FrameWeb

Vítor E. Silva Souza
(vitorsouza@inf.ufes.br)
http://www.inf.ufes.br/~vitorsouza

Department of Informatics
Federal University of Espírito Santo (Ufes),
Vitória, ES – Brazil
License for use and distribution

- This material is licensed under the Creative Commons license Attribution-ShareAlike 4.0 International;
- You are free to (for any purpose, even commercially):
  - Share: copy and redistribute the material in any medium or format;
  - Adapt: remix, transform, and build upon the material;
- Under the following terms:
  - Attribution: you must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use;
  - ShareAlike: if you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

More information can be found at:
http://creativecommons.org/licenses/by-sa/4.0/
FrameWeb

- Framework-based Design Method for Web Engineering:
  - Architectural design;
  - Web-based Information Systems (WISs);
  - Framework or container-based.
- Masters dissertation (2007 – Souza, Falbo & Guizzardi);
- Evolutions currently being proposed.
Motivation

• Use of frameworks:
  – Promote robust architectures;
  – Increase productivity (if WIS is not trivial);
  – State-of-practice (especially for the Web);
  – Drive the definition of standards;

• Bring frameworks to architectural models:
  – Lack of proposals;
  – More power (and responsibility) to software architects;
  – Code generation (MDD).
FrameWeb in a nutshell

• Definition of a basic architecture;
  – Separation of concerns in layers;
  – Use of frameworks;

• A UML profile for construction of architectural models:
  – Domain model;
  – Persistence model;
  – Application model
  – Navigation model.
Categories of frameworks

- Front Controller
- Dependency Injection
- Object/Relational Mapping
- Decorator
- Aspect-oriented Programming
- Authentication & Authorization
- etc...
Front Controller framework

1) Read Configuration

2) Instantiates and executes an action

3) Delegates the result to a view technology

Delegates request to the Front Controller

Request
Response

Web Server

Front Controller

Action

Business Logic

Web Pages
Object/Relational Mapping framework

![Diagram showing object/relational mappings]

- **ClassA**
  - `a1 : String`
  - `a2 : Integer`

- **ClassB**
  - `b1 : Date`

- **ClassC**
  - `c1 : String`
  - `c2 : Double`
  - `c3 : int`

**Domain classes**

**Object/Relational Framework**

**O/R Mappings**

**Relational DB**

- `A`
- `B`
- `C`
Dependency Injection framework

1) Request instance of SomeClass

2) Create instance and all dependencies

3) Return instance with dependencies satisfied

DI Configuration

Dependency Interface

DependencyClass

SomeClass

Client

Dependency Injection Framework
Decorator framework

Source: http://wiki.sitemesh.org/wiki/display/sitemesh/Home
Aspect-Oriented Programming framework

Same piece of code repeated in various methods

Aspects are separated and implemented only once.

An interceptor class executes the aspects in the proper moments.
Current focus of FrameWeb

Front Controller
Dependency Injection
Object/Relational Mapping

Optional
Decorator
Proposed architecture

- Front Controller
- View
- Control
- Business Tier
  - Domain
  - Application
- Data Access Tier
  - Persistence
- O/R Mapping
- Dependency Injection
- Decorator
Original frameworks (2007)

Struts²

sitemesh

spring

HIBERNATE

May 2015 FrameWeb
Running example – LabES Portal

Functionality
- name
- description

UserType
- enabledTypes
- description

User
- name
- birthDate
- gender
- institution
- profession
- address
- contactPhone
- email
- login
- password
- instructionLevel

Area
- name
- description
- superArea
- areasOfInterest

User
- type

Diagram:
- User
- Administrator
- LabES Member
- Professor

- Search Items
- Retrieve Items
- Authenticate User
- Manage User
- Manage Functionality
- Manage Area
- Manage Publication
- Manage Material
- Manage Project

«include>>
Running example – LabES Portal
Domain Model

- UML class diagram;
- O/R mappings added to classes via UML extensions;
- Use of sensible defaults;
- Guides the implementation of:
  - Classes of the Domain package;
  - O/R mapping for these classes.
<table>
<thead>
<tr>
<th>O/R Mapping</th>
<th>Extension</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the class is persistent, transient or mapped (not persistent itself, but its properties are persistent if another class inherits them)</td>
<td>Class stereotype</td>
<td>&lt;&lt;persistent&gt;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt;transient&gt;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt;mapped&gt;&gt;</td>
</tr>
<tr>
<td>Name of the table in which objects of a class will be persisted</td>
<td>Class constraint</td>
<td>table=name (class name)</td>
</tr>
<tr>
<td>If an attribute is persistent or transient</td>
<td>Attribute stereotype</td>
<td>&lt;&lt;persistent&gt;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt;transient&gt;&gt;</td>
</tr>
<tr>
<td>If an attribute can be null when the object is persisted</td>
<td>Attribute constraint</td>
<td>null</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not null</td>
</tr>
<tr>
<td>Date/time precision: store only the date, only the time or both (timestamp)</td>
<td>Attribute constraint</td>
<td>precision = (date</td>
</tr>
<tr>
<td>If the attribute is the primary-key of the table</td>
<td>Attribute stereotype</td>
<td>&lt;&lt;id&gt;&gt;</td>
</tr>
</tbody>
</table>
# Domain Model – UML Extensions

<table>
<thead>
<tr>
<th>O/R Mapping</th>
<th>Extension</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>How the ID attribute should be generated: automatically, obtained in a table, use of IDENTITY column, use of SEQUENCE column or none</td>
<td>Attribute constraint</td>
<td>generation = ( auto</td>
</tr>
<tr>
<td>If the attribute represents the versioning column.</td>
<td>Attribute stereotype</td>
<td>&lt;&lt;version&gt;&gt;</td>
</tr>
<tr>
<td>If an attribute should be stored in a large object field (e.g.: CLOB, BLOB)</td>
<td>Attribute stereotype</td>
<td>&lt;&lt;lob&gt;&gt;</td>
</tr>
<tr>
<td>Name of the column in which an attribute will be persisted</td>
<td>Attribute constraint</td>
<td>column=name (attribute name)</td>
</tr>
<tr>
<td>Size of the column in which an attribute will be persisted</td>
<td>Attribute constraint</td>
<td>size=value</td>
</tr>
<tr>
<td>If the association should be embedded (instead of having its own table, the associated child class' attributes are placed in the parent's table)</td>
<td>Attribute stereotype</td>
<td>&lt;&lt;embedded&gt;&gt;</td>
</tr>
</tbody>
</table>
# Domain Model – UML Extensions

<table>
<thead>
<tr>
<th>O/R Mapping</th>
<th>Extension</th>
<th>Possible Values</th>
</tr>
</thead>
</table>
| Inheritance mapping strategy: one table for each class using UNION, one table for each class using JOIN or single table for the entire hierarchy | Inheritance stereotype | `<union>`
|                                                                            |                        | `<join>`
|                                                                            |                        | `<single-table>` |
| Type of collection which implements the association: bag, list, set or map  | Association constraint | `collection = ( bag | list | set | map )` |
| Order of an association's collection: natural ordering (implemented in code) or order by columns (ascending or descending) | Association constraint | `order = ( natural | column names [asc | desc] )` |
| Cascading of operations through the association: nothing, persists, merges, deletions, refreshs or all | Association constraint | `cascade = ( none | persist | merge | remove | refresh | all )` |
| Association fetching strategy: lazy or eager.                              | Association constraint | `fetch = ( lazy | eager )` |
Persistence Model

- UML class diagram;
- Based on the DAO pattern;
  - Use of base DAO (e.g., nemo-utils) recommended;
  - DAOs show domain-specific operations (usually queries);
- No UML extensions needed;
- Guides the implementation of:
  - Classes and interfaces of the Persistence package.
Persistence Model – Base DAOs

**BaseDAO<IL, T>**

- `retrieveAll()` : Collection<T>
- `retrieveById(id : I) : T`
- `save(object : T) : void`
- `delete(object : T) : void`
- `getDomainClass() : Class`

**DomainObject**

**PersistentObject<IL, V>**

- `<<mapped>> DomainObjectSupport`
  - `uuid : String {not null}`
  - `equals(o : Object) : boolean`
  - `hashCode() : int`

- `<<mapped>> HibernatePersistentObject<IL, V>`
  - `<<id>> id : I`
  - `<<version>> version : V`
Persistence Model – LabES Portal

UserDAO

HibernateUserDAO

+ retrieveUserByLogin(login : String) : User
+ retrieveUsersByArea(area : Area) : List

HibernateUserTypeDAO

UserTypeDAO

HibernateAreaDAO

AreaDAO

HibernateFunctionalityDAO

+ retrieveFunctionalityByName(name : String) : Functionality

FunctionalityDAO
Navigation Model

- UML class diagram;
- Component types and controller configuration added to classes via UML extensions;
- Use of sensible defaults;
- Guides the implementation of:
  - Classes of the Control package;
  - Web pages of the View package;
  - Front Controller framework configuration.
Navigation Model – UML Extensions

- The type of each component is specified using UML stereotypes:

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Component type</th>
</tr>
</thead>
<tbody>
<tr>
<td>(none)</td>
<td>An action class, to which the Front Controller framework delegates the execution of the action.</td>
</tr>
<tr>
<td>&lt;&lt;page&gt;&gt;</td>
<td>A static or dynamic Web page.</td>
</tr>
<tr>
<td>&lt;&lt;template&gt;&gt;</td>
<td>A template that is processed by a template engine and is transformed into a Web page.</td>
</tr>
<tr>
<td>&lt;&lt;form&gt;&gt;</td>
<td>An HTML form.</td>
</tr>
<tr>
<td>&lt;&lt;binary&gt;&gt;</td>
<td>Any binary file that can be retrieved and displayed by the browser (e.g.: images, reports, documents, etc.).</td>
</tr>
</tbody>
</table>
The meaning of dependency associations depend on their source and destination:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page / template</td>
<td>Action class</td>
<td>A link in the page/template that triggers the execution of the action.</td>
</tr>
<tr>
<td>Form</td>
<td>Action class</td>
<td>Form data are sent to the action class when the form is submitted.</td>
</tr>
<tr>
<td>Action class</td>
<td>Page / template</td>
<td>The page/template is shown as one of the results of the action class.</td>
</tr>
<tr>
<td>Action class</td>
<td>Binary file</td>
<td>A binary file is shown as one of the results of the action class.</td>
</tr>
<tr>
<td>Action class</td>
<td>Action class</td>
<td>An action class is executed as result of another. This process is known as “action chaining”.</td>
</tr>
</tbody>
</table>
Navigation Model – UML Extensions

• Finally, UML constraints on dependencies configure the Front Controller framework:

  – Which method to call: method=\textit{name};
  – Which result to display: result=\textit{name};
  – Which result type to use: resultType=(\textit{binary} | \textit{chain} | \textit{dispatch} | \textit{redirect} | \textit{template}).
Navigation Model – LabES Portal
Application Model

- UML class diagram;
- No UML extensions needed;
- Guides the implementation of:
  - Classes and interfaces of the Application package;
  - Configuration of Dependency Injection framework.
Application Model – LabES Portal

```
usercontrol.controller::ManageUserService

ManageUserService

ManageUserServiceImpl

+ list(): SortedSet
+ create(object: Functionality): Functionality
+ retrieve(id: Long): Functionality
+ update(object: Functionality): Functionality
+ delete(object: Functionality): void

usercontrol.persistence::UserDAO

AuthenticateUserService

AuthenticateUserServiceImpl

+ login(login: String, password: String): User
+ logout(): void
+ remindPassword(login: String): User
```

May 2015
Some of the main limitations of the approach:

- The models may not be suited to other framework instances. What if I use JSF, CDI and JPA?
- The UML extensions do not prevent designers from including elements that do not belong in the model;
- There are no tools to help developers build models, verify models, generate code, etc.
Proposals:

– Adopt MDD/MDA concepts in the method;
  

– A base (common) meta-model for FrameWeb using the MDD/MDA concepts;

– An extensible set of packages suited to specific framework instances (e.g., Struts², JSF, VRaptor, etc);

– A design tool prototype.
Semantic FrameWeb

S-FRAMEWEB
S-FrameWeb

- FrameWeb extension to help integrate the resulting WIS into the Semantic Web;
- Proposed process:
Use of base ontology for educational portals:
• Use of ODM (Ontology Definition Metamodel, by OMG) for the S-FrameWeb Domain Model:

```
<<owlClass>> Functionality
+ name : string
+ description : string

<<objectProperty>> functionality

<<owlClass>> UserType
+ name : string
+ description : string

<<objectProperty>> type

<<owlClass>> Area
+ name : string
+ description : string

<<objectProperty>> subAreas

<<objectProperty>> areasOfInterest

<<owlClass>> User
+ name : string
+ birthDate : date
+ gender : char
+ institution : string
+ profession : string
+ address : string
+ contactPhone : string
+ email : string
+ login : string
+ password : string

XSD Datatypes
```
S-FrameWeb – LabES Portal

• Simplification of ODM:
S-FrameWeb: back to FrameWeb

- Domain Model based on S-FrameWeb Domain Model:
  - Specification of association navigabilities for the implementation of the classes;
  - Addition of the O/R mappings for the configuration of the ORM framework;
  - Use of the data types of the implementation platform instead of those defined by XSD;
- Persistence, Application and Navigation models as usual.
S-FrameWeb runtime components

- Developed for the Struts\textsuperscript{2} framework;
- Identifies requests coming from software agents and returns result in RDF/XML format:

![Diagram of S-FrameWeb runtime components]

Client (Web Browser)

\[ I_{\text{owl}} \quad I_1 \quad I_2 \quad \ldots \quad I_n \]

Action class

Pre-result Listeners

Result

May 2015
S-FrameWeb 2.0: work in progress

- Some of the main limitations of the approach:
  - Outdated Ontology Engineering process and modeling language;
  - Does not take into account many principles and best practices of Linked Data;
  - New technology (LD frameworks, triple stores, etc.) has been developed in the mean time;
• Proposals:
  – Use the SABiO approach for Ontology Engineering:
    • http://www.inf.ufes.br/~falbo/files/SABiO.pdf;
  – Use the OntoUML language and its extensions:
    • http://www.inf.ufes.br/~gguizzardi/OFSCM.pdf;
    • https://www.facebook.com/ontoumleditor.
  – Alternative 1: use triple stores, integrating JOINT:
    • http://www.sciencedirect.com/science/article/pii/S0957417413003382
  – Alternative 2: build a Semantic/LD layer on top of FrameWeb 2.0.
S-FrameWeb: ideas for the future

• Work on challenges of the Semantic Web:
  – Vocabulary mapping;
  – ID resolution;
  – Provenance tracking;
  – Data quality assessment;
  – Performance and scalability issues;
  – Semantic Web Services;
  – Tools, tools, tools!
  – Etc.

May 2015
http://nemo.inf.ufes.br/