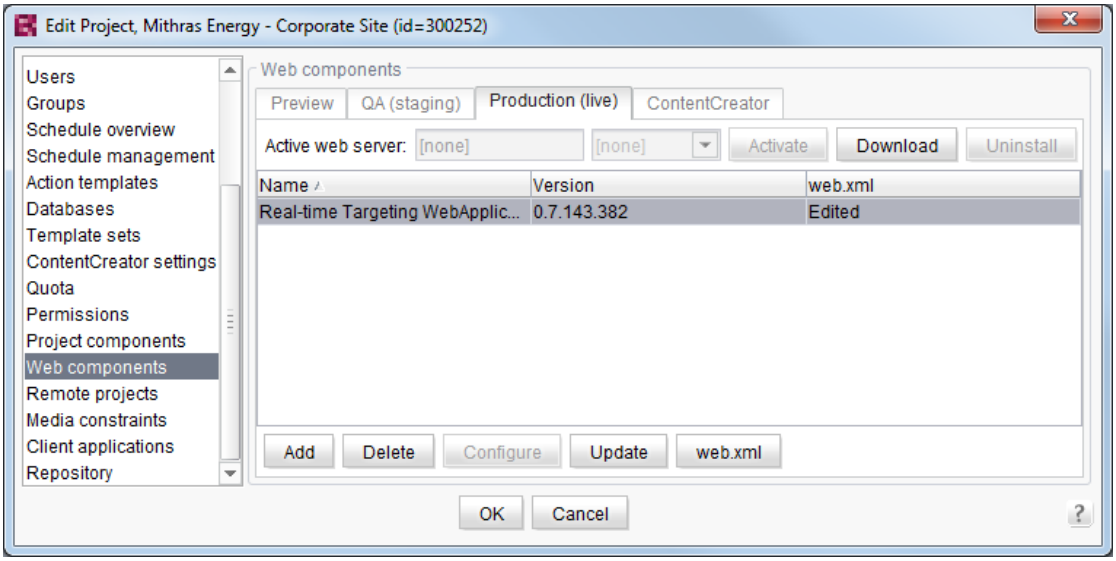

Figure 3-13: Production (Live) web component

Clicking the *web.xml* button opens a configuration dialog where the web.xml can be edited.

To communicate with Woopra™, the *Production* web.xml file requires additional filters that are not used by the other two web components (*also refer to section 3.5.2, page 25*). They are therefore initially commented out. Activate these filters by removing the relevant comment marks and then save the changes using the "Save" button.

Make sure that the URL patterns used in filter mapping match the generated pages (refer also to the structure example in section 3.5.2).

3.5.2 Pipeline filters

To output target-group-specific content on the website, the segments currently assigned to the visitor must be identified.

This task is handled by the *pipeline filters*:

- UxpServletFilter see section 3.5.2.1, page 26
- WoopraCookieFilter see section 3.5.2.2, page 26
- WoopraFirstVisitFilter see section 3.5.2.3, page 26
- GetWoopraUserInformationFilter see section 3.5.2.4, page 27
- RequestParamFilter see section 3.5.2.5, page 28
- ExtractUserInformationFilter see section 3.5.2.6, page 28

These filters provide for personalized display of the web page for the different target groups. They are modular and can be expanded as needed. The particular filter used is precisely defined in the associated *web.xml* file of the web component for preview, production or ContentCreator. Using this definition in the respective web.xml file, the filters run in succession.

In addition to the filters already available, custom filters can be implemented. A description related to this can be found in section 6.1 starting on page 52.

For each filter added to a web.xml file, a mapping is also required which defines when the particular filter takes effect. In this case, it is important to make sure that the file extension specified in the mapping is appropriate for the generated pages.

Structure of the filter definition in the web.xml file

```
<filter>
   <filter-name>FilterName</filter-name>
   <filter-class>FilterClass</filter-class>
</filter>
<filter-mapping>
   <filter-name>FilterName</filter-name>
   <url-pattern>*.FILEEXTENSION</url-pattern>
</filter-mapping>
```

Some filters have additional parameters that absolutely have to be specified. When installing and configuring the Real-time Targeting module, this condition is already fulfilled initially.

Additional data is required only for the GetWoopraUserInformationFilter, which uses the Woopra access data (see section 3.5.2.4). This filter and WoopraFirstVisitFilter also require information from the website that was added to Woopra (see sections 3.1, 3.5.2.3 and 3.5.2.4).

If the Profiling & Segmentation Service is to be replaced, the parameters need to be modified accordingly (see chapter 6, page 52).

If additional filters are to be used, data needs to be added to them based on the same schema.

3.5.2.1 UxpServletFilter

The *UxpServletFilter* is **mandatory** for controlling other filters.

*Since it is required, UxpServletFilter must **always** be set in the first position in the filter mapping.*

3.5.2.2 WoopraCookieFilter

WoopraCookieFilter records the value of the current web page visitor's Woopra cookie and saves it in the current request context under the *rtt.woopra.cookie* key.

The filter features the following parameter:

Name	Definition
cookie_name	Name of the set cookie.

3.5.2.3 WoopraFirstVisitFilter

When calling the web page for the first time, the visitor will not have a Woopra cookie yet. WoopraCookieFilter therefore will not work. However, since Woopra™ has information about the visitor at this time (such as the country), it makes sense to provide it directly as well. For this

reason, *WoopraFirstVisitFilter* sets the Woopra cookie and calls the Woopra Tracking API on the server side.

Since WoopraFirstVisitFilter is to intervene if WoopraCookieFilter does not find a Woopra cookie, WoopraFirstVisitFilter has to be added in the filter mapping after WoopraCookieFilter, but before GetWoopraLabelsFilter.

Name	Definition
cookie_name	Name of the set cookie
cookie_maxage	Validity of the cookie in seconds
cookie_domain	The domain for the cookie. If not specified, no domain will be set.
cookie_path	Information on restricting the cookie's validity to a certain path (default: /)
website	Website that is sent to Woopra™
woopra_url	URL for the Woopra tracking service
woopra_timeout	Timeout after which a request is marked as offline
request_timeout	Request timeout for the Tracking API call

3.5.2.4 GetWoopraUserInformationFilter

GetWoopraUserInformationFilter is used to query specific data about a visitor that has been captured and saved to the Woopra environment. For example, this data could be used when using *smart forms* (*see section 5.2.1, page 42*). *GetWoopraUserInformationFilter* uses the value of the Woopra cookie from the current request context for the query. The specific visitor data identified is saved under the *rtt.woopra.user.<attribute>* key, while the segments assigned to the visitor are passed to the *segments* key.

Name	Definition

woopra_access_id	Woopra™ access ID
woopra_access_secret	Woopra™ access secret
website	Website that is sent to Woopra™
woopra_api_version	Woopra API version
woopra-web_resource	Address of the Woopra REST interface
woopra_date_format	Format for date information sent to Woopra™
request_timeout	Request timeout for queries to Woopra™
connection_timeout	Connection timeout for queries to Woopra™

3.5.2.5 RequestParamFilter

Using *RequestParamFilter*, the target groups can be selected manually for simulation purposes in order to test how the *tags* are working. For this purpose, the filter analyzes the *segments* request parameter and stores its value in the current user session under the *segments* key.

The *reset* request parameter with the value *true* then removes the *segments* key from the current session.

3.5.2.6 ExtractUserInformationFilter

If in addition to the *Real-time Targeting* module the FirstSpirit *Dynamic Personalization* module is used, the *ExtractUserInformation* filter acts as a link between the two modules. It analyzes the user of the personalization in the current session and saves the user's name so that it can be sent to Woopra™.

Name	Definition
user_session_attribute_name	User of the personalization from the current session

user_name	Name of the user from the current session

3.6 Proxy support

To access the Internet, a proxy server may be used which must be specified at various locations depending on the element that requires it.

If the connections between the *Real-time Targeting* module and the Profiling & Segmentation Service are made using the proxy server, the proxy server must be provided to the module within the *Real-time Targeting Woopra Segment Provider Service* (*see Figure 3-14 and section 3.2, page 17*).

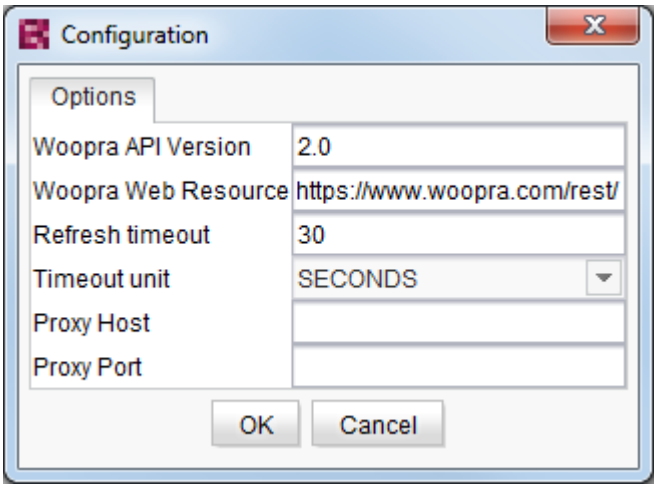

Figure 3-14: Real-time Targeting Woopra Segment Provider Service

For the web components, the use of a proxy server only means that an additional context parameter needs to be added to their associated *web.xml* file (also refer to *section 3.5, page 23*):

```
<context-param>
    <param-name>rtt.httpproxy</param-name>
    <param-value>proxy.mycompany.com:8081</param-value>
</context-param>
```

The rtt.httpproxy parameter name must not be changed.

The expected format for entering the proxy server is server:port.

Use of a proxy server can of course also be specified via the Java system properties `http.proxyHost` and `http.proxyPort`. However, this information is ignored when the parameter previously mentioned is used.

4 Project modifications

After installing the module and configuring the FirstSpirit components (*see chapter 3, page 15*), various modifications need to be made in your project:

- Providing the WoopraValueProvider see section 4.1, page 31
- Integrating the ContentTargeting tag see section 4.2, page 32
- Defining the personas see section 4.3, page 36
- Integrating the tracking code see section 4.4, page 38

WoopraValueProvider provides the segments identified from the Profiling & Segmentation Service within your project so that the editors can assign the content for specific target groups to them.

The *ContentTargeting tag* controls the correct display of this content on the website.

The *personas* allow the editors to check the target group specific assignments of various content variants.

The *tracking code* analyzes the behavior of the visitor on the website.

4.1 WoopraValueProvider

The segments identified from the Profiling & Segmentation Service (*see sections 2.4 and 3.1*) are provided to the editors by *GomIncludeValueProvider*. This involves a default interface provided by FirstSpirit that is used to pass dynamic numbers of values to the following input components:

- CMS_INPUT_CHECKBOX
- CMS_INPUT_COMBOBOX
- CMS_INPUT_LIST
- CMS_INPUT_RADIOBUTTON

Passing the dynamic values automatically to the respective input components is done by including the *CMS_INCLUDE_OPTIONS* data element, within which the class name of the *GomIncludeValueProvider* used must be specified. In this case it is the

`com.espirit.moddev.rtt.gui.WoopraValueProvider` class.

The following example shows the inclusion of *WoopraValueProvider* in the CMS_INPUT_CHECKBOX input component (*refer also to Figure 4-1*):

```
<CMS_INPUT_CHECKBOX
      name="st_segments" gridWidth="3" hFill="yes" useLanguages="no">
  <CMS_INCLUDE_OPTIONS type="public">
    <LABELS>
      <LABEL lang="*">#item</LABEL>
    </LABELS>
    <NAME>com.espirit.moddev.rtt.gui.WoopraValueProvider</NAME>
  </CMS_INCLUDE_OPTIONS>
  <LANGINFOS>
    <LANGINFO lang="*" label="Show To Target Group(s)" />
    <LANGINFO lang="DE" label="Relevanz für Zielgruppen" />
  </LANGINFOS>
</CMS_INPUT_CHECKBOX>
```

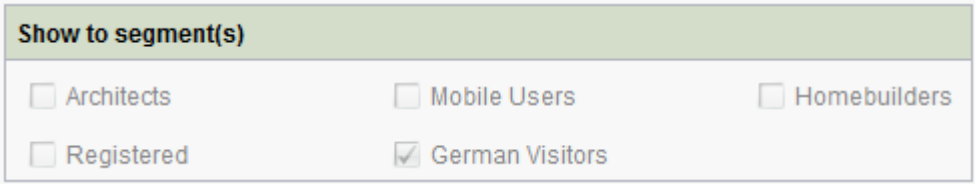

Figure 4-1: Displaying segments as checkboxes

More information on how to use the CMS_INPUT_OPTIONS data element is available in the FirstSpirit online documentation under the following path: Template Development/Forms/Data Elements/OPTIONS/PUBLIC.

4.2 ContentTargeting tag

After a generation process takes place, all content variants created by the editors are initially on the JSP page on the server. Only once the visitor queries the website does filtering for the specific target groups take place so that the visitor sees only the content relevant to him or her.

This filtering is done by integrating a tag library and the ContentTargeting tag provided by the library. This tag requires selection of a prefix in order to be used. The tag library is defined automatically in the *web.xml* file when the project is configured:

```
<jsp-config>
  <taglib>
      <taglib-uri>targeting</taglib-uri>
      <taglib-location>/WEB-INF/targeting.tld</taglib-location>
  </taglib>
</jsp-config>
```

The prefix and ContentTargeting tag are described in the following sections:

- Prefix see section 4.2.1, page 33
- ContentTargeting tag see section 4.2.2, page 34

4.2.1 Prefix

To use the *Real-time Targeting* module, the file extension of the page template being used first needs to be changed from *html* to *jsp*. In addition, the tag library defined in the *web.xml* file of the web application (*see section 4.2, page 32*) needs to be integrated in this page template.

More information about changing the file extension is available in the FirstSpirit Handbook for Developers.

To use the ContentTargeting tag, the *rtt* prefix is used in this document from this point forward.

Example of integrating taglib in JSP pages

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="targeting" prefix="rtt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

While any prefix (e.g. rtt) can be used, the URI must not be changed.

If the "rtt" prefix is changed, the new value is also to be used for the individual tags; this means that "<myPrefix:showContent>" would replace "<rtt:showContent>".

4.2.2 <rtt:showContent>

The *<rtt:showContent>* tag is used to show or hide the content enclosed in the tag. The visibility of the content is controlled by the visitor's affiliation with the defined segments or by entering points in time. The tag is evaluated by comparing the segments of the content to be displayed and of the website visitor. Only when they match is the content provided to the visitor.

The visibility of the content enclosed in the tag also depends on the configured logic. According to the logic, the visitor must either be in all (AND) or in at least one (OR) of the defined segments. If the visitor is not in any segment, the tag content that has not been defined for any segment is output. If the visitor is associated with multiple segments, the first content for which the defined rule is fulfilled is output.

To be able to react specifically to each defined segment, it is also possible to use freely definable IDs to group the content variants for output as needed.

It is recommended that editors always create some content without any segment assignment so that visitors who aren't members of a segment will be shown something.

*This content **must** always be output last to ensure that the content variants specific to target groups are checked first.*

Entering the start and/or end times also makes it possible to view the content based on time, providing for a wide variety of different scenarios. For instance, different content can be displayed for a segment at different periods of time.

Editors can choose a date in ContentCreator using a time bar. This allows the editors to view the content at a particular point in time in the past or future. To include the date, however, it must be passed as an attribute.

The following table contains an overview of all available attributes:

Attribute	Definition
segments	Comma-separated list of segments for which the content is to be visible

logic	Operator applied to the segments (AND or OR; default: AND)
id	Unique identifier for a group of content variants that are to be evaluated together As soon as some group content has been output, no further content is output. (default: defaultID)
startdate	Start date from which the content is to be displayed
enddate	End date up to which the content is to be displayed
referencedate	The date that was selected in ContentCreator using the time bar can be queried using the following call: `String.valueOf(((java.util.Date) session.getAttribute("fs.preview.#time")).getTime())` A similar query can be simulated in SiteArchitect using `#global.startTime`. However, this query only returns the date of the last change.

Example of how to use the ContentTargeting tag:

```
<rtt:showContent
     segments="Architects, Homebuilders" logic="OR" id="300642"
     startdate="1336226501635" enddate="1336226503635"
     referencedate="1336226502635">

[... content to display ...]

</rtt:showContent>
```

A more detailed example is provided in chapter 5, page 39.

4.3 Persona Simulator

Installing and configuring the *Real-time Targeting* module as described in chapter 3 includes the import of three format templates in the Template Store. They need to be referenced via a CMS_RENDER call in the page template being used.

The *Persona Box* format template makes the *Persona Simulator* available on both clients. It is added to the preview bar in ContentCreator. However, since this bar does not exist in the SiteArchitect preview, an equivalent bar is added there using the *Persona Preview Bar* format template. The *Persona Script* format template is used to provide required, additional CSS and JavaScript code.

```
[...]
<head>
   $CMS_RENDER(template:"persona_script")$
</head>
$CMS_RENDER(template:"persona_previewbar")$
$CMS_RENDER(template:"persona_box")$
[...]
```

The JavaScript uses the jQuery library.

Inclusion of jQuery version 1.9.1 or higher in the correct page template is mandatory. Otherwise, the Persona Simulator cannot be used and instead error messages will appear in the browser's JavaScript console.

In addition, the empty *Personas* data source based on a schema imported in the Template Store is added in the Content Store. The data source is used by the Persona Simulator to represent the various segments in the preview and in ContentCreator and to provide editors with a view of the website as seen by the specific target group(s). For this purpose, the corresponding personas need to be generated in the data source.

In each new dataset a name must be entered and an image needs to be selected. In addition, at least one checkbox must be selected to specify which segments will be represented by the persona (*see Figure 4-2*).

Providing the segments in the form of checkboxes is handled automatically by using the WoopraValuaProvider (*see section 4.1, page 31*).

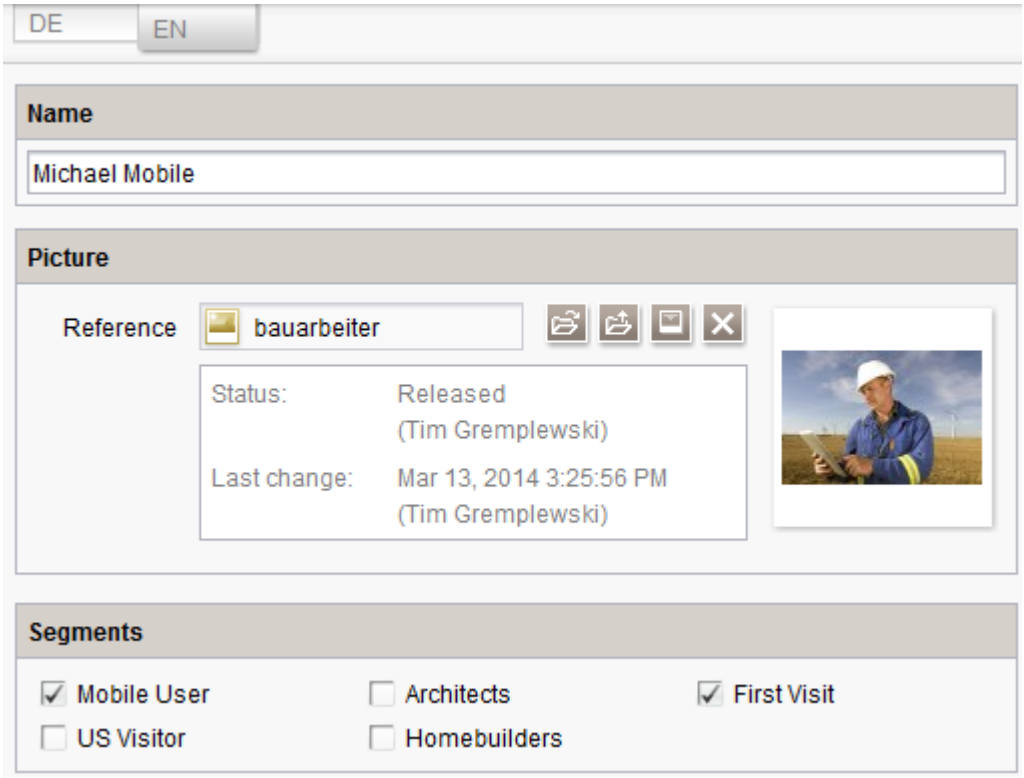

Figure 4-2: Persona

The personas generated in the data source can then be selected both in the preview and in ContentCreator using the Persona Simulator (*see Figure 4-3*). If the page contains specific content for the segment represented by the selected persona, this content is displayed.

For the correct presentation of specific content, the steps described in section 4.2 must be carried out first.

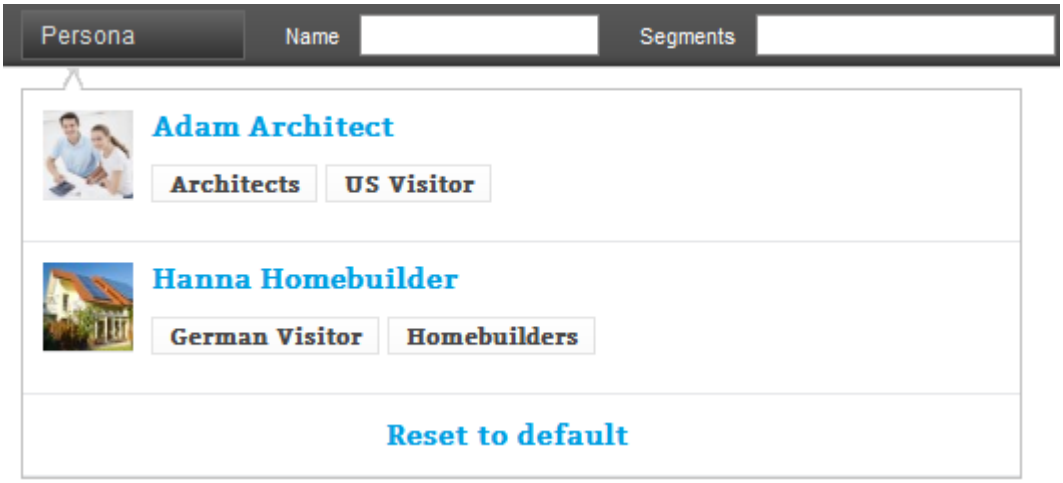

Figure 4-3: Persona Simulator

If multiple segments overlap for the representation, only the first applicable piece of content will be output. If the page contains only specific content that does not match the segments represented by the persona, only the default content of the page will be displayed.

Using the Persona Simulator, editors are given the option of checking the assignment of content variants they created for specific target groups in the preview and in ContentCreator.

Without the simulator, the editors would only ever be able to view the page's default content. The display of content specific to target groups in this case would only be possible once the content went live.

4.4 Tracking code

Woopra™ analyzes the behavior of website visitors using a tracking code. This code was imported to the FirstSpirit project in the form of a format template during installation and configuration of the Real-time Targeting module as described in chapter 3. In the project, the format template only has to be included in the page template by using a CMS_RENDER call.

```
$CMS_RENDER(template:"woopra_tracking", woopra_domain:"http://website")$
```

The *woopra_domain* is used to pass the domain (stored with Woopra) of the website to be tracked.
More detailed information about the tracking code is available in the Woopra™ documentation:

https://www.woopra.com/docs/setup/javascript-tracking/

5 Use in FirstSpirit

The *Real-time Targeting* module functions can be provided in FirstSpirit in a wide variety of ways. The best implementation in each case must therefore always be determined based on the prevailing requirements specific to the project.

The following sections cover how to generate content specific for target groups using an FS_LIST element as well as an additional *Real-time Targeting* module use case.

5.1 FS_LIST

This section shows an example of how *Real-time Targeting* functions are used in conjunction with an FS_LIST element, which in this case is used in a section template.

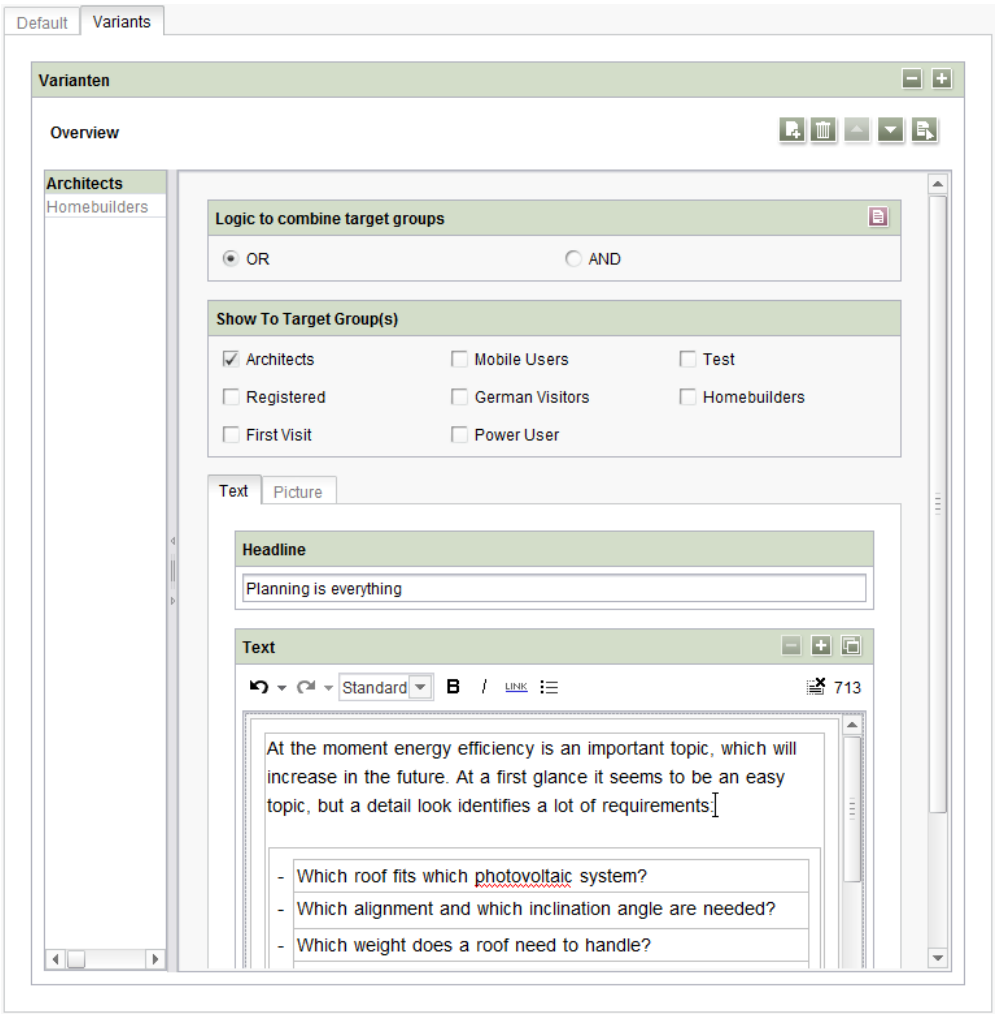

Figure 5-1: FS_LIST

In this example, each content variant specific to the target group(s) corresponds to an entry in the FS_LIST, which is contained here in the *Variants* tab page (*see Figure 5-1*). Each of these variants contains a number of checkboxes that represent the predefined segments. Analysis of these segments takes place automatically (*also refer to section 4.1, page 31*).

Since multiple checkboxes can be selected for use, it is also important to specify the way in which the selected segments are to be linked to each other. In this example, radio buttons were used for this purpose (*refer also to the table in section 4.2.2, page 34*).

The visibility of the particular variant depends on the website visitor's group membership. When an AND operator is used, the visitor must be included in all segments selected using the checkboxes; otherwise, inclusion in at least one of these segments is sufficient.

*If a visitor fulfills the visibility rules of multiple variants, the visitor will only be shown the **first** piece of content relevant to him or her.*

For visitors who cannot be assigned to any segment, default text should always be provided, which is included in the *Standard* tab page in this example.

The output of variants specific to the target group(s) and contained in the FS_LIST is controlled using the ContentTargeting tag, which is to be included in the HTML channel of the associated template (*also refer to section 4.2.2, page 34*).

```
<rtt:showContent
      segments="$CMS_VALUE(st_segments.toString(","))$"    // Checkboxes
      logic="$CMS_VALUE(st_logic)$"                        // Radiobuttons
      id="$CMS_VALUE(#global.section.id)$" >
[...]
</rtt:showContent>
```

The website visitor is shown the content enclosed in this tag according to the segments assigned to him or her.

In this example, a section specific to the target group(s) has been added to the *start page* in the *Mithras Energy* demo project. According to the definition visible in *Figure 5-1*, this includes in addition to general text (*see Figure 5-2*) a variant for *Architects* (*see Figure 5-3*).

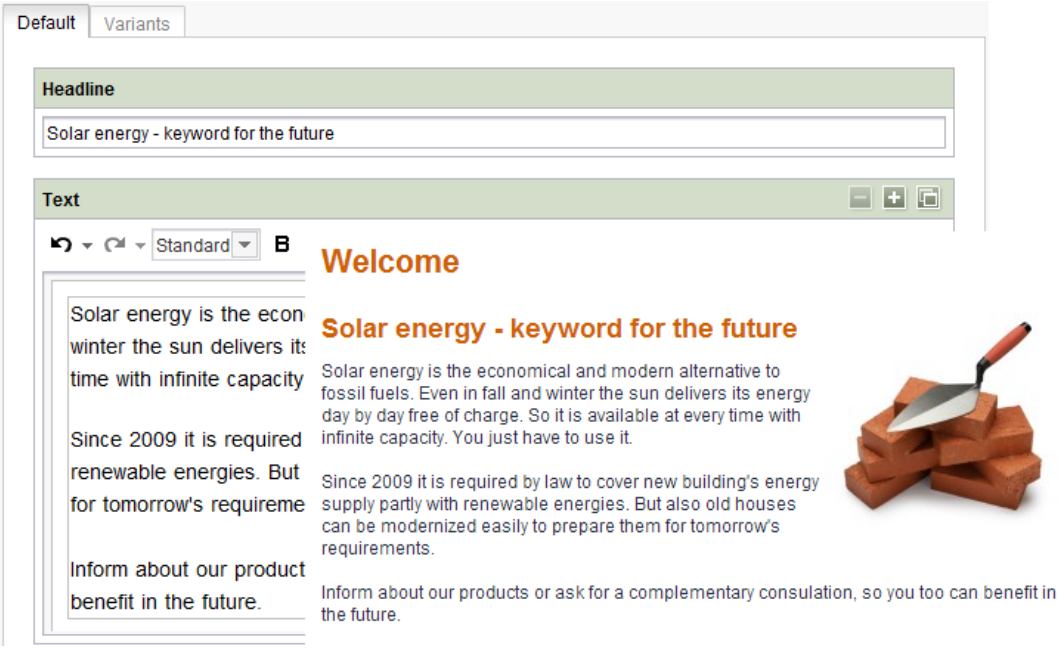

Figure 5-2: General view of section

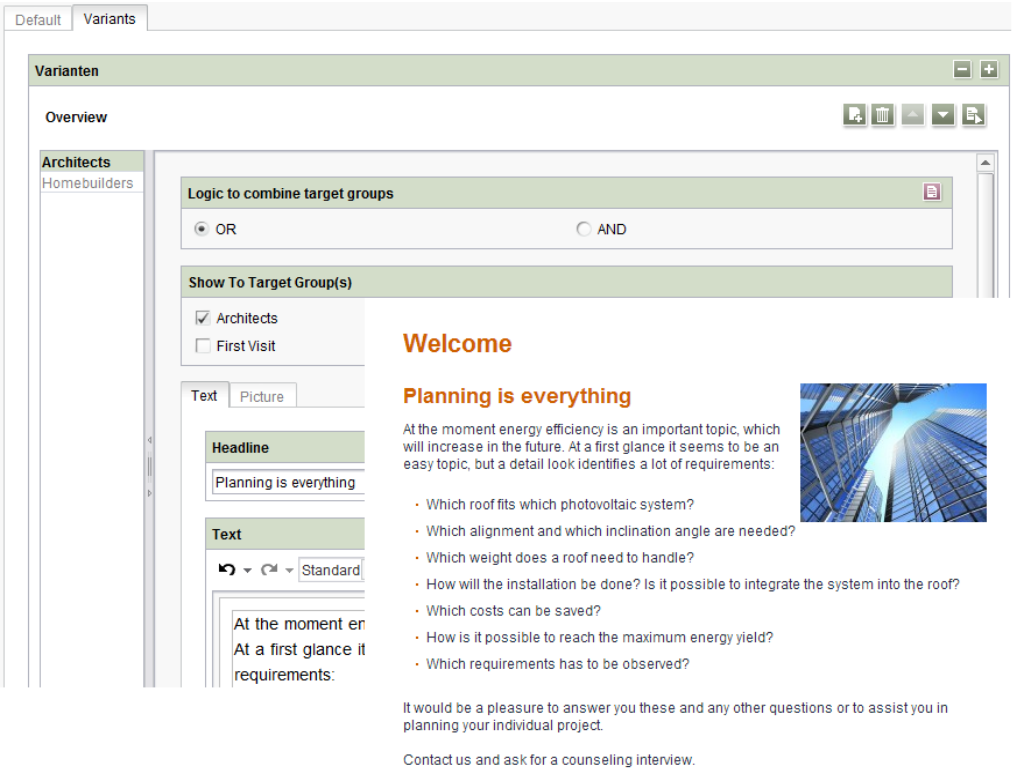

Figure 5-3: Specific view of the section for architects

5.2 Advanced use cases

In addition to providing website content that is specific to the target group(s), other use cases are possible which can be implemented based on the information collected about the website visitors.

The following sections describe examples of some of these use cases.

5.2.1 Smart forms

The *Real-Time Targeting* functions enable the use of what are called *smart forms*. These are forms with fields that are already filled with known data when called and thus only have to be edited as needed. The relevant data is collected the first time the form is sent and can then be used for any other form on the site.

In this way, the website visitor does not have to re-enter redundant information, which improves usability.

For the next example, the Real-time Targeting module must be installed and configured according to the steps described in chapters 3 & 4.

This example outlines how to use *smart forms* with the *FormEdit* FirstSpirit module in conjunction with *Real-time Targeting* functions. Use of the *FormEdit* module is of course not mandatory. Any other type of form can be used as well.

A form created using *FormEdit* is composed of three parts, the *start*, *block* and *end*, which are represented by section templates and were added to the project at the time the *FormEdit* module was installed.

In this example, only the visitor's name and e-mail address fields should be pre-filled. To collect this data initially, the following script code must be added to the *start* HTML area. This code must be appropriately modified if additional data is to be saved:

```
<script>
   var submit_$CMS_VALUE(fr_st_name)$ = false;
   $('#$CMS_VALUE(fr_st_name)$').submit(function(e) {
      var formName = $('#name').val();
      var formEmail = $('#email').val();
      woopra.identify({
         email: formEmail,
         name: formName
      }).push();
      setTimeout(function(){
         submit_$CMS_VALUE(fr_st_name)$ = true;
         $('#$CMS_VALUE(fr_st_name)$').submit();
      }, 2000);
      if(!submit_$CMS_VALUE(fr_st_name)$) {
         e.preventDefault();
      } });
</script>
```

This code reads the visitor's input from the form the visitor initially used and transmits the data to Woopra™ in this case (*see section 2.4, page 14*). There, the data is saved as the visitor's attributes and can be called up any time as needed.

The individual fields that make up the form need to be defined using an FS_LIST element within the *block* (*see Figure 5-4*). These fields are represented by additional section templates which are also provided with the *FormEdit* module installation.

*To ensure the aforementioned script works correctly, the fields "Name" and "E-mail" used must have unique "name" and "email" identifiers in **all** forms.*

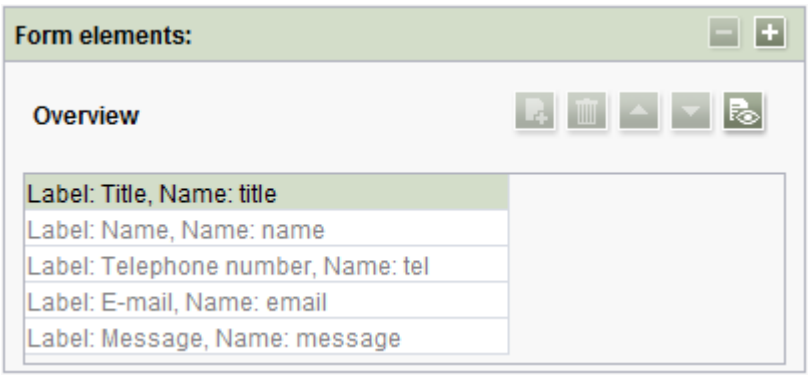

Figure 5-4: FormEdit – Form block

The "Text" form element is used for all text input, including for instance a visitor's name and e-mail address. To pre-populate these fields in other forms with data that has already been collected, an Else case must be added to the IF query for the *st_value* input component in the HTML area of the associated section template:

```
$CMS_SET(fr_st_value)$
   $CMS_IF(!st_value.isEmpty)$
      $CMS_VALUE(st_value)$
   $CMS_ELSE$
      <c:out value="${sessionScope['rtt.woopra.user.$CMS_VALUE(st_name)$']}"/>
   $CMS_END_IF$
$CMS_END_SET$
```

In addition, within the project properties for the respective web component, the *web.xml* file of the Woopra web application needs to be modified by adding the *GetWoopraUserInformationFilter* pipeline filter to it (*see section 3.5.2.4, page 27*). This filter allows for analysis of the attributes saved for a visitor in Woopra™.

Since these attributes are not yet captured when the visitor first visits the website, only empty forms that are structured according to the definition in the FS_LIST of the block are presented to the visitor initially (*see Figure 5-5 and Figure 5-6*).

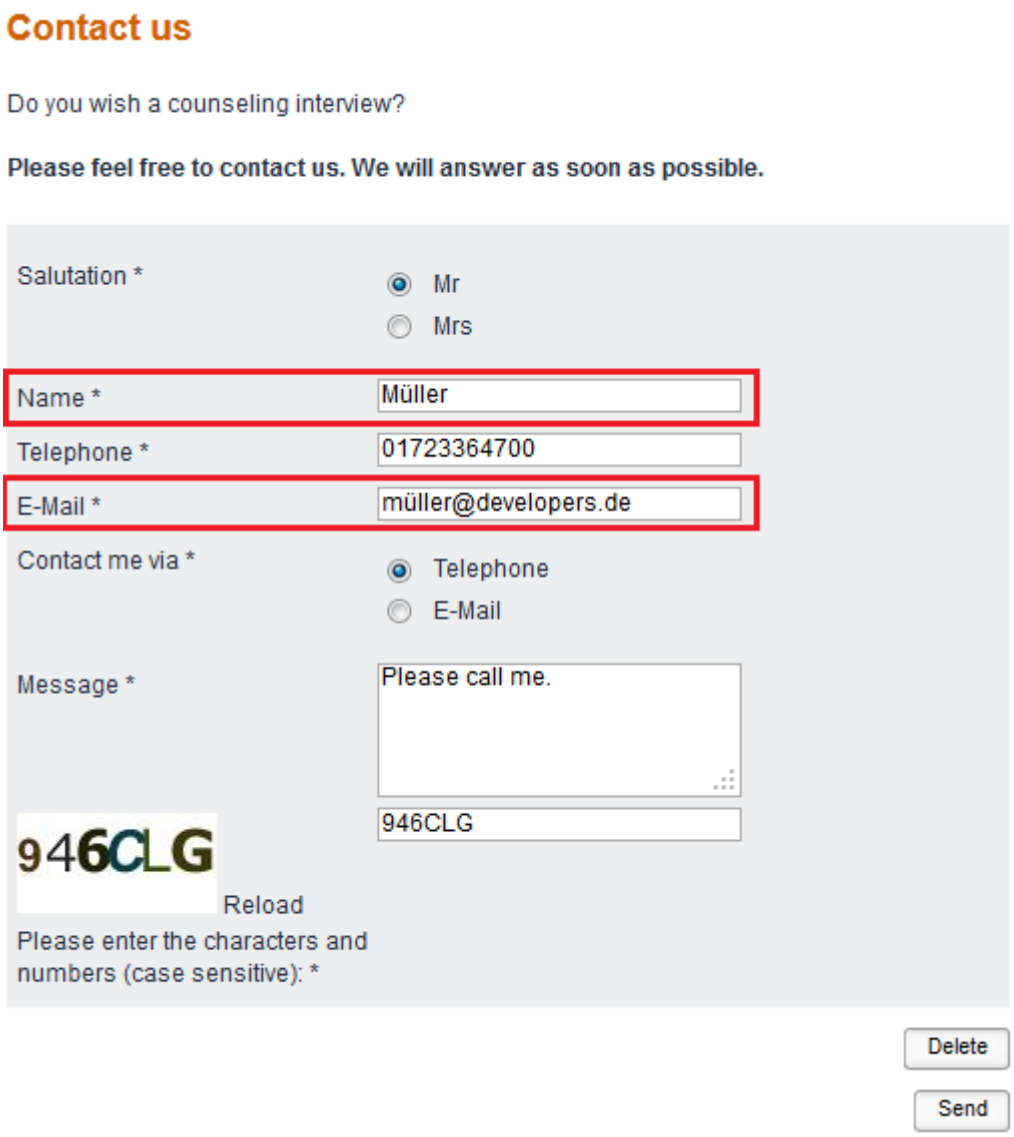

Figure 5-5: Form with visitor data that is transmitted to Woopra™

Only after the relevant data is input and this type of form is sent is the data transmitted to Woopra™ and stored there (*see Figure 5-6*).

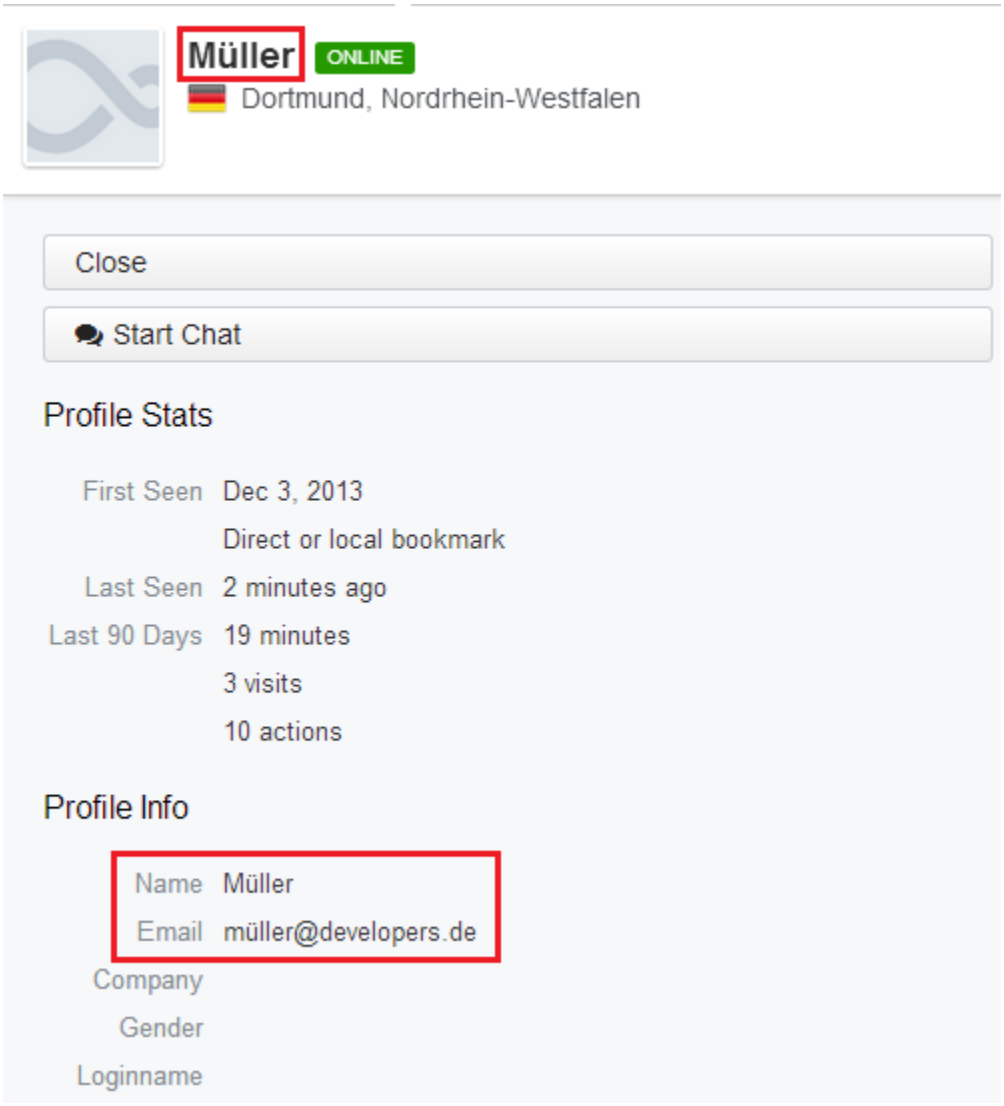

Figure 5-6: Woopra™ - Saved visitor data

In addition to the transmitted data, the visitor's activities on the site and the time of his/her last visit are captured.

Now when any other form is called that has "Name" and "E-mail address" fields with preset unique identifiers, the visitor data stored in Woopra™ is analyzed and written to the form (*see Figure 5-7*).

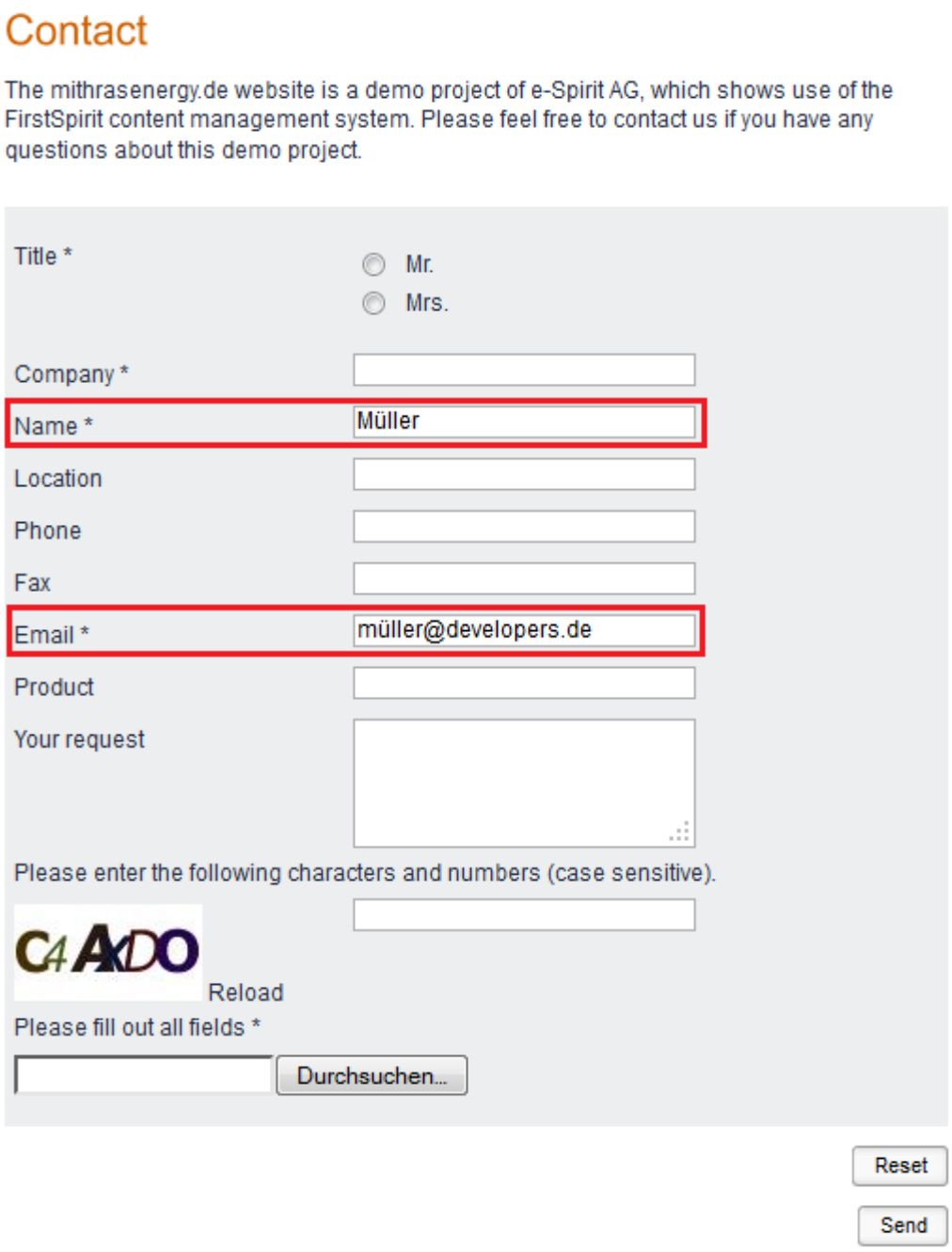

Figure 5-7: Pre-population of a second form

The visitor in this example has the option of editing the data displayed in the form. In other scenarios, however, it would also be possible for instance to only display the data to the visitor or to hide the entire form if all the required data is available.

5.2.2 Scoring

As already covered in chapter 1 under item 2c, a website visitor's interests for specific topics are usually identified using a scoring mechanism. The visitor receives a previously specified number of points for calling particular content and these points are transferred to the Profiling & Segmentation Service, which then adds them up. If the total number of points (aggregate) reaches a defined threshold value, the visitor is automatically assigned to a particular segment.

This section shows how scoring is used in conjunction with datasets. The mechanism, however, can also be used for other FirstSpirit elements.

In this example, website visitors are to be identified as architects or homebuilders based on the particular content they read. For this purpose, an FS_LIST element has been added to the datasets with entries that contain the option to select the two segments (Architects, Homebuilders) and a field for the number of points given (*see Figure 5-8*). This allows the editors to assign the datasets different score values for the two segments.

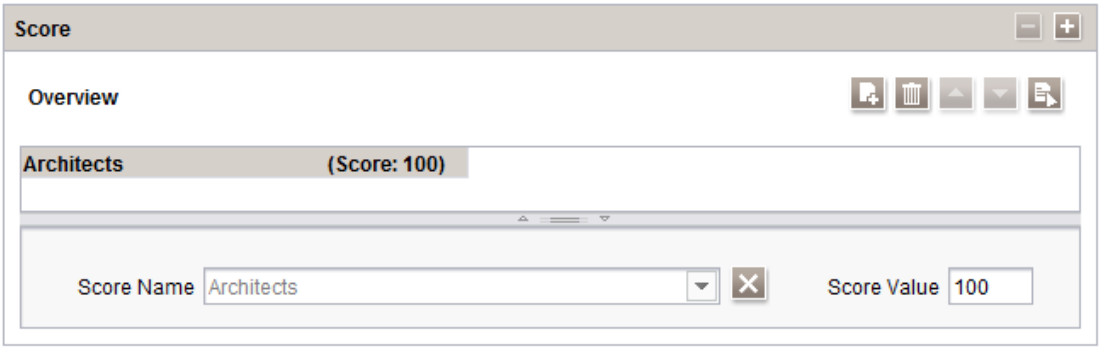

Figure 5-8: Defining a score value for architects

The segments added in the FS_LIST are transmitted together with the score values defined for them to the page context. The following lines are added to the HTML area of the associated table template:

```
$CMS_SET(#global.pageContext["set_scoring"],cs_scoring)$
```

The page context for its part is evaluated using the following query, which is added in the HTML area of the page template:

```
$CMS_IF(!#global.preview)$
   $CMS_IF(#global.pageContext["set_scoring"] != null)$
      $CMS_SET(scoring, #global.pageContext["set_scoring"])$
   $CMS_END_IF$
$CMS_END_IF$
```

The threshold to be reached and the assignment resulting from attaining this threshold are stored in the Profiling and Segmentation Service. In the *Settings* of the Woopra™ service used by the *Real-time Targeting* module, a new *Action Data Schema* was first added under *Schema*, and then the *Properties* for architects and homebuilders were added to the schema (*see Figure 5-9*).

More information on how to configure the Action Data Schema is available in the Woopra™ documentation:

https://www.woopra.com/docs/manual/configure-schema/

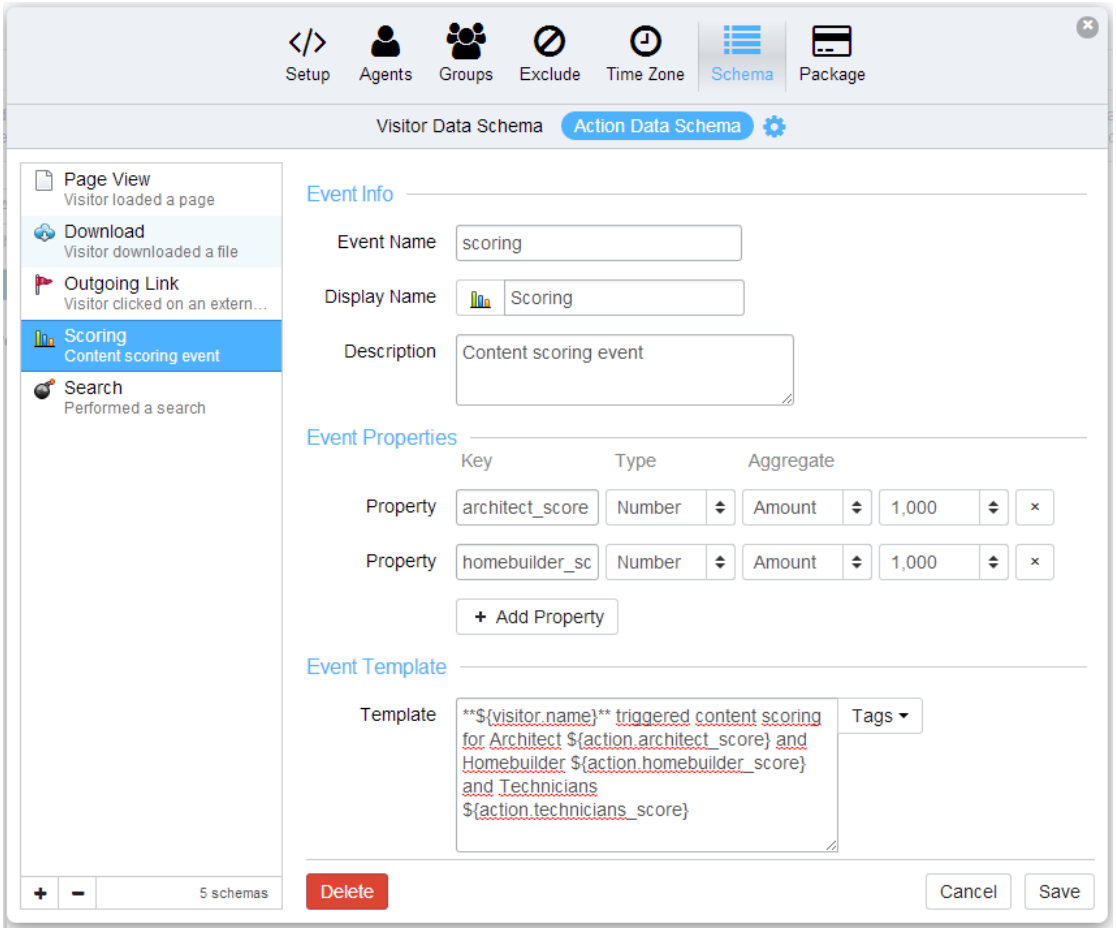

Figure 5-9: Woopra - Action Data Schema

These *Properties* can be used to define the threshold value when configuring a new segment.

In this example, a website visitor who has attained a score value of 1000 for architects should be automatically identified as such and is assigned to the *Architects* segment (*see Figure 5-10*). The equivalent requirement applies to homebuilders.

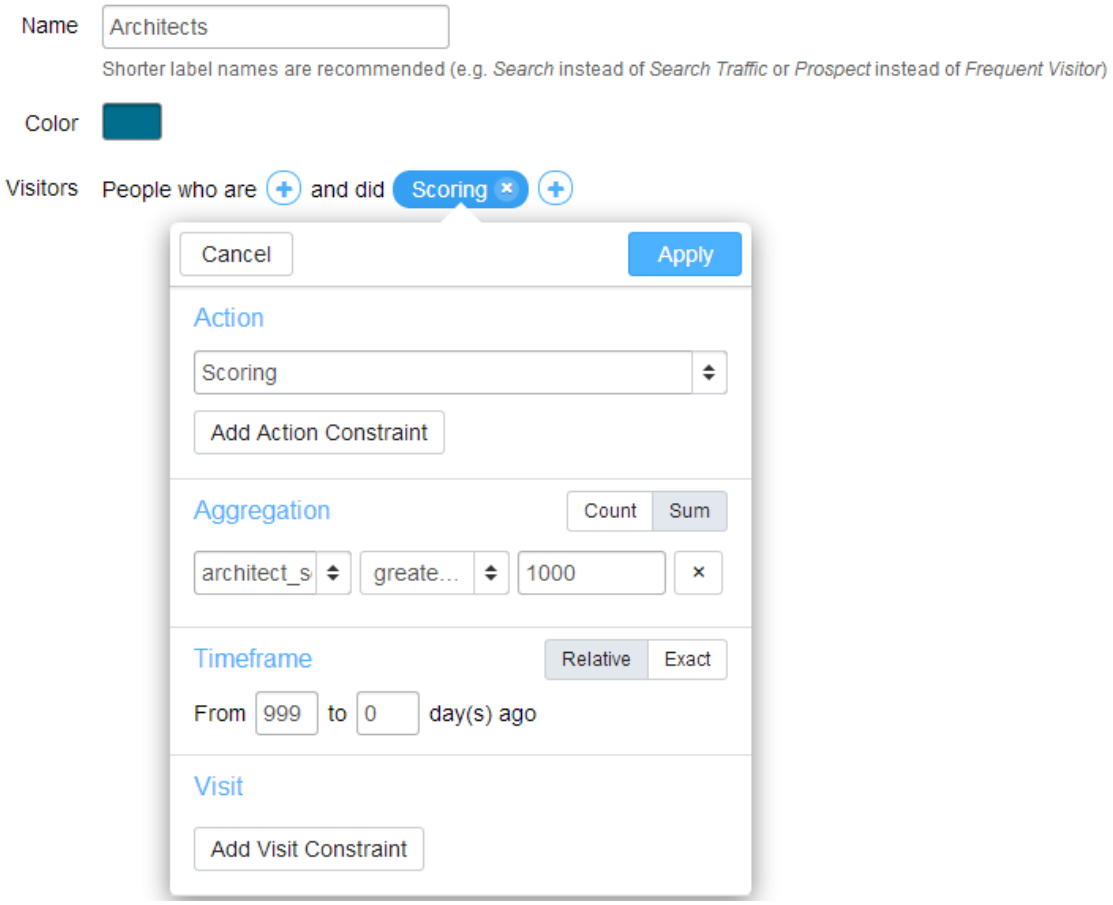

Figure 5-10: Woopra - Scoring for architects

The *Properties* of the *Action Data Schema* are also used in the tracking code, which analyzes the behavior of website visitors (*also refer to section 4.4, page 38*). The tracking code was imported to the project during installation, as described in chapter 3, and the following lines need to be added to it in order to transmit the score values to the Profile & Segmentation Service:

```
<script>
[...]
$CMS_SET(hasScore, false)$
$CMS_SET(scores, "")$
```

```
$CMS_IF(scoring != null)$
    $CMS_FOR(entry, scoring)$
       $CMS_IF(entry.st_score_value > 0)$$CMS_SET(hasScore, true)$$CMS_END_IF$
     $CMS_IF(#for.index > 0)$$CMS_SET(scores, scores + ",")$$CMS_END_IF$
     $CMS_SET(scores, scores + entry.st_score_name + " : " + entry.st_score_value)$
    $CMS_END_FOR$
$CMS_END_IF$

woopra.track();

$CMS_IF(hasScore)$
     woopra.track('scoring', {
       $CMS_VALUE(scores)$
     });
$CMS_END_IF$
</script>
```

*When expanding the tracking code, watch out for the **woopra.track();** line. This expression is also present at the end of the initially imported format template and accordingly must be removed from there when adding code in order to prevent duplicates.*

In this case, this code refers the score value stored in the data sources to the corresponding property in Woopra™ as soon as the dataset is viewed by a website visitor. The information about this action appears in Woopra™ at the same time (*see Figure 5-11*).

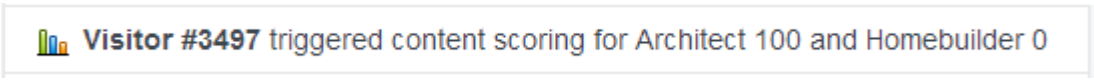

Figure 5-11 – Identifying a potential architect

6 Additions

By default, the *Real-time Targeting* module uses the Woopra™ Profiling & Segmentation Service. If other or additional services are required, these can be added to the User Experience Pipeline (UXP) at any time for projects. The module at this location has a completely open architecture and is not limited to a specific service.

Two different scenarios are basically possible:

a) Woopra™ is retained as a service.

Additional filters are added to the UXP which connect additional systems and provide more information about the website visitor. Implementing additional filters is described in section 6.1.

b) Woopra™ is replaced by a different service.

This type of Profiling & Segmentation Service must fulfill some requirements in order to be suitable for use with the *Real-time Targeting* module. These requirements are explained in section 6.2.

6.1 Implementing additional filters for the UXP

To output target-group-specific content on the website, the segments currently assigned to the visitor must be identified. This task is handled by the filters that request the respective segments from the Profiling & Segmentation Service and write them to the visitor's current session. The information is then used by the ContentTargeting tag to display the relevant content to the visitor.

If another Profiling & Segmentation Service is to be used, a corresponding filter must be implemented that handles identification of segments.

For this purpose, the *rtt-api.jar* file must be integrated in your project and a new class needs to be created that extends the *AbstractUxpFilter* filter:

```
public class ExampleIpFilter extends AbstractUxpFilter
```

This extension provides for various methods.

The *init()*, *doFilter()* and *destroy()* methods are only used to control the filter process.

The *run()* method contains the actual logic of a filter. To prevent multiple calls, the first thing to analyze in the filter is whether it has already been run by using the *shouldFilter()* method.

The implemented filter should be provided as a web component of a FirstSpirit module. The filter can then be configured in the web.xml file of the web component.

During installation and configuration of additional filters in the FirstSpirit preview and production environment, it is important to place these filters in the web.xml file only after the two filters (UxpServletFilter and RequestParameterFilter).

The two sections that follow use examples to describe the implementation of two custom filters:

- ExampleIpFilter see section 6.1.1.1, page 53
- ExampleLocationFilter see section 6.1.1.2, page 54

The *ExampleLocationFilter* is to determine the country associated with a visitor by reading the visitor's IP via the *ExampleIpFilter* and to write it to the session so that it can be output later on the website using JSP.

6.1.1.1 ExampleIpFilter

This example is not suitable for production use.

The IP address of the current visitor is identified in the *run()* method of the *ExampleIpFilter* using the request and is saved in the current request context under the variable name *rtt.user.ip*.

```
RequestContext ctx = ((UxpServletRequest)servletRequest).getRequestContext();
String ipAddress = request.getHeader("X-FORWARDED-FOR");
if (ipAddress == null) {
    ipAddress = request.getRemoteAddr();
}
ctx.set("rtt.user.ip", ipAddress);
```

6.1.1.2 ExampleLocationFilter

> *This example is not suitable for production use.*

The IP address of the current visitor is first read out from the *rtt.user.ip* variable of the request context in the *run()* method of the *ExampleLocationFilter* filter. The associated country is then identified using the REST service from FreeGeolp. This is saved under the variable name *rtt.user.country* in the current session.

```
RequestContext ctx = ((UxpServletRequest)servletRequest).getRequestContext();
String ip = (String) ctx.get("rtt.user.ip");
String country = null;
String line;
BufferedReader in = null;
try {
   in = new BufferedReader( new InputStreamReader(
              new URL("http://freegeoip.net/xml/"+ip).openStream ()));
   while ((line = in.readLine ()) != null) {
      if(line.contains("CountryName")) {
         country =
          line.substring(line.indexOf(">")+1,line.indexOf("</CountryName>"));
         break;
      }
   }
   ctx.getRequest().getSession().setAttribute("rtt.user.country", country);

} catch (IOException e) {
   LOGGER.error("A Problem occurred while getting the country!");

} finally {
   if (in != null) {
      try {
         in.close();
      } catch (IOException e) {
         LOGGER.error("A Problem occurred while reading the geoip service!");
      }
   } }
```

If an IP is to be predefined for testing purposes, this can be done using a manipulated request header for whose *X-FORWARDED-FOR* value the desired IP is predefined.

Alternatively, request parameters can also be used for the IP entry. In this case, the *ip* and *key* parameters must be added to the URL. The value for the *key* parameter is *GSJqcIQ5qXFbXiUaNyEk* and must not be changed. The desired IP must be entered as the value for the *ip* parameter.

6.1.1.3 Output in the template

In the template, the value set by the filter can be read out from the session using a JSTL expression; for example:

```
<c:out value='${sessionScope["rtt.user.country"]}'/>
```

6.2 Replacing the Profiling & Segmentation Service

To replace Woopra™ with a different Profiling & Segmentation Service, the new service must fulfill certain requirements:

- **Providing the defined target groups**

  The other provider's solution must have an interface that returns all defined segments. A GomIncludeValueProvider (*see section 4.1, page 31*) must be implemented for this interface. This mechanism makes it possible to define different content variants for the various segments.

- **Real-time queries**

  The other provider's Profiling & Segmentation Service must have an interface via which all of a website visitor's information can be queried at the time of a request in *real-time*. Membership to segments is a particularly important part of this information. The interface is addressed within the UXP via a filter to be implemented. Implementing additional filters is described in section 6.1.

- **Tracking mechanism**

  The other provider's solution must have a tracking mechanism in order to analyze the actions of the website visitor. This can be accomplished, for instance, using JavaScript or on the server side.

The implementation should be provided in the form of a FirstSpirit module which is installed on the FirstSpirit Server in addition to the *Real-time Targeting* module. In this case, the Woopra components of the *Real-time Targeting* module can serve as a reference.

6.2.1 Providing a list of all segments

The list of all segments is provided to the editors via the *GomIncludeValueProvider* interface to be implemented, which is documented in the FirstSpirit Access API:

http://www.espirit.com/odfs50/dev/de/espirit/firstspirit/access/store/templatestore/gom/GomIncludeValueProvider.html

This involves a default interface provided by FirstSpirit that is used to pass dynamic numbers of values to particular input components (*see section 4.1, page 31*).

It is recommended that communication is established between the FirstSpirit server and the desired Profiling & Segmentation Service by way of a FirstSpirit service, which in turn is addressed from *GomIncludeValueProvider*. In addition, a cache should be implemented for the returned target groups in order to prevent continuous communication between the server and the Profiling & Segmentation Service.

6.2.2 Filter for identifying a visitor's information

The website content output specific to the target group(s) is controlled via the segments assigned to the visitor. The Profiling & Segmentation Service supplies the segments at the request of the filter and they are passed to the visitor's session.

If another Profiling & Segmentation Service is to be used, a corresponding filter must be implemented that handles identification of segments *(see section 6.1, page 52*).

6.2.3 Tracking visitor behavior

Most providers analyze the behavior of the website visitor using a tracking code. This code must be integrated in the page template used within the project.

It is recommended that you implement a project component which imports a format template containing the tracking code into the project. This then only needs to be referenced via a CMS_RENDER call in the page template.

7 Tutorial

This tutorial uses examples to describe the steps necessary to integrate the *Real-time Targeting* components into an existing FirstSpirit project.

The RTT sample project is provided for this purpose and must be installed on the FirstSpirit server first. A project is also included in which all steps described in this tutorial have already been carried out.

The second step is to install the Real-time Targeting module on the FirstSpirit Server and then to configure it for the server and for the imported project. Next, the preview and ContentCreator components must be installed (*see chapter 3, Installation and Configuration*).

Once these steps are completed, the project templates can be adapted in the proceeding steps.

7.1 Page template

The page template consists of a fixed header and two fixed teaser sections. The body section of the page (contentcenter) is placed before the teaser sections, and in this example a dynamic section will be added to it which will display other content depending on the visitor's segments. Another body section comes next in which a contact form with SmartForms will subsequently be added.

7.1.1 Integrating taglibs

To achieve the functionality desired, tab libraries first need to be integrated into the header of the page template.

```
<head>
[...]
<%@taglib prefix="rtt" uri="/WEB-INF/targeting.tld"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
[...]
</head>
```

7.1.2 Integrating CSS and JavaScript

In addition, RTT module CSS and JavaScript files need to be included in the page template. The files only need to be included in the preview, since they are only used there. The *persona_script* format template contains the actual calls.

```
<head>
[...]
$CMS_IF(#global.preview)$$CMS_RENDER(template:"persona_script")$$CMS_END_IF$
[...]
</head>
```

7.1.3 Integrating the preview bar and personas

The preview bar is used to select personas in SiteArchitect. It simulates the MultiPerspectivePreview bar from ContentCreator. and should therefore only be displayed in the SiteArchitect preview. The display of the personas in the preview is then handled by the *persona_box* format template, both for ContentCreator and for SiteArchitect.

```
<body>
[...]
$CMS_IF(#global.preview && !#global.is("WEBEDIT"))$
  $CMS_RENDER(template:"persona_previewbar")$
$CMS_END_IF$
$CMS_IF(#global.preview)$
  $CMS_RENDER(template:"persona_box")$
$CMS_END_IF$
[...]
</body>
```

7.1.4 Creating personas

The desired personas in the Persona table need to be maintained in the data sources; for example:

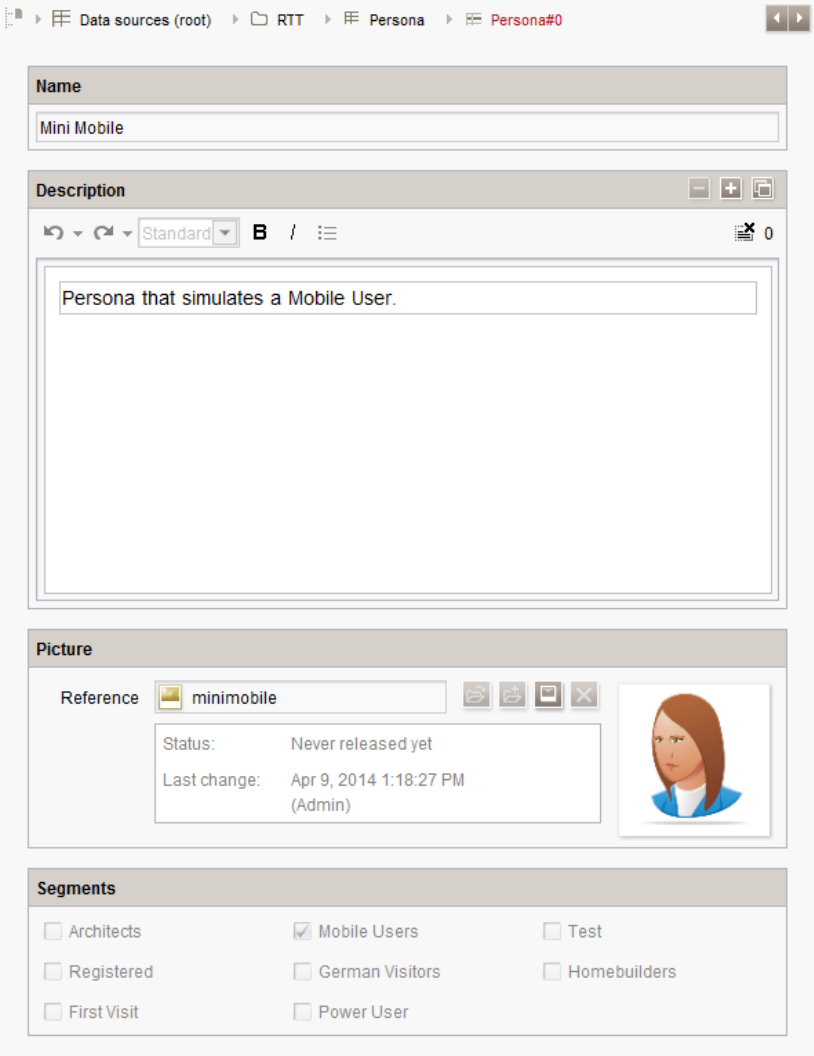

Figure 7-1: Adding a new persona

The personas are then already visible in the preview or in ContentCreator.

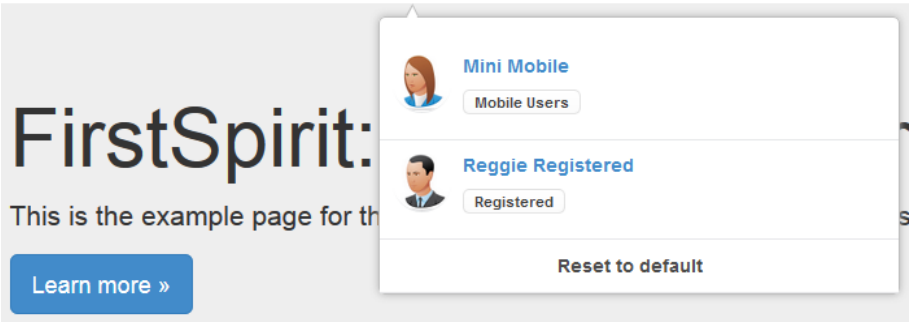

Figure 7-2: Persona Simulator

7.2 Section templates

To ensure that the created personas also affect a section, RTT functionality can be added to a new or existing section.

7.2.1 Teaser section

Since there is still no section in the sample project, we will add a new "Teaser" section and create the input components required for the headline and text. Two input components for the validity period as well as a list for variants are also added specifically for RTT.

```
<CMS_INPUT_DATE name="st_startdate">
<LANGINFOS>
  <LANGINFO lang="*" label="Startdate"/>
  <LANGINFO lang="DE" label="Startdatum"/>
</LANGINFOS>
</CMS_INPUT_DATE>

<CMS_INPUT_DATE name="st_enddate">
<LANGINFOS>
  <LANGINFO lang="*" label="Enddate"/>
  <LANGINFO lang="DE" label="Enddatum"/>
</LANGINFOS>
</CMS_INPUT_DATE>

<FS_LIST name="st_rtt_sectionlist" hFill="yes">
<DATASOURCE type="inline" useLanguages="no">
  <LABELS>
    <LABEL lang="*">
              if (#item.st_targetSegments.empty
              , if (#item.st_startdate.empty
              , if (#item.st_enddate.empty
              , "New variant"
              , "Valid to " + #item.st_enddate.format("MM/dd/yyyy")
              )
              , "Valid from " + #item.st_startdate.format("MM/dd/yyyy")
              + if(!#item.st_enddate.empty, " to "
              + #item.st_enddate.format("MM/dd/yyyy"))
```

```
            )
            , #item.st_targetSegments.map(x ->
            x.getLabel("%lang%")).toString(" " +

            #item.st_targetSegmentLogic.getLabel("%lang%") + " ") +
            if(!#item.st_startdate.empty, ", Valid from " +
              #item.st_startdate.format("MM/dd/yyyy")) +
            if(!#item.st_enddate.empty, if(#item.st_startdate.empty,",
            Valid") + " to " + #item.st_enddate.format("MM/dd/yyyy")))
      </LABEL>
    <LABEL lang="DE">
            if (#item.st_targetSegments.empty
            , if (#item.st_startdate.empty
            , if (#item.st_enddate.empty
            , "Neue Variante"
            , "Gültig bis " + #item.st_enddate.format("dd.MM.yyyy"))
            , "Gültig von " + #item.st_startdate.format("dd.MM.yyyy") +
            if(!#item.st_enddate.empty, " bis " +
            #item.st_enddate.format("dd.MM.yyyy")))
            , #item.st_targetSegments.map(x ->
            x.getLabel("%lang%")).toString(" " +
            #item.st_targetSegmentLogic.getLabel("%lang%") + " ") +
            if(!#item.st_startdate.empty, ",
            Gültig von " + #item.st_startdate.format("dd.MM.yyyy")) +
            if(!#item.st_enddate.empty,
            if(#item.st_startdate.empty,", Gültig") + " bis " +
            #item.st_enddate.format("dd.MM.yyyy")))
      </LABEL>
  </LABELS>
  <ACTIONS>
    <ACTION name="ADD"/>
    <ACTION name="REMOVE"/>
    <ACTION name="UP"/>
    <ACTION name="DOWN"/>
    <ACTION name="EDIT"/>
  </ACTIONS>
  <COLUMNS>
    <COLUMN show="no">#identifier</COLUMN>
    <COLUMN show="yes" width="150">#text</COLUMN>
  </COLUMNS>
```

```
  <LAYOUT>
    <ADD component="stackedview" constraint="hide"/>
    <ADD component="toolbar" constraint="top"/>
    <ADD component="overview" constraint="left"/>
    <ADD component="simpleview" constraint="center"/>
  </LAYOUT>
  <TEMPLATES source="sectiontemplates">
    <TEMPLATE uid="rtt_variant_text_image"/>
  </TEMPLATES>
</DATASOURCE>
<LANGINFOS>
  <LANGINFO lang="*" label="Variants"/>
  <LANGINFO lang="DE" label="Varianten"/>
</LANGINFOS>
</FS_LIST>
```

In the presentation channel, additional information is added to two items in the standard output. First, each variant requires a unique ID which we will generate from the *rttSection* tag and the ID of the section.

```
$CMS_SET(rttSection, "id=\"rttSection" + #global.section.id + "\" ")$
```

This ID will later be set as the ID of the Div that displays the teaser.

```
<div $CMS_VALUE(rttSection)$class="col-md-4"$CMS_VALUE(editorId())$>
```

The next step is to place the JSTL tag *rtt:showContent* around the teaser content in order to show or hide it depending on the user's segments. The ID and validity periods must be passed as parameters.

```
<rtt:showContent id="$CMS_VALUE(#global.section.id)$"
$CMS_VALUE(setStartDate)$$CMS_VALUE(setEndDate)$referencedate="$CMS_VALUE(#global.startT
ime.getTimeInMillis())$">
```

7.2.2 Teaser variants section

The actual variants will have a custom section template that is a copy of the previous teaser section. In the copy, the input component for the variants is removed and instead two new input

components are added for the segments and their linking logic.

```
<CMS_INPUT_RADIOBUTTON name="st_targetSegmentLogic" gridHeight="1" gridWidth="2"
hFill="yes" useLanguages="no">
<ENTRIES>
  <ENTRY value="OR">
    <LANGINFOS>
      <LANGINFO lang="*" label="OR"/>
      <LANGINFO lang="DE" label="ODER"/>
    </LANGINFOS>
  </ENTRY>
  <ENTRY value="AND">
    <LANGINFOS>
      <LANGINFO lang="*" label="AND"/>
      <LANGINFO lang="DE" label="UND"/>
    </LANGINFOS>
  </ENTRY>
</ENTRIES>
<LANGINFOS>
  <LANGINFO lang="*" label="Logic to combine segments" description="Should segments be
combined with OR or AND?"/>
  <LANGINFO lang="DE" label="Segmente verknüpfen mit" description="Mit welcher Logik
sollen die Segmente verknüpft werden?"/>
</LANGINFOS>
</CMS_INPUT_RADIOBUTTON>

<CMS_INPUT_CHECKBOX name="st_targetSegments" gridWidth="3" hFill="yes"
useLanguages="no">
<CMS_INCLUDE_OPTIONS type="public">
  <LABELS>
    <LABEL lang="DE">#item</LABEL>
    <LABEL lang="*">#item</LABEL>
  </LABELS>
  <NAME>com.espirit.moddev.rtt.gui.WoopraValueProvider</NAME>
</CMS_INCLUDE_OPTIONS>
<LANGINFOS>
  <LANGINFO lang="*" label="Show to segment(s)" description="Information is relevant and
should be shown to the selected segments."/>
  <LANGINFO lang="DE" label="Relevanz für Segmente" description="Informationen sind
relevant und sollen den ausgewählten Segmenten angezeigt werden."/>
```

```
</LANGINFOS>
</CMS_INPUT_CHECKBOX>
```

In the presentation channel, additional information is added to multiple items in the standard output. First, each variant requires a unique ID which we will generate from the *rttSection* tag and the ID of the section and add to the surrounding Div.

```
<div id="rttSection$CMS_VALUE(#global.section.id)$" [...]
```

In this case as well, the JSTL tag *rtt:showContent* is placed around the teaser content in order to show or hide it depending on the user's segments. In addition to the ID and validity periods, the linking logic of the segments and the selected segments themselves must be passed as parameters.

```
<rtt:showContent segments="$CMS_VALUE(st_targetSegments.toString(","))$"
logic="$CMS_VALUE(st_targetSegmentLogic)$" id="$CMS_VALUE(#global.section.id)$"
$CMS_VALUE(setStartDate)$$CMS_VALUE(setEndDate)$
referencedate="$CMS_VALUE(#global.startTime.getTimeInMillis())$">
```

After completing these tasks, new sections with variants can be added to the page and using the Persona Simulator, the editor can test in the preview the perspective of users with particular segments.

7.3 Setting up Woopra™ tracking

If new pages are also going to be tracked by *Woopra*™, the following line must be added to the end of the body in the page template described in *section 7.1*.

```
$CMS_RENDER(template:"woopra_tracking", woopra_domain:"mydomain.com")$
```

The domain in this case must be adapted to the custom domain that is to be tracked.

7.4 Scoring setup

In the next step for the page, the option is to be integrated in which a score is specified that later makes it possible to assign the user in *Woopra*™ to specific segments based on the user's score.

To ensure that multiple, freely definable attributes and values can be assigned to each page, we will use an FS_LIST input component in this case. In order to be able to fill it with any values

required, we will first add a new section template that provides selection and input options for the score name and the score value.

```
<CMS_INPUT_COMBOBOX name="st_score_name" noBreak="yes" useLanguages="no">
<ENTRIES>
  <ENTRY value="homepage_score">
    <LANGINFOS>
      <LANGINFO lang="*" label="Homepage"/>
    </LANGINFOS>
  </ENTRY>
</ENTRIES>
<LANGINFOS>
  <LANGINFO lang="*" label="Score Name" description="Name for rtt scoring."/>
  <LANGINFO lang="DE" label="Score Name" description="Name für RTT score."/>
</LANGINFOS>
</CMS_INPUT_COMBOBOX>
```

We will then reference this in the FS_LIST input component in the page template.

```
<FS_LIST name="pt_scoring" hFill="yes" rows="10">
<DATASOURCE type="inline" useLanguages="no">
  <LABELS>
    <LABEL lang="*">#item.st_score_name + "\t\t (Score: " + #item.st_score_value +
")"</LABEL>
  </LABELS>
  <ACTIONS>
    <ACTION name="ADD"/>
    <ACTION name="REMOVE"/>
    <ACTION name="UP"/>
    <ACTION name="DOWN"/>
    <ACTION name="EDIT"/>
  </ACTIONS>
  <COLUMNS>
    <COLUMN show="no">#identifier</COLUMN>
  </COLUMNS>
  <LAYOUT>
    <ADD component="toolbar" constraint="top"/>
    <ADD component="overview" constraint="center"/>
    <ADD component="simpleview" constraint="bottom"/>
  </LAYOUT>
```

```
  <TEMPLATES source="sectiontemplates">
    <TEMPLATE uid="scoring"/>
  </TEMPLATES>
</DATASOURCE>
<LANGINFOS>
  <LANGINFO lang="*" label="Score" description="Add score."/>
  <LANGINFO lang="DE" label="Score" description="Fügen Sie einen Score hinzu."/>
</LANGINFOS>
</FS_LIST>
```

As the last step, the score needs to be added to the *Woopra*™ tracking code set up in *section 7.3*. For this purpose we write the values selected in the input components to the format template's tracking code (*woopra_tracking*).

```
<script>
[...]
$CMS_SET(hasScore, false)$
$CMS_SET(scores, "")$

$CMS_IF(pt_scoring != null)$
    $CMS_FOR(entry, pt_scoring)$
       $CMS_IF(entry.st_score_value > 0)$$CMS_SET(hasScore, true)$$CMS_END_IF$
     $CMS_IF(#for.index > 0)$$CMS_SET(scores, scores + ",")$$CMS_END_IF$
     $CMS_SET(scores, scores + entry.st_score_name + " : " + entry.st_score_value)$
    $CMS_END_FOR$
$CMS_END_IF$

woopra.track();

$CMS_IF(hasScore)$
     woopra.track('scoring', {
       $CMS_VALUE(scores)$
     });
$CMS_END_IF$
</script>
```

The respective element's score is then increased for each page visitor by the configured value. Additional rules can be created in *Woopra*™ in order to assign the visitor to a segment starting from a particular value (*see section 5.2.2*).

7.5 Setting up SmartForms

To add a *SmartForm* to the page, we will use the FirstSpirit *FormEdit* module as the basis. For this purpose, it must be installed in accordance with the installation instructions, and the templates need to be imported to the project. Alternatively, the *SmartForms* can also be used without *FormEdit*, in which case only the form framework will need to be generated using another method. Using *SmartForms*, the visitor's name and e-mail are to be filled out automatically when the form in this example is called again.

7.5.1 Creating a form

The first step is to create a new form in the *contentbottom* content area by adding *form start*, *form block* and *form end* sections. The text blocks *name* and *email* are added inside the *form block* element. The unique identifiers are particularly important in order for the scripts of the *SmartForms* to function properly.

7.5.2 "FormStart" template modifications

Information needs to be added to the "FormStart" template in order to use *SmartForms*. Using the added script, the visitor's "name" and "email" values are forwarded to *Woopra*™ before the form is sent.

```
<script>
	var submit_$CMS_VALUE(fr_st_name)$ = false;
	$('#$CMS_VALUE(fr_st_name)$').submit(function(e) {
		var formName = $('#name').val();
		var formEmail = $('#email').val();
		woopra.identify({
			email: formEmail,
			name: formName
		}).push();
		setTimeout(function(){
			submit_$CMS_VALUE(fr_st_name)$ = true;
			$('#$CMS_VALUE(fr_st_name)$').submit();
		}, 2000);
		if(!submit_$CMS_VALUE(fr_st_name)$) {
			e.preventDefault();
		}
	});
</script>
```

7.5.3 "formText" template modifications

To pre-populate the input components for "name" and "email" with values from Woopra™ when calling the form again, more information needs to be added to the *formText* template. If the user has entered information, this value is displayed; otherwise, the stored value from *Woopra*™ is displayed. This value is already in the user session due to the defined filter (see *web.xml modifications* in the next section).

```
$CMS_SET(fr_st_value)$
  $CMS_IF(!st_value.isEmpty)$
    $CMS_VALUE(st_value)$
  $CMS_ELSE$
    <c:out value="${sessionScope['rtt.woopra.user.$CMS_VALUE(st_name)$']}"/>
  $CMS_END_IF$
$CMS_END_SET$
```

7.5.4 web.xml modifications

Within the project properties for the respective web component, the *web.xml* file of the *Woopra*™ web application needs to be modified by adding the *WoopraFirstVisitFilter* and *GetWoopraUserInformationFilter* pipeline filters to it (*see sections 3.5.2.3 and 3.5.2.4)*. These filters allow for identification of the attributes saved for a visitor in Woopra™.

8 Legal notices

The *Real-time Targeting* module is a product of e-Spirit AG, Dortmund, Germany.

Only a license agreed upon with e-Spirit AG is valid with respect to the user for using the module.

Details regarding any third-party software products in use but not created by e-Spirit AG, as well as the third-party licenses and, if applicable, update information can be found in the file "third-party-dependencies.txt" included with the module.

9 Disclaimer

This document is provided for information purposes only. e-Spirit may change the contents hereof without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. e-Spirit specifically disclaims any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. The technologies, functionality, services, and processes described herein are subject to change without notice.