

Software Productivity

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba

Syntax definition for DSLs

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba

What is a DSL?

```
application todo
entity User {
  username :: String (id,name)
  tasks    -> List<Task>
  log      -> List<Task>
}
entity Task {
  description :: String
  done        :: Bool
}
```

```
define page home() {
  section{
    header{"Users"}
    list{
      for(user : User) {
        listitem{
          navigate(tasks(user))
          {output(user.username)}
          " (" output(user.tasks.length) ")"
        }
      }
    }
    navigate(newuser()) {"Add new user"}
  }
}
```

```
define page newuser() {
  var user : User := User {}
  form{
    group("Add new user") {
      derive editRows from user for ( username )
    }
    action("Add", add())
    action add() {
      user.save();
      return tasks(user);
    }
  }
}
```

Defining a DSL...

- Domain abstractions, concepts
- Syntax
- Semantics
- Pragmatics

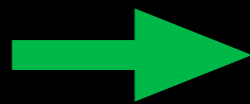
Semantics through transformations

```
application todo
entity User {
  username : String (id.name)
  tasks -> List<Task>
  log -> List<Task>
}
entity Task {
  description : String
  done : Bool
}

define page home() {
  section()
  header("Users")
  list()
  for (user : User) {
    listitem()
    navigate(tasks(user))
    (output(user.username))
    (" output(user.tasks.length) ")
  }
  navigate(newuser()) ("Add new user")
}

define page newuser() {
  var user : User := User ()
  form("Add new user") {
    desire addNew from user for ( username )
  }
  action add() {
    user.save()
    return tasks(user)
  }
}

define page tasks(user : User) {
  var newTask : Task := Task (done := false)
  action addTask() {
    user.tasks.add(newTask)
    newTask.save()
  }
  action done(task : Task) {
    task.done := true
    task.save()
  }
  action archive(task : Task) {
    user.log.add(task)
    user.save()
  }
  section()
  header("Tasks for " output(user.username))
  table()
  for (task : Task in user.tasks) {
    row()
    output(task.done)
    output(task.description)
    form()
    if (task.done) {
      action "Done", done(task)
    } else {
      action "Archive", archive(task)
    }
  }
  form()
  input(newTask.description)
  action "Add Task", addTask()
}
navigate(home()) ("Home")
```



```
build.xml
1 <project name="todo" default="compile">
2
3
4 <property name="hibernate.c3p0.acquire_increment" value="1">
5 </property>
6 <property name="hibernate.c3p0.min_size" value="0">
7 </property>
8 <property name="hibernate.c3p0.max_size" value="0">
9 </property>
10 <property name="hibernate.c3p0.timeout" value="0">
</property>
<property name="hibernate.c3p0.max_statements" value="0">
</property>
<property name="hibernate.c3p0.idle_test_period" value="0">
</property>
<property name="hibernate.c3p0.preferred_test_connections" value="0">
</property>
<property name="hibernate.connection.driver_class" value="org.hi
</property>
</project>

hibernate.properties
hibernate.c3p0.acquire_increment=1
hibernate.c3p0.min_size=0
hibernate.c3p0.max_size=0
hibernate.c3p0.timeout=0
hibernate.c3p0.max_statements=0
hibernate.c3p0.idle_test_period=0
hibernate.c3p0.preferred_test_connections=0
hibernate.connection.driver_class=org.hi
hibernate.connection.url=jdbc:hs

User.java
package todo.domain;

import java.util.*;
import javax.persistence.*;
import todo.domain.*;
import java.io.Serializable;
import utils.*;
import java.io.PrintWriter;
import org.webdsl.lang.Environment;

@Entity
@Table(name = "_User", uniqueConstraint = {
})
public class User {
    public void validateDelete()
}

addtask1.java
package todo.actions;

import todo.domain.*;

public interface addtask1 {
    public void addTask(Task task);
}

HomePage.java
package todo.actions;

import java.io.PrintWriter;
import org.webdsl.lang.Environment;
import utils.TemplateCall;

public class HomePage {
    public void render(PrintWriter s, Environment env) {
        s.println("<!DOCTYPE HTML PUBLIC");
        s.println("<html>");
        s.println("<head>");
        s.println("<title>" + getPageTitle());
        s.println("<link href=\"" + "/todo" + "\"");
        s.println("<link href=\"" + "/todo" + "\"");
        s.println("<script src=\"" + "/todo" + "\"");
        s.println("<script src=\"" + "/todo" + "\"");
        s.println("<script type=\"text/javascript\"");
        s.println("</head>");
        s.println("<body>");
        renderIncomingSuccessMessages(s);
        s.flush();
    }
}

EntityType.java
package utils;

public class EntityType {
    public static Object loadEntity(org.hibernate.Session hibSession) {
        try {
            return hibSession.load(c,s);
        } catch (Exception e) {
            return null;
        }
    }
}
```

Syntax for entities...

```
entity Person {  
  fullname : String  
  email : String  
  homepage : String  
}
```

```
entity Publication {  
  title : String  
  author : Person  
  year : Int  
  abstract : String  
  pdf : String  
}
```

Defining the syntax, sentences...

context-free syntax

Definition* -> Model

Entity -> Definition

"entity" Id "{" Property* "}" -> Entity

Id ":" Sort -> Property

Id -> Sort

with constructors

context-free syntax

```
Definition* -> Model {cons("Model")}
```

```
Entity -> Definition
```

```
"entity" Id "{" Property* "}"
```

```
-> Entity {cons("Entity")}
```

```
Id ":" Sort -> Property {cons("Property")}
```

```
Id -> Sort {cons("SimpleSort")}
```

capital letter is
mandatory!

ASTs and signatures

```
module Entity
signature
  constructors
    SimpleSort : Id -> Sort
    Property   : Id * Sort -> Property
    Entity     : Id * List(Property) -> Entity
              : Entity -> Definition
    Model     : List(Definition) -> Model
              : String -> Id
```

Defining the syntax, words

lexical syntax

`[a-zA-Z][a-zA-Z0-9_]* -> Id`

`[0-9]+ -> Int`

`"\" ~ [\"\\n]* \"\" -> String`

`[\\ \\t\\n\\r] -> LAYOUT`

`"//\" ~ [\\n\\r]* [\\n\\r] -> LAYOUT`

Putting it all together...

```
module DataModel
  exports
    sorts Id Int Entity Property Sort...
  context-free syntax
    "entity" Id "{" Property* "}"
      -> Entity {cons("Entity")}
    Id ":" Sort -> Property {cons("Property")}
    ...
  lexical syntax
    [a-zA-Z][a-zA-Z0-9\_]* -> Id
    ...
```

Syntax definition language...

- Literals: "while"
- Zero or more : Stm^*
- One or more: TypeDec^+
- Zero or more, with separator: $\{\text{Exp } ", " \}^*$
- One or more, with separator: $\{\text{Id } ". " \}^+$
- Optional: $\text{Expr}?$
- Alternative: $\{\text{Expr } ", " \}^* \mid \text{LocalVarDec}$
- Sequence: $(A_0 \dots A_n)$

Lexical

- Digits: `[0-9]`
- Hexa: `[0-9a-fA-F]`
- Whitespace: `[\ \t\r\n]`
- Strings: `~ [\"\\n\r]`

Reject and follow restriction

lexical syntax

"goto" -> Id {reject}

lexical restrictions

Id -/- [A-Za-z0-9]

"goto" -/- [A-Za-z0-9]

context-free syntax

"goto" Id -> Stm {cons("Goto")}

Multi-line comments

lexical syntax

BlockComment -> LAYOUT

"/*" CommentPart* "*/" -> BlockComment

~[\/*] -> CommentPart

Asterisk -> CommentPart

Slash -> CommentPart

BlockComment -> CommentPart

[\/] -> Slash

[*] -> Asterisk

lexical restrictions

Asterisk -/- [\/]

Slash -/- [*]

Priorities

context-free syntax

```
Exp "*" Exp -> Exp {left, cons("Times")}
```

```
Exp "/" Exp -> Exp {left, cons("Div")}
```

```
Exp "+" Exp -> Exp {left, cons("Plus")}
```

```
Exp "-" Exp -> Exp {left, cons("Minus")}
```

```
Exp ">" Exp -> Exp {cons("Gt"), non-assoc}
```

context-free priorities

```
{left: Exp "*" Exp -> Exp
```

```
      Exp "/" Exp -> Exp
```

```
}
```

```
> {left: Exp "+" Exp -> Exp
```

```
      Exp "-" Exp -> Exp
```

```
}
```

```
> {non-assoc:
```

```
      Exp ">" Exp -> Exp
```

```
}
```


Start symbol and brackets

exports

context-free start-symbols Exp

context-free syntax

Id -> Exp {cons("Var")}

IntConst -> Exp {cons("Int")}

" (" Exp ")" -> Exp {bracket}

Where are
we now?

Intended learning outcome

- Develop web applications using the WebDSL
- Identify and justify necessary changes to a DSL
- Define, implement and change DSLs

Intended learning outcome

- Analyze and explain benefits and drawbacks of developing web applications with
 - general purpose languages and frameworks
 - DSL
- Develop a business case for a DSL

Syntax definition for DSLs

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba

Software Productivity

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba