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
  }
```

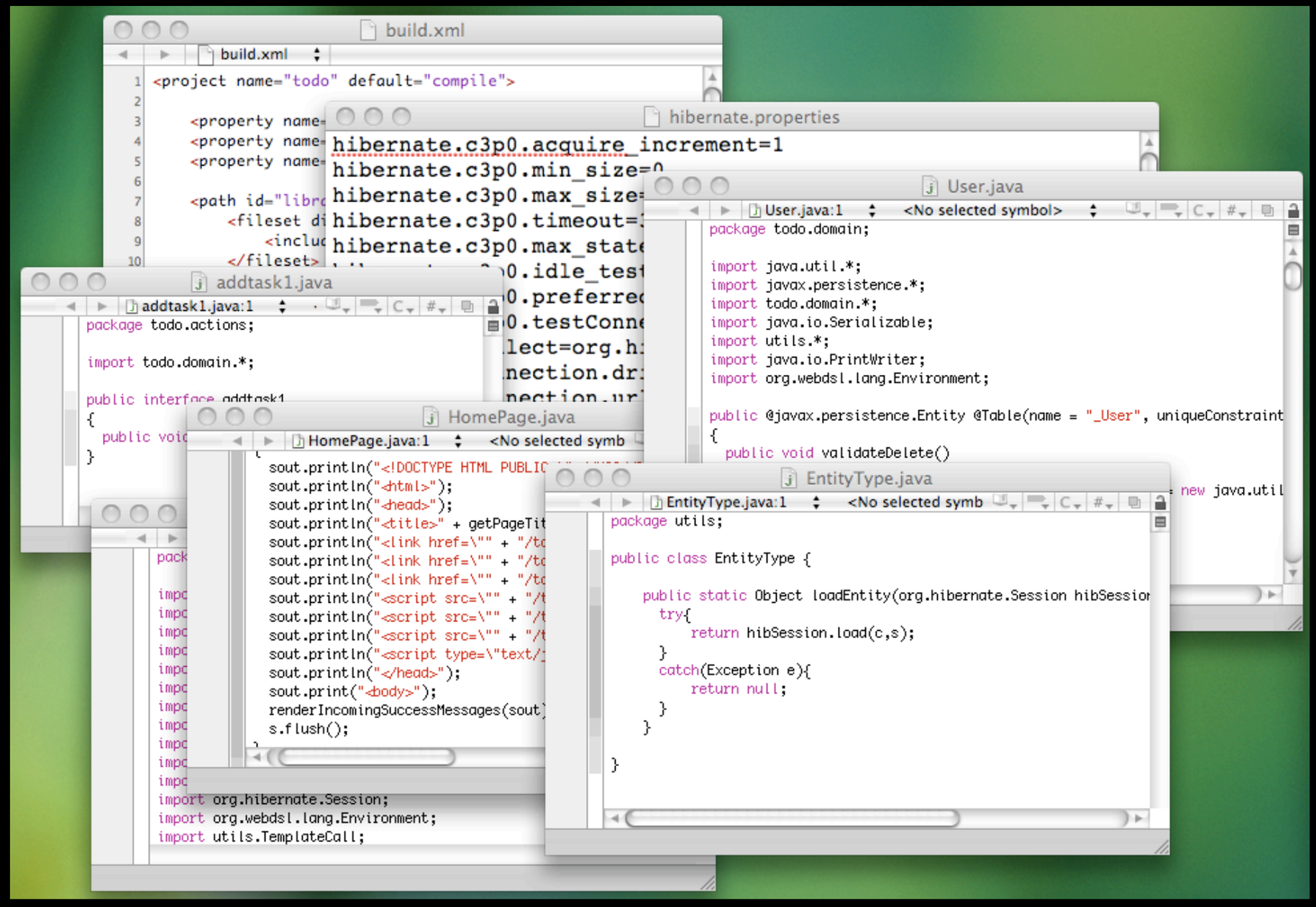
```
define page home() {
  section{
    header{"Users"}
    list{
      for(user : User) {
        listitem{
          navigate(tasks(user))
            {output(user.username)}
            " (" output(user.tasks.length) ")"
        }
      }
    }
    navigate(newuser()){"Add new user"}
  }
}
```

```
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



Syntax for entities...

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

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

Full example from Eelco Visser, *Domain-Specific Language Engineering*, GTTSE 2007.

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: `{Exp ","}*`
- One or more, with separator: `{Id "."}+`
- Optional: `Expr?`
- Alternative: `{Expr ","}* | LocalVarDec`
- Sequence: `(A0...An)`

Lexical

- Digits: `[0-9]`
- Hexa: `[0-9a-fA-F]`
- Whitespace: `[\ \t\12\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