A Profile Approach to Using UML Models for Rich Form Generation

Tomas Cerny
Czech Technical University
FEE
Prague, Czech Rep
cernyto3@fel.cvut.cz

Eunjee Song
Baylor University
CS
Waco, TX, USA
eunjee song@baylor.edu
Introduction

• Model-driven development (MDD)
  ◦ advantages vs. issues

• User Interface
  ◦ Manual vs. MDD
  ◦ Initial development vs. maintenance
Motivation example

Simple class diagram

```
Person
- name: String
- born: Date
- email: String
- link: String
- salary: Integer
- graduated: Date
- notes: String
- id: Integer
```

```
Car
- brand: String
- color: Integer
- id: Integer
```

1 owns 1
**What can we get**

- Name: No
- Born: 04/30/2010
- Email: bob
- Link: bob page
- Salary: 100
- Graduated: today
- Notes: 
- Id: 1

**What we want**

- Name: *No*
- Email: *bob*
- Salary: 100
- Link: http://bob page
- Born: *04/30/2010*
- Graduated: *today*
- Notes: 

* required fields
Manual development

• Adding validation
  ◦ We know just data types
• Field order
• Dates restrictions
• Texts
  ◦ Email
  ◦ HTTP link
  ◦ HTML
  ◦ Long vs. short text
• Hidden fields
Manual development

- Impact of a change in a lower layer?
- Maintenance

MDD

- Models does not provide sufficient information for the code generation
Conventional libraries

- **Object-relational mapping (ORM)**
  - JPA standard
    - Hibernate, Open JPA, Toplink
- **Validation**
  - Server vs. client
    - Hibernate validator (HBV)
- **Annotations for extending the information for a particular field**
  - Still not enough
- **View annotations**
  - FormBuilder (FB)
  - This is enough
Promotion to MDD

- How do we extend models?
  - Modeling element UML
  - Our own models
  - Stereotypes, tag values for UML
    - UML Profiles

- UML Profiles
  - Advantages?
MD-JPA

UML Profile for (ORM)

• **JPA standard**
  - Towards a UML profile for model-driven object-relational mapping, Alexandre Torres, Renata Galante and Marcelo S. Pimenta, Instituto de Informatica Universidade Federal do Rio Grande do Sul, Brazil, 2009
  - XXIII Brazilian Symposium on Software Engineering
MD-HBV

UML Profile for (HBV)

- Successfully used in production
MD-FB

UML Profile for (FB)
UML Profiles

- JPA
  - Attribute
  - Class
  - Column
  - Embedded
  - Entity
  - Enumerated
  - Id
  - Lob
  - OrderBy
  - Persistent
  - Temporal
  - TemporalType
  - Transient
  - Version

- HibernateValidator
  - AssertFalse
  - AssertTrue
  - Attribute
  - CreditCardNumber
  - Digits
  - EAN
  - Email
  - Future
  - Length
  - Max
  - Min
  - NotEmpty
  - NotNull
  - Operation
  - Past
  - Pattern
  - Patterns
  - Range
  - Size

- FormBuilder
  - Attribute
  - Color
  - CompareTo
  - Comparison
  - FormOrder
  - Html
  - Ignore
  - InputLength
  - JSPattern
  - Link
  - Password
  - TableColumn
  - TextArea
  - Type
Back to our example
Back to our example

```java
@Entity
@Table(name = "Person", catalog = "FB")
public class Person implements java.io.Serializable {
    private String name;
    private Date born;
    ...
    @Column(name = "name", nullable = false, length = 100)
    @NotEmpty
    @Length(max = 100)
    @FormOrder(1)
    public String getName() {return this.name;}
    public void setName(String name) {
        this.name = name;
    }
    @Column(name = "born", nullable = false)
    @Temporal(TemporalType.DATE)
    @NotEmpty
    @Past
    @FormOrder(5)
    public String getBorn() {return this.born;}
    ...

    <h:form id="formPerson">
        <util:inputText label="Name"
                        edit="#{edit}"
                        value="#{bean.name}"
                        required="true"
                        size="30"
                        minlength="0"
                        maxlength="100"
                        title="#{text[t.person.name]}"
                        rendered="#{empty nameRender ? 'true' : nameRender}"
                        id="#{prefix}name"/>
        ...
        <util:inputDate label="Born"
                         edit="#{edit}"
                         value="#{bean.born}"
                         required="true"
                         title="#{text[t.person.born]}"
                         rendered="#{empty bornRender ? 'true' : bornRender}"
                         id="#{prefix}born"/>
        ...
    </h:form>
```
Advantages of our approach I.

- No manual intervention for code generation
  - Tedious and error-prone
- Maintenance
- User input validation integration
  - Client-side validation
- Usability support (HCI)
  - Knowledge of field meaning
  - Uniform strategy
- Approaches
  - On demand
  - Static pre-generation
Advantages of our approach II.

- Client HTTP, WML, standalone
- Simple integration of role-based security
- UML profiles compatible with UML tools
- Support developers to use MDD
Conclusion

• New approach for rich form generation
• UML Profiles for
  ○ ORM
  ○ Validation
  ○ View form generation
• Tool capable of rich form generation
• Advantages
  ○ Maintenance
  ○ Usability
  ○ Validation