Concept - Personalization

Personalization

Scope for 5.3

First steps for personalization are

- Implicit personalization (unknown person, e.g. geo-location, device)
- Explicit personalization (known user with profile)
- Segmentation of visitors based on categories, visit data, user profiles
- Serve content based on segment
  - Using dynamic components / page variants
- Pages/components show contents from pools
- Great previews
- Simple impersonation

What it will not be in

- No behavior analysis
- Complex content tools
- Better editors for authors

Terminology

Trait: One aspect/criterion of the personalization, i.e. time of day, user profile, location, category or a combination of the before. A trait should/will be taggable to any content.
Constraint: A condition where a trait has a specific value (or value range).
ConstraintSet: A condition where multiple traits have a specific value (or value range).
Segment (~~audience~~): A single contraint or a set of constraints.
Segmentation: The process of combining/assembling traits into a segment.
(Page) Variant: A variant of a page associated with a segment. Such a page will always have a master-page.

Variation (template variation): A variation of a template, see: Configuring variations in a page template.

Implementation

Trait

- Most atomic facet/characteristic/property relevant for segmentation
- Will consist of a POJO **class**
- **TraitDetector** will implement a **Filter** and will reside somewhere in the **filter chain** (depending on needs)
    - Stores instance of Trait class in TraitCollector
        - or to put it differently: a **Value** of a **Trait**
    - One seperate filter per trait (and not a delegating super filter)
- **TraitVoter** decides whether current request "matches"
    - evaluator for traits (more specific: for trait classes)
    - might extend AbstractTraitVoter (that requires TraitCollector)
    - is used to "build" a **constraint**

Example trait

- **Trait** Country
- **TraitDetector** in FilterChain will use GeoIP database to resolve country of visitor by his IP address
- Will store *Country("ch")* in TraitCollector
- **TraitVoter** (a **constraint**) returns true if country in TraitCollector is "ch", false if it's another country

Notes

Possible default TraitVoters:

- Current user's preferences for categories matches selected categories on page
- Current visitor's device matches channels of page/page variant

Questions

How to persist voters in config:

- info.magnolia.ui.form.field.transformer.Transformer

(UI) elements of a traits

- an associated UI element (custom control) to create a **trait voter** (constraint)
    - to assemble them in a ConstraintSet or segment
    - to target contents for this specific category
- and another UI element (custom control) to create a **value** of a **trait** (for impersonation)
- relevant input of user is a value or an interval of a specific type (date, string, long) (operator: = < > <= >= ≠)
- label that gives an idea of what the trait voters "evaluates" String.format("Country ='%s'; %s", "ch", "Switzerland")
    - use i18nIzer?

TraitCollector

- Simple map to "collect" all trait instances of one request/visit
    - This map can only contain one instance of a specific trait

- See: Concept - Personalization - AggregationState vs. @Injected + @LocalScoped new object (TraitCollector)

ConstraintSet

- A set of (multiple) contraints
- Consists of a **VoterSet** and thus multiple **Voters**
- **ConstraintSet** will be stored at page variant's level (for ease of activation)
- **ConstraintSetRegistry** registers all ConstraintSets upon module startup
- Are managed by the system (in contrast to a segment, which is user-manager in a SegmentationApp)

Example

- Adding a date constraint to a page that is already associated with a segment (i.e. campaign page)
  - Will result in a ConstraintSet (segment AND date)

Notes

- We need a register for ConstraintSets (Segments) (so we only have to Node2Bean' them on startup)
- We need a "cache map" URI–ConstraintSets (for us to be able to evaluate only relevant ConstraintSets, without hitting JCR, before Cache)

Segment

- Collection of shared traits (might be just one)
- Is a **ConstraintSet**(extends)
  - But has a **name** & **description**
- Can be considered as an "audience"
- **SegmentEvaluator** will choose if current traits in TraitCollector match this segment
  - Uses combination (and/or/not/etc.) of TraitVoters
- **SegmentManager** will return a list of available segments
  - getSegments()
  - getSegmentByPage(Node node)
  - getSegmentBySite(Site site)?
  - getAvailableSegmentsByPage(Node node)
- System-managed vs. User-managed segements
  - **User-managed**: user creates segments in Segmentation App (editable, visible)
  - **System-managed**: implicit **ConstraintSets** created at-page-level (i.e. by adding an additional trait, for example date: visible from-to)

Example segment

- **Segment** Swiss users that are logged in
- Consists of two **TraitVoters (ConstraintSet)** (country = "ch" AND user-is-logged-in)
- **SegmentEvaluator** will return this segment, if user's IP address is swiss and he is logged in

Notes

- Segments are stored as flat lists with the possibily to structure them in **folders** (mgnl:folder)
- **Multiple segments** on single **page variant** are possible and evaluated with **OR**?
  - If additonal constraint is added (= new ConstraintSet on page level), evaluation happens with AND: ((segment1 OR segment2) AND constraint)
- ConstraintSets are stored on page level
- Segments are stored in a seperate workspace (manageable by the SegmentationApp)

Questions

- Activation: how to activate dependencies?

Segmentation

- The task of creating segments

Basics

- Will be done in **SegmentationApp** (**SegmentManagementApp)**
- Custom UI for App & Dialog
- Dialog will be customizable just like any other Magnolia dialog
    - Consists of TraitVoter (Constraint) UI elements
- Segments will be "structurable" in folders
- UI for SegmentationApp and segmentation dialog is currently planned by Andreas

Actions in SegmentationApp

- Create/edit segment
- Create/rename folder

New UI elements in SegmentationApp

- tbd

Variants

- Any content can have a variant
- **Scope for now** variant of a page
- Basics applicable for any content, App+UI bound to page variants

Basics

- A variant of a page associated/connected with a segment
- Prefixed (i.e. **variant-#**) subnode of master page
- Mixin: **mgnl:variant** (thus, can be used for other contents in the future too, not just mgnl:page)
- Properties (both mandatory):
    - **mgnl:variationOf**: single-valued reference (weak reference?) and
    - **mgnl:assignedSegments**: multi-valued String or
    - **mgnl:constraintSet**: sub node with constraint set attached
- Creating a page variant delegates to interface
    - For 5.3 creates a **full copy** of the master
    - Future: might be using "real" JCR references
- A page variant can have multiple segments whereas the master page will never have a segment
- In a first stept we will **not support nested page variants**

Actions in PageEditor

- Creation/modification of page variants will done in PageEditor
- Tree View
    - Create page variant
        - We need a **separate action** as it should be possible to **target pages** to specific traits **without creating a segment**
            - creates ConstraintSet at page-variant-level
    - Assign/add segment
        - Will show available segments to choose from
        - create variant of master page
        - and store selected segment in variant
- Edit View:
    - Change segment(s)
    - Assign Segment(s) (add segment to page variant)
    - Delete page variant

Notes

Add segment to page variant is either

- selecting one segment from a list or (a user-managed segment)
- adding another constraints (from a reduced list of constraints)
  - choosing one of these will create a ConstraintSet on page level
  - this dialog differs to the one used in the SegmentationApp

New UI elements in PageEditor

- New UI element **VariantSwitcher**
  - Requires **SegmentManager**
  - Above green edit bar in PageEditor
  - Edit mode:
    - Screenshot: 
    - Switch inbetween master and variant pages
    - Will only show segments linked to page variants
  - Preview mode:
    - Screenshot: 
    - Is sticky, i.e. will always be displayed on top of edit bar even if page doesn't have a page variant
    - Shows all available segments in system to be able to preview all of them

Dynamic content elements/components

- Custom component models that query content by
  - getting the relevant trait in the TraitCollector and
  - searching (JCR-Query-2) for elements that are "tagged" with value
  - i.e.: **SELECT** e.* **FROM** [mgnl:events] **AS** e **WHERE** e.tags **IN** ('tag-A', 'tag-B', 'tag-C');
- Tagging will happen just like selecting a (multiple) category(ies) and adding it to an element

Rendering

- Introduce new **ContentDecorator**which is Personalization-aware
  - This includes a **NodeWrapper** and a **PropertyWrapper**
    - both completely "hide" whether a master page or a variant is currently being rendered
  - Uses **SegmentEvaluator** to get the best-suiting segment
  - And will use **VariantChooser** to select the appropriate page variant (subnode of master-page linked to segment) and wrap it
- (Freemarker/STK/*)Renderer renders page variant

Notes

Possible default VariantChoosers:

- SegmentVariantChooser
- CategoryVariantChooser
- TimeOfDayVariantChooser
- UserAgentVariantChooser (always selects master page when user agent is bot)

Preview

- Allows to preview a page in all its available variants or, put it in a different way, in all the segments assigned to that page.
- Works in PageEditor while in preview mode.
- See: New UI elements in PageEditor

Impersonation

- Browse/preview site/pages "as if"
- So called **Personas** usually match a segment
- Simpler impersonation will be achieved by setting **ValueSets** of traits (this could be just one **value**: i.e. the date, see Trait)
- Storing trait instances in TraitCollector (**PreviewFilter**) and overriding previously detected ones
- Separate App/View

Questions

- "Fake"
  - Switch constraints in TraitCollector
  - might result in strange behavior: seeing stuff as *eric* with JCR rights of *superuser*
- "Real"
  - switch MgnlUser (user "eric")
  - wrap/swap context (more specific to preview): PreviewContext
  - Is JCR security an issue ("subject"?)

Cache

See Personalization and Cache.

Proposed module structure

- **module-personalization** (reactor)
  - **module-personalization-integration**
  - **module-personalization-samples**
  - module-personalization-stk

groupId: info.magnolia.personalization

Known limitations

Reverse proxy / third party caching with personalization might be an issue.

Stories

See BL-206 - Getting issue details... STATUS

Appendix

Relation of traits, constraints(sets), value(sets) and segments

Segmentation / Campaigns

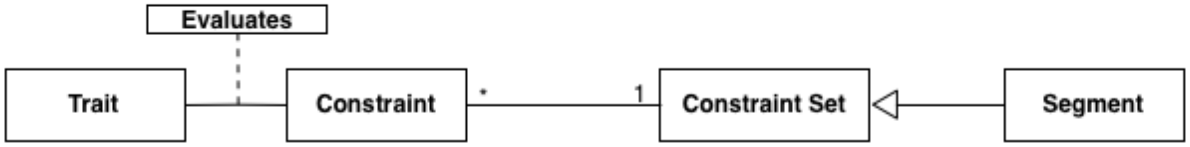
Country, UserInfo, UserProfile

country = CH, user-logged-in

country = CH **AND** user-logged-in

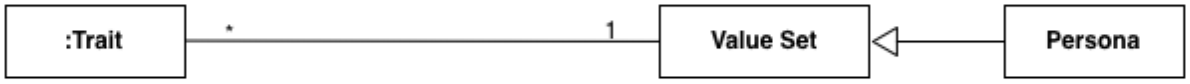
swiss user with a profile

Implementation



Impersonation

Implementation



country = CH

country = CH **AND** user-logged-in

swiss user with a profile