etnetera

Formio

Need a form?

Radek Beran

27.3.2014

Motivation

- I hate solving of repetitive problems where it is not necessary
- I want to solve them in a simple way
- I want to have an easy to use tool that i can use anywhere…

Formio

- Form definition & binding framework
- Easy-to-use configurable handy tool
- Validation of form data (using bean validation)
- Support for file uploads, configurable max. request/file size
- Immutable, composable form definitions
- One simple entry point to API
- Easy integration to existing frameworks
- For environments with or without HttpServletRequest
- Simply unit testable forms

Simple form example

```
private static final FormMapping<Person> personForm =
Forms.automatic(Person.class, "person").build(/* Opt. config */);
…


FormData<Person> formData = new FormData<Person>(person, null);
FormMapping<Person> filledForm = personForm.fill(formData);
// push filledForm to template …
…


FormData<Person> formData = personForm.bind(
  new HttpServletRequestParams(request));
if (formData.isValid()) {
  // save person - formData.getData() …
} else {
  // …
}
```

Formio API

- **Forms – entry point**
  - FormMapping, BasicFormMappingBuilder
  - FormField, FieldProps for field definition
- **ParamsProvider**
- **ValidationResult**
  - ConstraintViolationMessage
- **Config**
  - Formatters, Formatter
  - CollectionBuilders, CollectionBuilder,
  - BeanExtractor
  - Binder, Instantiator, ArgumentNameResolver
  - BeanValidator
  - PropertyMethodRegex
- **UploadedFile**

Data binding

- **Setter and construction method binding**
- **Immutable objects supported (Instantiator abstraction)**
  - Default or non-default constructors
  - Static factory methods
- **Collections and arrays**
- **Complex nested objects and lists of complex objects**
  - And arbitrary combination recursively
- **Primitives (and their wrapper classes) supported everywhere**
- **String, Enums, BigDecimal, BigInteger, Date supported by default formatters**

Data for templates

- **Just push the filled form to the template**
- **Use its properties**
  - **validationResult**
    - success
    - fieldMessages
    - globalMessages
  - **fields** – Map<String, FormField>
  - **name** – unique name/path of mapping
  - **labelKey** – key for caption of nested complex object carried by mapping
  - **filledObject**
  - **required**
  - **nested** – map with nested mappings
  - **list** – list of nested mappings for complex objects if this is MappingType.LIST mapping

Data for templates (2)

- **Properties of FormField**
  - **name** – unique name/path of form field
  - **labelKey** – key for localization of label (name without brackets)
  - **filledObject**, **filledObjects** (for group of checkboxes, multiselect)
  - **required**

Form definition

```
private static final FormMapping<RegDate> regDateMapping =
  Forms.basic(RegDate.class, "regDate").fields("month", "year")
  .build();

private static final FormMapping<Registration> registrationForm =
  Forms.basic(Registration.class, "registration")
    // whitelist of properties to bind
    .fields("attendanceReasons", "cv", "interests", "email")
    .nested(Forms.basic(Address.class, "contactAddress",
      Forms.factoryMethod(Address.class, "getInstance"))
      .fields("street", "city", "zipCode").build())
    .nested(Forms.basic(Collegue.class, "collegues",
      null, MappingType.LIST)
      .fields("name", "email")
      .nested(regDateMapping)
      .build())
  .build();
```

Form definition (2)

Or equivalent automatic one

```
private static final FormMapping<Registration> registrationForm =
  Forms.automatic(Registration.class, "registration")
    .nested(Forms.automatic(Address.class, "contactAddress",
      Forms.factoryMethod(Address.class, "getInstance")).build())
  .build();
```

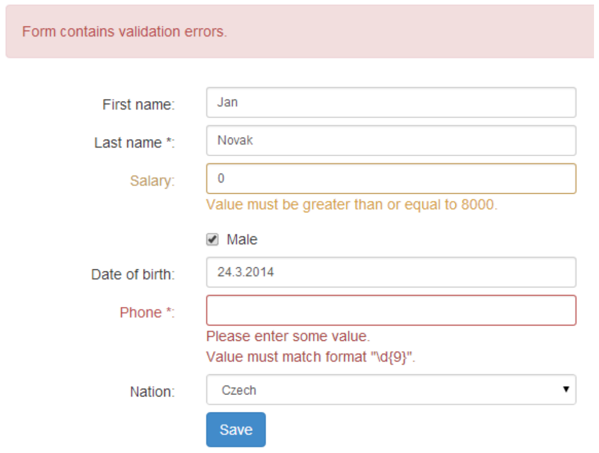
- **Automatic mapping**
  - Introspects readable (and settable – via setters/instantiators) properties (incl. nested objects, list of nested objects)
  - @Ignored on getters can be used
  - Custom field definitions have precedence
- **Custom field definition**
  - new FieldProps(propertyName, pattern, formatter)

Validation

- **Using bean validation API (**JSR-303**)**
- **ValidationResult**
  - ConstraintViolationMessage(s) for global and field constraints
  - Fields (properties) and also mappings (whole complex objects) are validated
  - Binding and validation errors
- **Custom message bundle can be configured**
  - Default are properties for class of form data
- **Fallback to ValidationMessages.properties with default messages**
- **Message bundle need not to be used**
  - Error messages carry possible translation and also unresolved key with arguments

Other validation features

- **Conditional validation**
  - Using bean validation groups
- **Custom validators can be written**
  - NotEmpty
  - RodneCislo
  - Email
  - CustomMultiFieldValidator, …
- **Error, warning, info severity supported and recognized from payload of annotations**
- **Required flag automatically filled from Size, NotNull and derived constaints in default validator implementation**
  - For FormFields
  - For mappings of complex objects



Form contains validation errors.

First name:	Jan
Last name *:	Novak
Salary:	0
	Value must be greater than or equal to 8000.
	✓ Male
Date of birth:	24.3.2014
Phone *:	
	Please enter some value.
	Value must match format "\d{9}".
Nation:	Czech

Configuration

- Config object (Created using Forms.config()…build())
- Immutable
- Default configuration available
- Custom configuration can be supplied
- Automatically propagated to nested mappings

File uploads

- Using Apache FileUpload API
- Max. request size, size of one uploaded file, threshold for storing data in memory can be set via Formio API
- Global/field ConstraintViolationMessages created when limits are exceeded
- Automatic binding to UploadedFile, collections/arrays of UploadedFile(s)
- UploadedFileWrapper as a complex object for indexed lists of uploaded files (nested list mappings)

What about AJAX?

- **AJAX compatible**
  - You can expose server methods/API for update of state called from client
  - On the server: Form definition can be filled with current data updated with sent data
  - Filled nested mapping (inc. its validation result) can be sent back to client
- **Cooperating AJAX framework can be used**
  - Twinstone TDI, jQuery, …

Formio vs. Spring

- **Formio Pros:**
  - Even easier to learn and use than Spring
  - Functional form processing, immutable/composable/reusable form definition
  - Immutable view/domain objects can be used (via constructors, static factory methods), arbitrarily nested – no need for custom Spring MVC WebArgumentResolver(s) or HttpMessageConverter(s)
  - Minimum of dependencies (bean validation, fileupload)
  - Message bundle for form data class *with fallback* to common ValidationMessages
  - Not bound to any particular UI architecture like MVC, but fully usable in Spring MVC or other frameworks
  - Automatic construction of form field names/paths, also for nested objects/lists
- **Cons:**
  - Spring is widely used and offers usable solution in Spring MVC

Further development

- Available in Maven central
- Intensive coverage with unit tests
- Getting started documentation
- Static/dynamic definition of flags
  - Readonly, disabled, …
- ArgumentProvider (from ArgumentSource),
- Example TFS parts for form fields
- Filter of suspicious user input for ParamsProvider

Q & A?

Demo: http://formio-demo.herokuapp.com/, https://github.com/beranradek/formio-demo