Concept - Personalization - Trait Registry

Introduction

The initial concept doesn't comprise of a central trait registry which contains all information regarding a trait (i.e. fields, transformers, formatter, etc.).

Trait Registry

Why?

- "Dynamic" dialogs (similar to configuration on the fly)
- Everything regarding a trait in on place, rather than distributed of mulitple configuration locations
- Add traits on-the-fly (e.g dialogs)

Where?

- In any module under the node
    - traits/

How?

- Trait will have a TraitDescriptor + ConfiguredTraitDescriptor
- Use the "default" registry mechanism to regsiter additions/modification

TraitDescriptor

A TraitDescriptor should contain:

- constraintField
    - FieldDefinition
        - To edit the actual contraint (Voter)
    - Transformer
    - voterClass=info.magnolia.trait.CountryVoter
- valueField
    - FieldDefinition
        - To edit the value of a trait
- formatter
- previewParameter
    - converters for preview
    - enables preview of trait in Preview App
- traitClass=info.magnolia.trait.Country
    - The actual class (POJO) of the trait

Improvements

Adding the voterClass to the constraint's FieldDefiniton would simplify the dynamic ChooseDialog (We have to read from JCR-level and have to know which field to choose and display in the dialog. The only thing stored in JCR is the voter class).

```
public interface P13nFieldDefintion {
    Class<?> getVoterClass();
}
```

```
ConfiguredP13nFieldDefintion extends FieldDefinition implements P13nFieldDefintion {
        private Class<?> voterClass;
        public Class<?> getVoterClass() {
                return voterClass;
        }
    public void setVoterClass(Class<?> voterClass) {
        this.voterClass = voterClass;
    }
}
```

Result:

info.magnolia.p13n.trait.BaseTraitTransformer#getVoterType() could go and we read voter class from P13nFieldDefinition