Bean Validation: Practical Examples from a Real-World Java EE 7 Application

The Institute *for* Quantitative Social Science at Harvard University

Presenters: **Gustavo Durand** and **Stephen Kraffmiller**

Visit Data Science (IQSS) at: datascience.iq.harvard.edu/

Session Abstract

JSR 303 / 349 introduced Bean Validation for providing a facility for validating objects, object members, methods, and constructors. In Java EE environments, Bean Validation integrates with Java EE containers and services to enable developers to easily define and enforce validation constraints, enabling the rules to be defined once, rather than wherever they need validation (such as at the UI or API layer). This session briefly introduces the standard concepts of Bean Validation annotations and demonstrates some practical techniques for more-complex needs, such as using custom validators and defining cross-field validation. Finally, it discusses how the speakers and their colleagues incorporated Bean Validation for the dynamic, data-driven rules of their application.

Outline

1. Standard Bean Validation

The Bean Validation model is supported by constraints in the form of annotations placed on a field, method, or class of a JavaBeans component, such as a managed bean. All Java EE 7 Built-in Constraints may be found here [Java EE 7 Validation Constraints]

Sample Constraints:

- **@NotNull**
- **@Min**(*integer minimum*); **@Max**(*integer maximum*);
- **@Past**; **@Future**
- **@Size**(min = ; max =);

Hibernate Validator

Bean Validation API Hibernate Validator adds some useful constraints to those built into EE 7. The full list can be found here Hibernate Custom Constraints

Some of the more interesting ones include:

- **@NotEmpty**
- **@NotBlank**
- **@Email**(regexp=, flags=)
- **@SafeHtml** (whitelistType=, additionalTags=)
- **@URL**(protocol=, host=, port=, regexp=, flags=)

Apply Constraints to Entity Fields

Simple Implementation

```
@Entity
public class Person {
        @NotBlank(message = "Please enter first name.")
        private String firstname;

        @NotBlank(message = "Please enter last name.")
        private String lastname;

        @Past( message = "Birthdate must be in the past.")
        private Date DateOfBirth;
}
```

Multiple Constraints

A single field may have multiple constraints:

```
public class Person {
        @NotBlank(message = "Please enter first name.")
        @Size(max = 16, message = "First name must be at most 16 characters.")
        private String firstname;

        @NotBlank(message = "Please enter last name.")
        @Size(min=2,max = 16, message = "Last name must be between 2 and 16 characters.")
        private String lastname;
}
```

Inheriting Constraints

Because constraints are inherited a teacher would have to have valid first and last names, date of birth, degree, and certification.

```
public class Teacher extends Person {
        @NotNull(message = “Each teacher must have a degree.”,
groups = TeacherChecks.class)
        private Degree degree;

        @AssertTrue(message = “Teacher must be certified.”, groups = TeacherChecks.class)
        private boolean certified;
}
public interface TeacherChecks {
}
```

Course

We’ll add another class “Course” to illustrate validating by group

```
public class Course{
    @NotBlank
    private String title;
    @Min(12)
    private int seatCount;
    @AssertTrue(message = "The classroom must be available", groups = CourseChecks.class)
    private boolean available;
    @Valid
    private Teacher teacher;
    //Constructors, getters and setters
}
```

Test a Course

```
public class GroupTest {
        private static Validator validator;
        @BeforeClass
        public static void setUp() {
        ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
        validator = factory.getValidator();
        }
        @Test
        public void establishCourse() {
        Course course = new Course( “Geometry”, 2 );
        Set<ConstraintViolation<Course>> constraintViolations = validator.validate( course );
        assertEquals( 1, constraintViolations.size() );
        course.setSeatCount(12);
        //retest
        constraintViolations = validator.validate( course );
        assertEquals( 0, constraintViolations.size() );
```

Test a Course (Continued)

```
        // but is the room available?
        constraintViolations = validator.validate( course, CourseChecks.class);
        assertEquals( 1, constraintViolations.size() );
        assertEquals("The classroom must be available",constraintViolations.iterator().next().
getMessage());

        // let's make the room available
        course.setAvailable( true );
        assertEquals( 0, validator.validate( course, CourseChecks.class ).size() );
```

Test a Course (Continued)

```
        // now let's add a teacher.
        Teacher john = new Teacher( "John”, “Doe”, “1978-08-16” );
        john.setDegree(new Degree(“PhD”));
        course.setTeacher( john );
        constraintViolations = validator.validate( course, TeacherChecks.class );
        assertEquals( 1, constraintViolations.size() );
        assertEquals( “Teacher must be certified.”, constraintViolations.iterator().next().getMessage() );

        // ok, John gets  certified
        john.setCertified(true);
        assertEquals( 0, validator.validate( course, TeacherChecks.class ).size() );
        // just checking that everything is in order now
        assertEquals( 0, validator.validate(course,  Default.class, SchoolClassChecks.class, TeacherChecks.class).
size());
        }
```

2. Custom Constraints / Validators

Write your own constraints tailored to your specific validation requirements.

- Define Constraint Annotation
- Implement Constraint Validator
- Apply Annotation

Apply Annotation to Fields

```
@Entity
public class Dataset{
……
@CheckDate()
private String acceptanceDate;

@NotBlank(message = “Deposit Date is required.”)
@CheckDate(“YYYY-MM-DD”)
private String depositDate;

@CheckDate(“YYYY”)
private String publicationDate;

@CheckDate([“YYYY”, “YYYY-MM”])
private String collectionDate;
……
}
```

Custom Validation Annotation

The first step to writing a custom constraint is to write the annotation as follows:

```
@Target( {FIELD})
@Retention(RUNTIME)
@Constraint(validatedBy = CheckDateValidator.class)
public @interface CheckDate {
          //required
          String message() default "Please enter a valid date for this field. “;
          Class<?>[] groups() default {};
          Class<? extends Payload>[] payload() default {};

          //optional
          String[] dateFormat();
}
```

Validator Class

```
public class CheckDateValidator implements ConstraintValidator<CheckDate, String> {
        String dateFormat[] = [“”];

        @Override
        public void initialize(CheckDate constraintAnnotation) {
                dateFormat = constraintAnnotation.dateFormat();
        }

        @Override
        public boolean isValid(String value, ConstraintValidatorContext context) {
                return validate(value); // shown in next slide
        }
```

Validator Class (Continued)

```
private boolean validate(String dateString) {
boolean valid = false;
if (dateString.length == 0){
          return true;
}
for (String format : dateFormat) {
          if (!valid && (format.isEmpty() || format.toUpperCase().equals("YYYY-MM-DD")  )) {
          valid = isValidDate(dateString, "yyyy-MM-dd");
          }
          if (!valid && (format.isEmpty() || format.toUpperCase().equals("YYYY-MM"))) {
          valid = isValidDate(dateString, "yyyy-MM");
          }
          if (!valid && (format.isEmpty() || format.toUpperCase().equals("YYYY"))) {
          valid = isValidDate(dateString, "yyyy");
          }
}
          return valid;
}
```

Validator Class (Continued)

```
private boolean isValidDate(String dateString, String pattern) {
        boolean valid=true;
        Date date;
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        sdf.setLenient(false);
        try {
                dateString = dateString.trim();
                date = sdf.parse(dateString);
                }catch (ParseException e) {
                        valid=false;
                }
                return valid;
}
}
```

3. Cross-field Validation

Bean Validation allows you to also set annotations at the Class level.

Most common Use Case:

Cross-field Validation - testing the value of one field based on the value of one or more other fields

3. Cross-field Validation (continued)

Use a custom Validator:

- Define Constraint Annotation
- Implement Constraint Validator
- Apply Annotation

Apply Annotation to Fields

```
@NotNullIfAnotherFieldHasValue(
        fieldName = "department",
        fieldValue = "Other",
        dependFieldName = "otherDepartment")
public class SampleBean {
    private String department;
    private String otherDepartment;

...
}
```

Custom Validation Annotation

```
/**Validates that field {@code dependFieldName} is not null if field {@code fieldName} has value {@code fieldValue}.*/
@Target({TYPE, ANNOTATION_TYPE})
@Retention(RUNTIME)
@Constraint(validatedBy = NotNullIfAnotherFieldHasValueValidator.class)
@Documented
public @interface NotNullIfAnotherFieldHasValue {
    String fieldName();
    String fieldValue();
    String dependFieldName();

    String message() default "{NotNullIfAnotherFieldHasValue.message}";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};

    @Target({TYPE, ANNOTATION_TYPE})
    @Retention(RUNTIME)
    @Documented
    @interface List {
        NotNullIfAnotherFieldHasValue[] value();
    }
}
```

Validator Class

```
public class NotNullIfAnotherFieldHasValueValidator implements ConstraintValidator<NotNullIfAnotherFieldHasValue,
Object> {

    private String fieldName;
    private String expectedFieldValue;
    private String dependFieldName;

    @Override
    public void initialize(final NotNullIfAnotherFieldHasValue annotation) {
        fieldName          = annotation.fieldName();
        expectedFieldValue = annotation.fieldValue();
        dependFieldName    = annotation.dependFieldName();
    }

...
```

Validator Class (continued)

```
@Override
public boolean isValid(final Object value, final ConstraintValidatorContext ctx) {

    if (value == null) {
        return true;
    }
    try {
        final String fieldValue       = BeanUtils.getProperty(value, fieldName);
        final String dependFieldValue = BeanUtils.getProperty(value, dependFieldName);

        if (expectedFieldValue.equals(fieldValue) && dependFieldValue == null) {
            ctx.disableDefaultConstraintViolation();
            ctx.buildConstraintViolationWithTemplate(ctx.getDefaultConstraintMessageTemplate())
                .addNode(dependFieldName)
                .addConstraintViolation();
                return false;
        }
    } … // catch clauses omitted
    return true;
}
```

Custom Validation Annotation

```java
/**Validates that field {@code dependFieldName} is not null if field {@code fieldName} has value {@code fieldValue}.*/
@Target({TYPE, ANNOTATION_TYPE})
@Retention(RUNTIME)
@Constraint(validatedBy = NotNullIfAnotherFieldHasValueValidator.class)
@Documented
public @interface NotNullIfAnotherFieldHasValue {
    String fieldName();
    String fieldValue();
    String dependFieldName();

    String message() default "{NotNullIfAnotherFieldHasValue.message}";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};

    @Target({TYPE, ANNOTATION_TYPE})
    @Retention(RUNTIME)
    @Documented
    @interface List {
        NotNullIfAnotherFieldHasValue[] value();
    }
}
```

Apply Annotation to Fields

```
@NotNullIfAnotherFieldHasValue.List({
   @NotNullIfAnotherFieldHasValue(
      fieldName = "department",
      fieldValue = "Other",
      dependFieldName = "otherDepartment"),
   @NotNullIfAnotherFieldHasValue(
      fieldName = "position",
      fieldValue = "Other",
      dependFieldName = "otherPosition")
})
public class SampleBean {
   private String department;
   private String otherDepartment;
   private String position;
   private String otherPosition;
...
}
```

4. Dynamic Data Driven Application

Software framework for publishing, citing and preserving research data (open source on github for others to install)

Provides incentives for researchers to share:

•Recognition & credit via data citations

•Control over data & branding

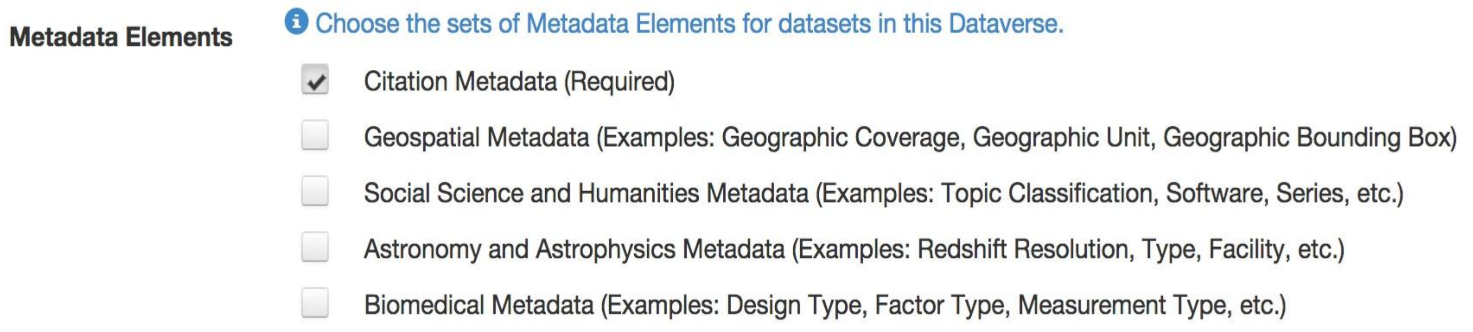
•Fulfill Data Management Plan requirements

The Dataverse Project

MetadataBlocks

Dataverse provides metadata support for any domain / research field:

- general metadata blocks available for all datasets
- domain specific metadata blocks



Metadata Elements

Choose the sets of Metadata Elements for datasets in this Dataverse.

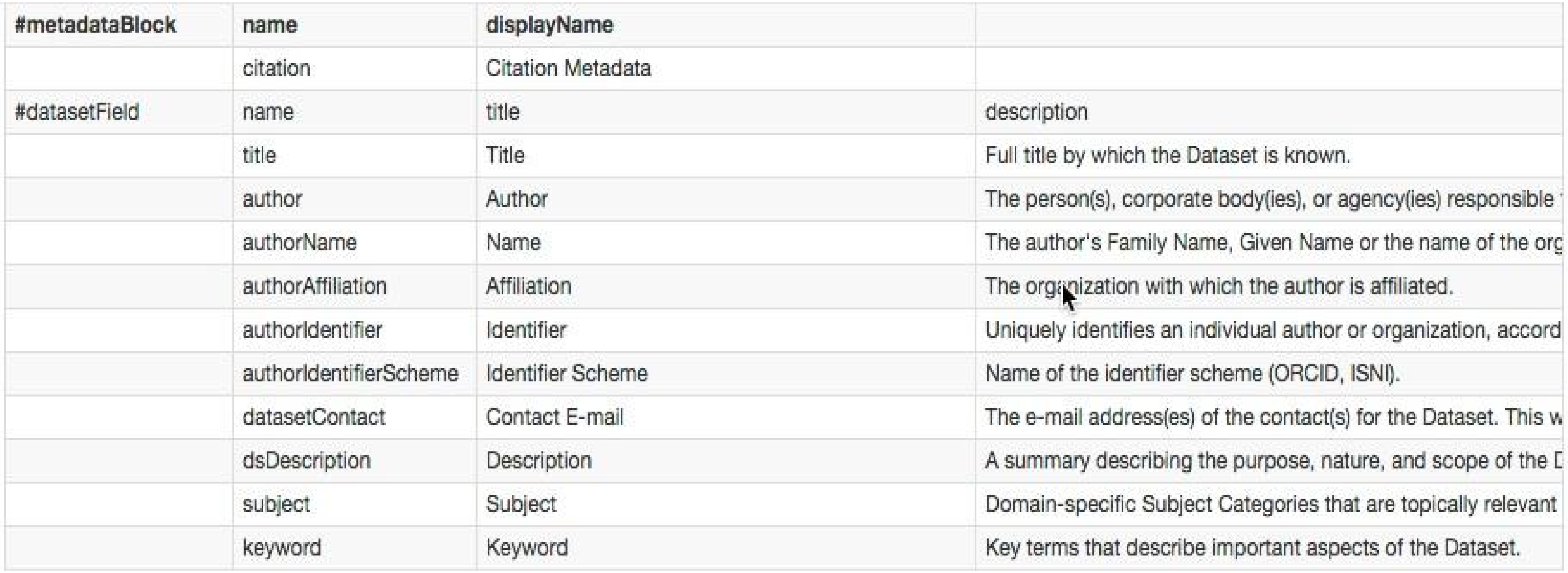
- [✓] Citation Metadata (Required)
- [ ] Geospatial Metadata (Examples: Geographic Coverage, Geographic Unit, Geographic Bounding Box)
- [ ] Social Science and Humanities Metadata (Examples: Topic Classification, Software, Series, etc.)
- [ ] Astronomy and Astrophysics Metadata (Examples: Redshift Resolution, Type, Facility, etc.)
- [ ] Biomedical Metadata (Examples: Design Type, Factor Type, Measurement Type, etc.)

DatasetFieldTypes

DatasetFieldTypes need to:

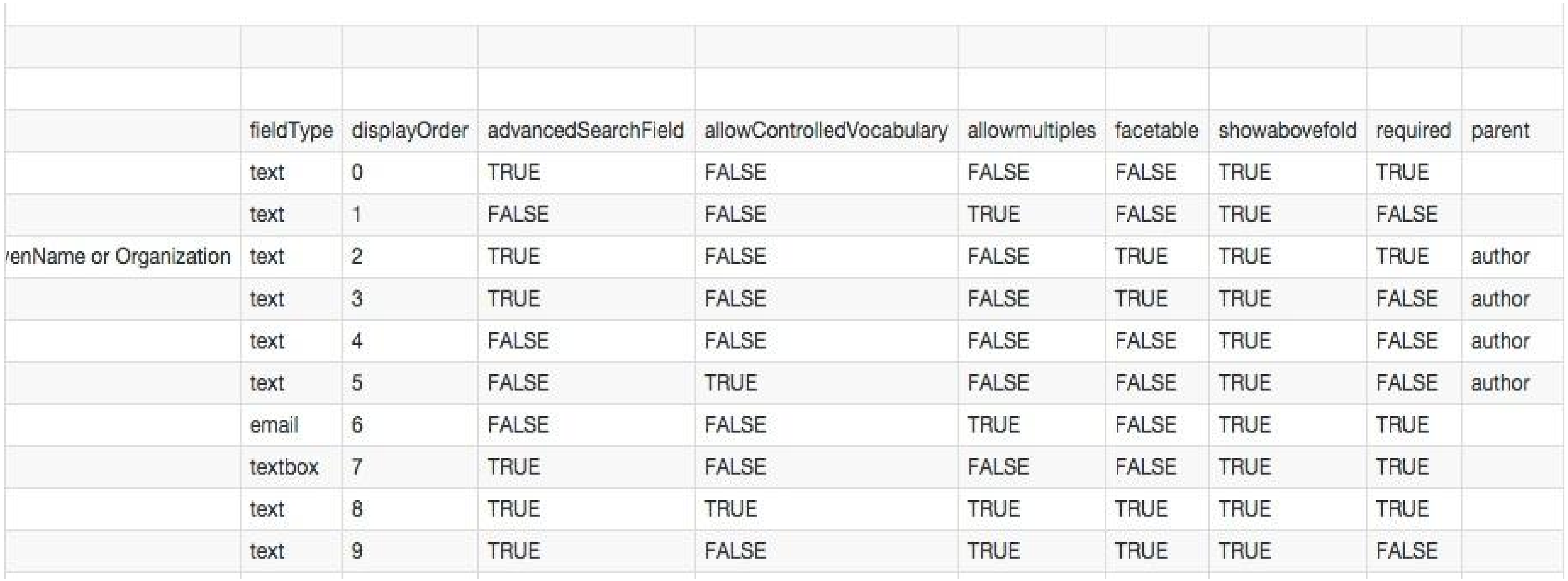
- Define different types of fields (date, float, e-mail, etc.)
- Define whether field is required (possibly different requirements per dataverse)
- Define whether a field can have one or multiple values
- Set a list of Controlled Vocabulary to be used with the field

TSV files



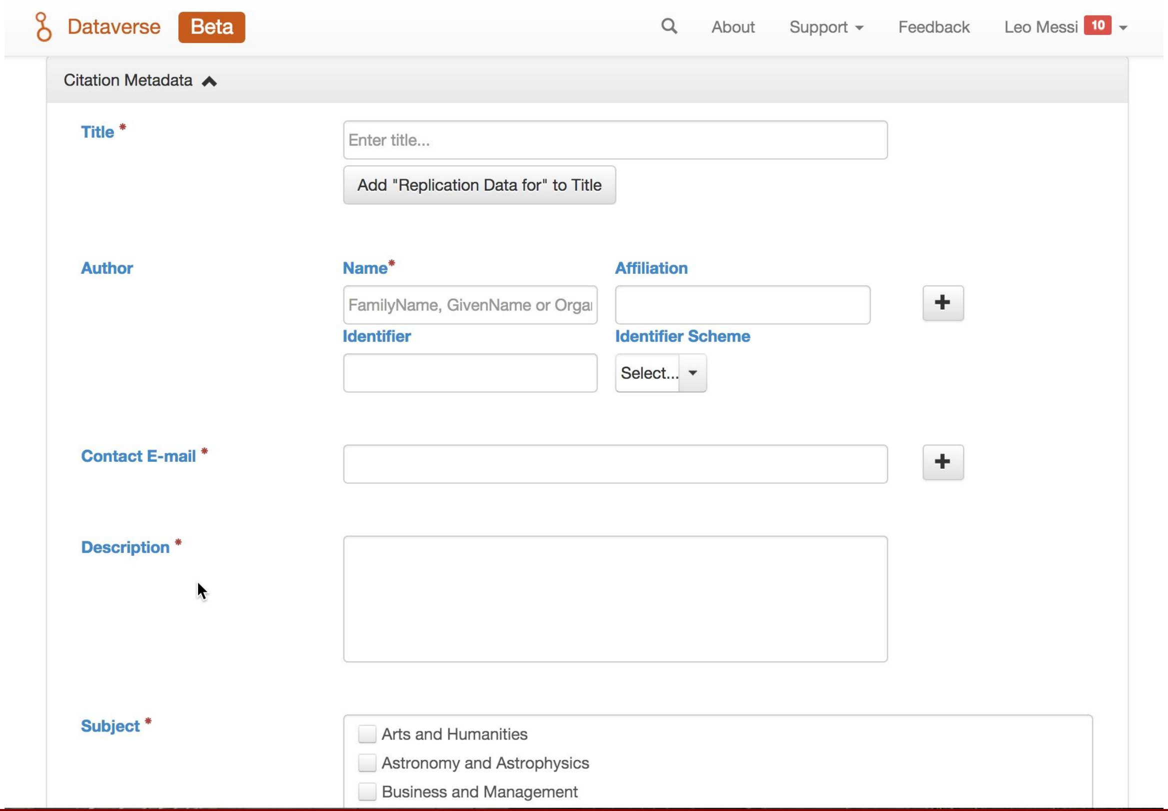
#metadataBlock	name	displayName	
	citation	Citation Metadata	
#datasetField	name	title	description
	title	Title	Full title by which the Dataset is known.
	author	Author	The person(s), corporate body(ies), or agency(ies) responsible
	authorName	Name	The author's Family Name, Given Name or the name of the org
	authorAffiliation	Affiliation	The organization with which the author is affiliated.
	authorIdentifier	Identifier	Uniquely identifies an individual author or organization, accord
	authorIdentifierScheme	Identifier Scheme	Name of the identifier scheme (ORCID, ISNI).
	datasetContact	Contact E-mail	The e-mail address(es) of the contact(s) for the Dataset. This w
	dsDescription	Description	A summary describing the purpose, nature, and scope of the
	subject	Subject	Domain-specific Subject Categories that are topically relevant
	keyword	Keyword	Key terms that describe important aspects of the Dataset.

TSV files



	fieldType	displayOrder	advancedSearchField	allowControlledVocabulary	allowmultiples	facetable	showabovefold	required	parent
	text	0	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	
	text	1	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	
enName or Organization	text	2	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	author
	text	3	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	author
	text	4	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	author
	text	5	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	author
	email	6	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	
	textbox	7	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	
	text	8	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	
	text	9	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	

Add Dataset



Title *

Enter title...

Add "Replication Data for" to Title

Author

Name*

FamilyName, GivenName or Orga

Affiliation

Identifier

Identifier Scheme

Select...

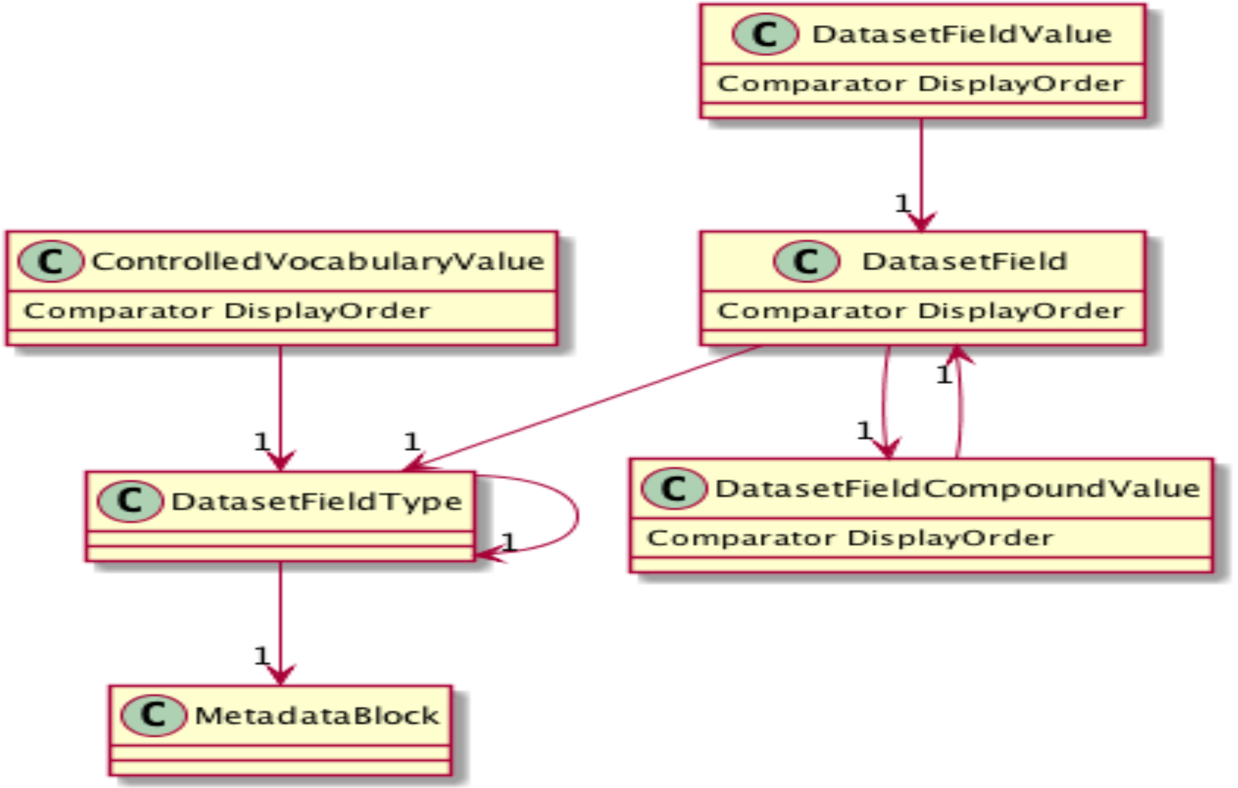
Contact E-mail *

Description *

Subject *

- Arts and Humanities
- Astronomy and Astrophysics
- Business and Management

DatasetField Object Model



Apply Annotation to Fields

```
@Entity
@ValidateDatasetFieldType
public class DatasetField implements Serializable {
```

```
@Entity
@ValidateDatasetFieldType
public class DatasetFieldValue implements Serializable {
```

Custom Validation Annotation

```
@Target({TYPE, ANNOTATION_TYPE})
@Retention(RUNTIME)
@Constraint(validatedBy = {DatasetFieldValidator.class, DatasetFieldValueValidator.class})
@Documented
public @interface ValidateDatasetFieldType {

  String message() default "Failed Validation for DSFType";

  Class<?>[] groups() default {};

  Class<? extends Payload>[] payload() default {};

}
```

Validator Class

```
public class DatasetFieldValueValidator implements ConstraintValidator<ValidateDatasetFieldType, DatasetFieldValue> {

    public void initialize(ValidateDatasetFieldType constraintAnnotation) {
    }

    public boolean isValid(DatasetFieldValue value, ConstraintValidatorContext context) {
        context.disableDefaultConstraintViolation(); // to have different messages depending on the different issue

        DatasetFieldType dsfType = value.getDatasetField().getDatasetFieldType();
        String fieldType = dsfType.getFieldType();

        if (fieldType.equals("float")) {
            try {
                Double.parseDouble(value.getValue());
            } catch (Exception e) {
                context.buildConstraintViolationWithTemplate(" " + dsfType.getDisplayName() + " is not a valid number.").
addConstraintViolation();
                return false;
            }
        }
```

Validator Class (continued)

```
public class DatasetFieldValidator implements ConstraintValidator<ValidateDatasetFieldType, DatasetField> {

    public void initialize(ValidateDatasetFieldType constraintAnnotation) {}

    public boolean isValid(DatasetField value, ConstraintValidatorContext context) {
        context.disableDefaultConstraintViolation(); // to have different messages depending on the different issue

        DatasetFieldType dsfType = value.getDatasetFieldType();
        //Additional logic turns off validation for templates
        if(dsfType.isPrimitive() && dsfType.isRequired() && value.getTemplate() == null && StringUtils.isBlank(value.getValue())) {
            if (value.getParentDatasetFieldCompoundValue() != null && value.getParentDatasetFieldCompoundValue().
getParentDatasetField().getTemplate() != null){
                return true;
            }
            context.buildConstraintViolationWithTemplate(dsfType.getDisplayName() + " is required.").addConstraintViolation();
            return false;
        }
        return true;
    }
}
```

Using the Validator

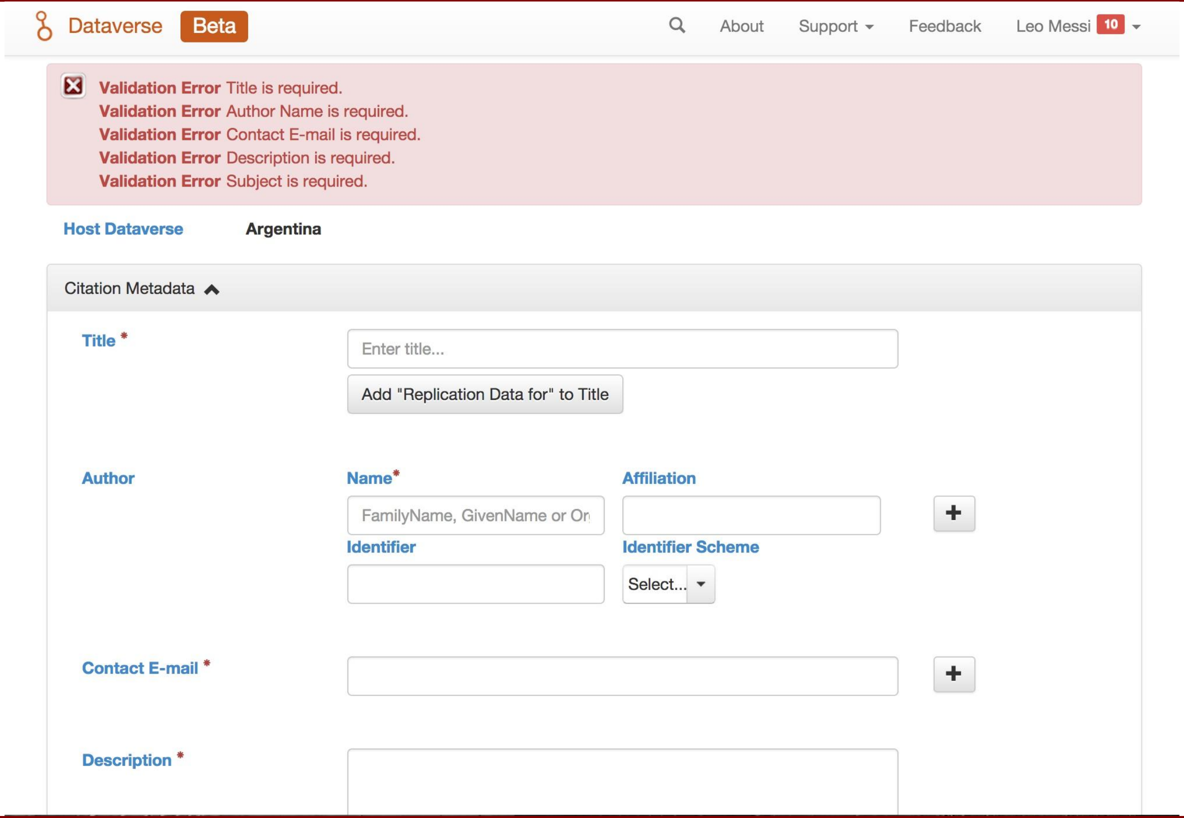
DatasetPage.java:

```
public String save() {
    // Validate
    boolean dontSave = false;
    ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
    Validator validator = factory.getValidator();

… // (see next slides)

    if (dontSave) {
        return "";
    }
        … // continue save process
```

Using the Validator (continued)

```
        for (DatasetField dsf : workingVersion.getFlatDatasetFields()) {
            Set<ConstraintViolation<DatasetField>> constraintViolations = validator.validate(dsf);
            for (ConstraintViolation<DatasetField> constraintViolation : constraintViolations) {
                FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Validation Error", constraintViolation.getMessage()));
                dontSave = true;
                break; // currently only support one message, so we can break out of the loop after the first constraint violation
            }
            for (DatasetFieldValue dsfv : dsf.getDatasetFieldValues()) {
                Set<ConstraintViolation<DatasetFieldValue>> constraintViolations2 = validator.validate(dsfv);
                for (ConstraintViolation<DatasetFieldValue> constraintViolation : constraintViolations2) {
                    FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Validation Error", constraintViolation.getMessage()));
                    dontSave = true;
                    break; // currently only support one message, so we can break out of the loop after the first constraint violation
                }
            }
        }
```

Using the Validator (continued)



Validation Error Title is required.
Validation Error Author Name is required.
Validation Error Contact E-mail is required.
Validation Error Description is required.
Validation Error Subject is required.

Host Dataverse Argentina

Citation Metadata

Title * Enter title...

Add "Replication Data for" to Title

Author Name* FamilyName, GivenName or Or Affiliation

Identifier Identifier Scheme Select...

Contact E-mail *

Description *

Using the Validator (continued)

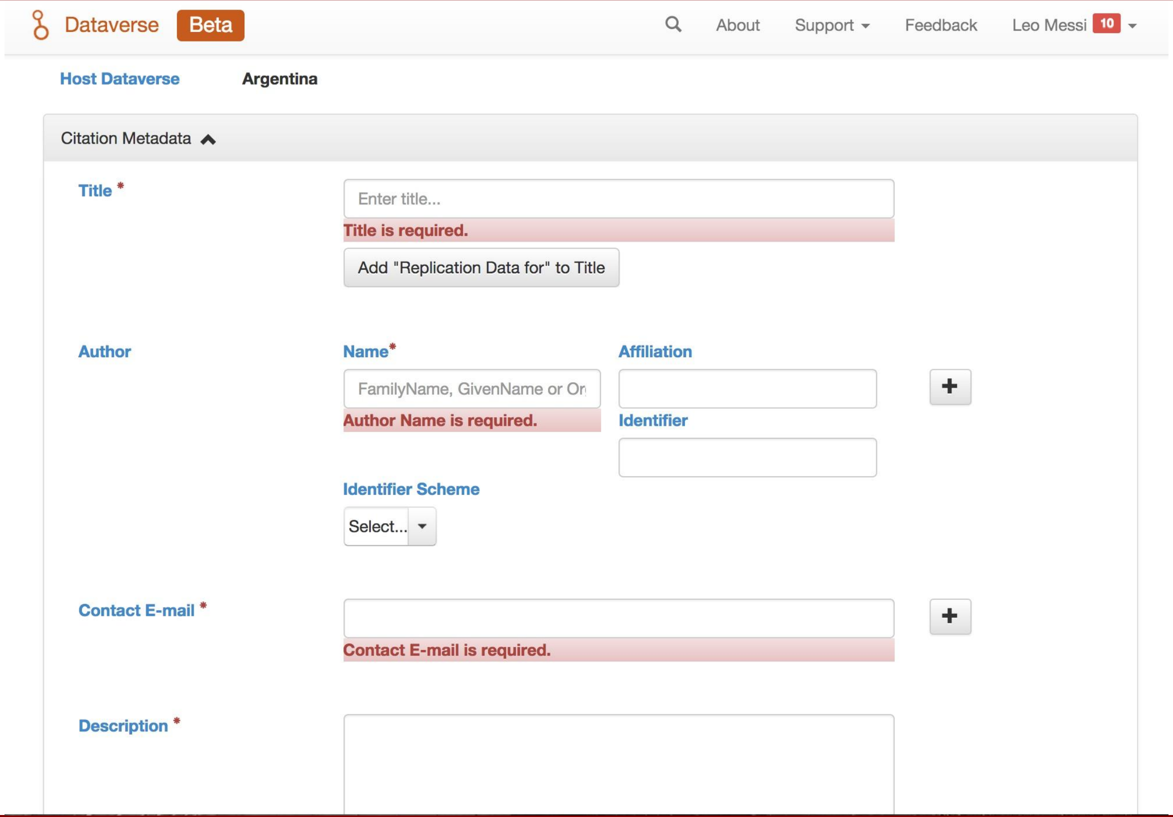
DatasetField.java and DatasetFieldValue.java:

```
 @Transient
  private String validationMessage;

  public String getValidationMessage() {
      return validationMessage;
  }

  public void setValidationMessage(String validationMessage) {
      this.validationMessage = validationMessage;
  }
```

Using the Validator (continued)

```
    for (DatasetField dsf : workingVersion.getFlatDatasetFields()) {
        dsf.setValidationMessage(null); // clear out any existing validation message
        Set<ConstraintViolation<DatasetField>> constraintViolations = validator.validate(dsf);
        for (ConstraintViolation<DatasetField> constraintViolation : constraintViolations) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Validation Error", constraintViolation.getMessage()));
            dsf.setValidationMessage(constraintViolation.getMessage());
            dontSave = true;
            break; // currently only support one message, so we can break out of the loop after the first constraint violation
        }
        for (DatasetFieldValue dsfv : dsf.getDatasetFieldValues()) {
            dsfv.setValidationMessage(null); // clear out any existing validation message
            Set<ConstraintViolation<DatasetFieldValue>> constraintViolations2 = validator.validate(dsfv);
            for (ConstraintViolation<DatasetFieldValue> constraintViolation : constraintViolations2) {
                FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Validation Error", constraintViolation.getMessage()));
                dsfv.setValidationMessage(constraintViolation.getMessage());
                dontSave = true;
                break; // currently only support one message, so we can break out of the loop after the first constraint violation
            }  }  }
```

Using the Validator (continued)



Host Dataverse Argentina

Citation Metadata

Title *

Enter title...

Title is required.

Add "Replication Data for" to Title

Author

Name*

FamilyName, GivenName or Or

Author Name is required.

Affiliation

Identifier

Identifier Scheme

Select...

Contact E-mail *

Contact E-mail is required.

Description *

Using the Validator with the API

```
try {
        createdDataset = engineSvc.submit(new CreateDatasetCommand(dataset, dataverseUser));

} catch (EJBException | CommandException ex) {
        Throwable cause = ex;
        StringBuilder sb = new StringBuilder();
        sb.append(ex.getLocalizedMessage());
        while (cause.getCause() != null) {
                cause = cause.getCause();
                if (cause instanceof ConstraintViolationException) {
                        ConstraintViolationException cve = (ConstraintViolationException) cause;
                        for (ConstraintViolation<?> violation : cve.getConstraintViolations()) {
                                sb.append(" Invalid value: '").append(violation.getInvalidValue()).append("' for ")
                              .append(violation.getPropertyPath()).append(" at ")
                              .append(violation.getLeafBean()).append(" - ")
                              .append(violation.getMessage());
                        }
                }
        }
        throw new SwordError(UriRegistry.ERROR_BAD_REQUEST, "Couldn't create dataset: " + sb.toString());
}
```

Thanks!

Further Talks

9/30 8PM: Lean Beans (Are Made of This): Command Pattern Versus MVC [BOF5619]

9/30 9PM: When the PrimeFaces Bootstrap Theme Isn't Enough [BOF5475]

Contact Info

gdurand@iq.harvard.edu

skraffmiller@hmdc.harvard.edu

Visit Data Science (IQSS) at: datascience.iq.harvard.edu/

Open source code at github.com/IQSS/dataverse