

Automatic generation of web CRUD applications



Supervisor: Marco Brambilla

Context



More than 80% people have access to the web and web applications are replacing old software.

Different mobile frameworks rely on web development.

Generators provide complex code, require too much configuration and are not based on the newest technologies.

Goals



- ☐ Code's simplicity
- ☐ Lack of configuration
- ☐ Rapid development
- ☐ Automatic documentation & testing
- ☐ Coherence and standardization

Input



- ☐ Database schema
- ☐ Set of Java classes with custom annotations

Transformation



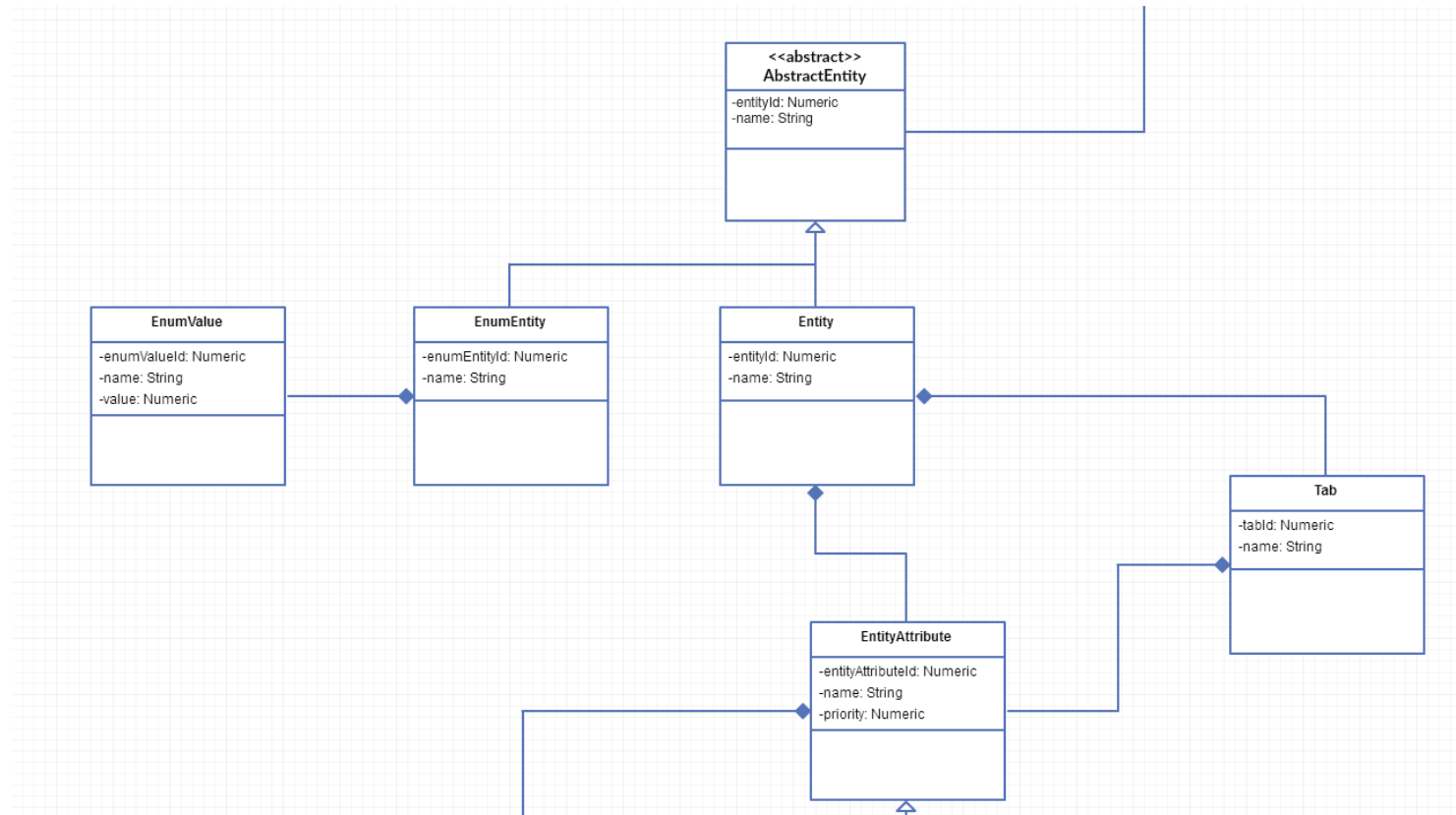
The input model is transformed into an instance of the designed metamodel.

☐ Project

☐ Entity

☐ Field

☐ Security



Output

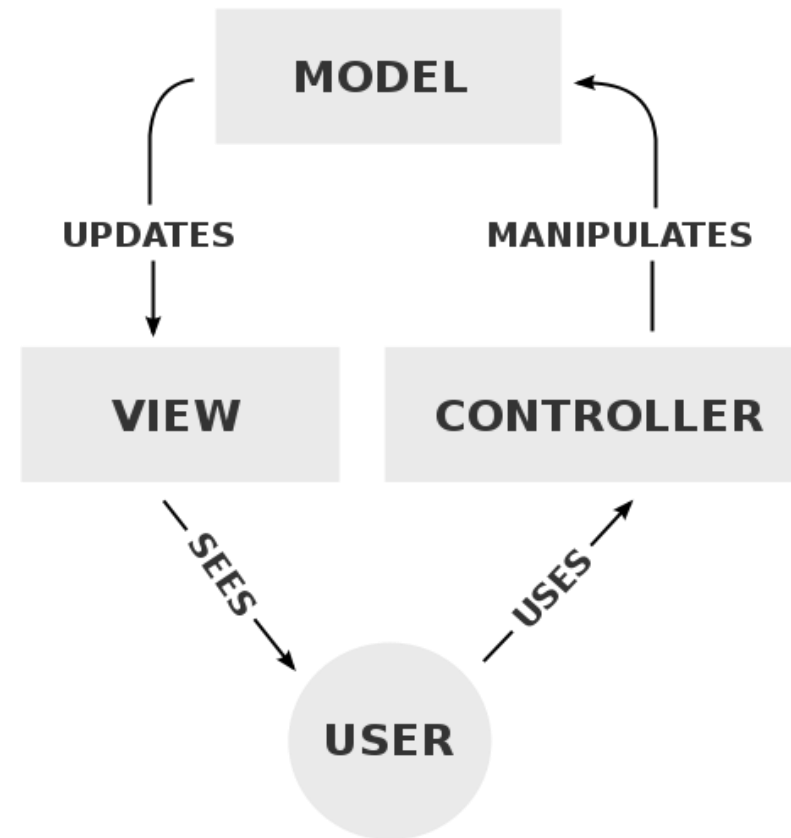


Server side [Spring Framework]

- ☐ Repository layer's component
- ☐ Service layer's component
- ☐ Controller layer's component

Client side [AngularJS]

- ☐ Service component
- ☐ Controller component
- ☐ Template component



Focus on: annotation



Metamodel contains the structure for additional information that helps the generation.

Annotations are the Java input for this data.

Entity: @Cache, @MaxDescendantLevel, @IgnoreMenu,...

Field: @Between, @DescriptionField, @Tab,...

Validation: @Size, @NotBlank, @NotNull



Focus on: security



Generated application support different levels of security

- ☐ Multirole management
- ☐ Access level

Access level are defined towards entityGroup, entity and field, in a hierarchy solution.


The framework provides two different policy

- ☐ Block with restriction
- ☐ Access with permission

Focus on: documentation

The framework adopted for documentation is Swagger, which grants:

- ❑ Textual description of each REST interface
- ❑ Technical description of each JSON
- ❑ Possibility to test each REST API

 **Apollo REST API Explorer**

/api/docs/session [Show/Hide](#) [List Operations](#) [Expand Operations](#) [Raw](#)

GET	/api/json/session/whoami	Returns the Signed-in user's principals
GET	/api/json/session/signin	Signin a user
POST	/api/json/session/signin	Signin a user
Parameters		
Parameter	Default Value	Description
username	-	User Name
password	-	User Password
GET	/api/json/session/signout	Signout a user
GET	/api/json/session/signout	Signout a user

Focus on: self generation



Designed metamodel can be represented as an instance of itself.

This means that, after the first core's implementation, it has been possible to automatically generate a web interface towards the metamodel.

This allows developers (and non-developers..) performing CRUD operations on their domain problems's instance.

Focus on: continuous generation



The key idea is to allow modifying and re-generating just a part of the project.

Re-generation must be able to keep track of custom code added by developers.

It has been implemented by means of Git API at runtime.

Test case: EBSN-Backoffice



EBSN-Backoffice is a management software that provides CMS and customer care's feature for an Ecommerce platform.

It is built on top of a PostgreSQL database of more than 200 tables.

The framework allowed generating it in a few hours.

Test case: EBSN-Storepicking



EBSN-Storepicking is a warehouse management systems that keep track of orders, products and other different aspects of a warehouse.

It is built on top of a Mysql database of 25 tables.

The framework generated a first version of this software after few hours of work.

Results



The implemented framework provided good results in a real world environment.

It generated a basic version of an application, with easy-to-read code, a precise scaffolding and the related documentation.

Generated application is provided with a basic graphic layout that would require some time to be customized.

The end



Thank you