Composing

Domain-Specific Languages

Eelco Visser
Delft University of Technology
http://eelcovisser.org

BLDL Opening 5/11/2009
Bergen University
Programming Languages

- Abstraction
- structured control-flow
- expressions
- objects
- procedures
- garbage collection
The Next Level of Abstraction?

Formal Methods
Domain-Specific Languages

not: general-purpose programming
not: general-purpose logic specifications

Specialization
Software System

Composition of Specialized Sub-domains
Software System

Separation of Concerns

Composition of Specialized Sub-domains
Loose Coupling

Lack of Static Verification
Separation of Concerns
requires
Linguistic Integration
World Wide Web
I'm designing a new language for web programming. It is called WebDSL.

Posted by Eelco on November 4, 2009 at 12:05

Edit

Save
Client

request

response

Server

Architecture
A flowchart showing the process from request to response:

1. Convert Request Parameters
2. Access Control
3. Validate Forms
4. Render Page Or Redirect
5. Handle Actions
6. Update Database
<table>
<thead>
<tr>
<th>Navigation</th>
<th>Access Control</th>
<th>Data Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forms</td>
<td>Data Model</td>
<td>Actions</td>
</tr>
<tr>
<td>Page Templates</td>
<td>Data Binding</td>
<td>Object Relational Mapping</td>
</tr>
</tbody>
</table>

(Some) Concerns in Web Applications
<table>
<thead>
<tr>
<th>data model</th>
<th>entity class</th>
<th>classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>orm</td>
<td>JPA</td>
<td>active record</td>
</tr>
<tr>
<td>actions</td>
<td>Java/Seam</td>
<td>Ruby</td>
</tr>
<tr>
<td>data validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>navigation</td>
<td>servlets</td>
<td>controllers</td>
</tr>
<tr>
<td>page templates</td>
<td>JavaServer Faces</td>
<td></td>
</tr>
<tr>
<td>forms</td>
<td>JSF tags</td>
<td>embedded html</td>
</tr>
<tr>
<td>data binding</td>
<td>JSF + Seam</td>
<td></td>
</tr>
<tr>
<td>access control</td>
<td>Acegi</td>
<td></td>
</tr>
</tbody>
</table>
<s:link view="/entry.xhtml" value="#{entry.title}" propagation="none">
    <f:param name="entry" value="#{entry.id}"/>
</s:link>

@Name("entry")
public class EntryBean implements EntryBeanInterface {
    @RequestParameter("entry")
    private Long entryId;
}
<table>
<thead>
<tr>
<th>Navigation</th>
<th>Access Control</th>
<th>Data Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forms</td>
<td>Data Model</td>
<td>Actions</td>
</tr>
<tr>
<td>Page Templates</td>
<td>Data Binding</td>
<td>Object Relational Mapping</td>
</tr>
</tbody>
</table>

- **data model**
  - orm
  - actions
  - data validation
  - navigation
  - page templates
  - forms
  - data binding
  - access control

- Boilerplate code
- Loose coupling
- No/little static verification
- separate languages for separate concerns

- linguistically integrated

- compile-time consistency checking

- 100% code generation (no scaffolding)
entity Blog {
    name :: String (id)
    entries -> Set<Entry>
    authors -> Set<User>
}

entity Entry {
    url :: String (id)
    title :: String (name)
    text :: WikiText
    blog -> Blog (inverse=Blog.entries)
    author -> User (inverse=Entry.author)
    created :: DateTime
}

entity User {
    username :: String
    password :: Secret
    blogs -> Set<Blog> (inverse=Blog.authors)
    entries -> Set<Entry> (inverse=Entry.author)
}
<table>
<thead>
<tr>
<th>Navigation</th>
<th>Access</th>
<th>Data</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forms</td>
<td>Data Model</td>
<td>Actions</td>
<td>Object Relational Mapping</td>
</tr>
<tr>
<td>Page Templates</td>
<td>Data Binding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
define page entry(e : Entry) {
    header{output(e.title)}
    output(e.text)
    "Posted by " output(e.author)
    " on " output(e.created)
    navigate(editentry(e)){"Edit"}
}
```

http://myblog.com/my-new-weblanguage

My New Web Language

I'm designing a new language for web programming. It is called WebDSL

Posted by Eelco on November 4, 2009 at 12:05

Edit
define page editentry(e : Entry) {
    form{
        input(e.url)
        input(e.title)
        input(e.text)
        input(e.author)
        input(e.created)
        action("Save", action{
            return entry(e);
        })
    }
}
entity Entry {
   url validate(isUniqueEntry(this),
     "Entry with that URL already exists")
}

http://myblog.com/edit/my-new-weblanguage

Edit Blog Entry

my-new-weblanguage
My New Web Language
I'm designing a new language for web programming. It is called WebDSL

Eelco
11/4/2009
Save
access control rules

principal is User
   with credentials username, password

rule page entry(e : Entry) {
   !e.private
   || principal in e.blog.authors
}

rule page newentry(b : Blog) {
   principal in e.blog.authors
}

rule page editentry(e : Entry) {
   e.author == principal
}
- separate languages for separate concerns
- linguistically integrated
- compile-time consistency checking
- 100% code generation (no scaffolding)
- order of magnitude decrease in code size
Declarative Access Control for WebDSL: Combining Language Integration and Separation of Concerns


Abstract

In this paper, we present the extension of WebDSL, a domain-specific language for web application development, with abstractions for declarative definition of access control. The extension supports the definition of a wide range of access control policies concisely and transparently as a separate concern. In addition to regulating the access to pages and actions, access control rules are used to infer navigation options not accessible to the current user, preventing the presentation of inaccessible links. The extension is an illustration of a general approach to the design of domain-specific languages for different technical domains to support separation of concerns in application development, while preserving linguistic integration. This approach is realized by means of a transformational semantics that weaves separately defined aspects into an integrated implementation.

http://researchr.org
Domain-Specific Language Engineering
Component-Based Language Engineering

DM → PIL
UI → PIL
AC → PIL
DV → PIL

Stratego/XT

WebDSL → Core WebDSL → PIL
Python
Java
C#
- separate languages for separate concerns
- linguistically integrated
- compile-time consistency checking
- 100% code generation (no scaffolding)
- order of magnitude decrease in code size

http://webdsl.org

http://strategoxt.org

http://researchr.org