
# Meta-Language Support for Type-Safe Access to External Resources

Mark Hills, Paul Klint, and Jurgen J. Vinju

5th International Conference on Software Language Engineering (SLE 2012)
September 28, 2012
Dresden, Germany

http://www.rascal-mpl.org

# Rascal: A Meta-Programming One-Stop-Shop

- Wide variety of meta-programming tasks -- grammar inference, OCL-based test case generation, model transformation, ontology design, etc

- This diversity is a challenge: wide variety of languages (including dialects), external systems, and data formats

- Typical solution: many different tools, glue code

- We want to pull all this into Rascal: the “one-stop-shop”

Picture from: http://www.mountainhighlands.com/listings/colabrese.html

# A Typical Scenario: Software Evolution

# A Typical Scenario: Software Evolution

- Given a software system *S*, written in some language *L*

# A Typical Scenario: Software Evolution

- Given a software system *S*, written in some language *L*
  - Extract files and history of *S* from source repository

# A Typical Scenario: Software Evolution

- Given a software system *S*, written in some language *L*
  - Extract files and history of *S* from source repository
  - Parse files in *S* using grammar for *L*

# A Typical Scenario: Software Evolution

- Given a software system *S*, written in some language *L*
  - Extract files and history of *S* from source repository
  - Parse files in *S* using grammar for *L*
  - Extract, manipulate, and save facts about *S*

# A Typical Scenario: Software Evolution

- Given a software system *S*, written in some language *L*
  - Extract files and history of *S* from source repository
  - Parse files in *S* using grammar for *L*
  - Extract, manipulate, and save facts about *S*
  - Crosscheck history of *S* with bug tracking system

# A Typical Scenario: Software Evolution

- Given a software system *S*, written in some language *L*
  - Extract files and history of *S* from Subversion
  - Parse files in *S* using SDF grammar for *L*
  - Extract, manipulate, and save facts about *S* in CSV files
  - Crosscheck history of *S* with Bugzilla using JDBC

# The Rascal Solution

- Given a software system *S*, written in some language *L*
  - Extract files and history of *S* from Subversion using Rascal Subversion library
  - Parse files in *S* using Rascal grammar for *L* created using the SDF to Rascal grammar importer library
  - Extract, manipulate, and save facts about *S* in CSV files using the Rascal CSV file parser/importer library
  - Crosscheck history of *S* with Bugzilla using the Rascal JDBC library

# Exploring the Rascal Solution: Zooming in on CSV

Workbook1

Sheets | Charts | SmartArt Graphics | WordArt

| | A | B | C | J | K | L | M | N | O | R | S | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | product | version | file | \echo | \expressior | \for | \foreach | \functionDef | \global | \if | \inlineHTML | \interfaceDef |
| 2 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 38 | 0 | 0 | 0 | 3 | 11 | 0 | 0 |
| 4 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 8 | 18 | 2 | 0 | 0 | 0 | 10 | 6 | 0 |
| 5 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 15 | 46 | 0 | 0 | 0 | 0 | 7 | 15 | 0 |
| 6 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 16 | 3 | 0 | 0 | 2 | 5 | 0 | 0 |
| 7 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 76 | 0 | 0 | 0 | 0 | 18 | 0 | 0 |
| 8 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 22 | 0 | 0 | 0 | 3 | 6 | 0 | 0 |
| 9 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 21 | 0 | 0 | 0 | 2 | 6 | 0 | 0 |
| 11 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 178 | 8 | 2 | 0 | 9 | 43 | 0 | 0 |
| 12 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 33 | 4 | 0 | 0 | 2 | 13 | 0 | 0 |
| 14 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 17 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| 15 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 61 | 2 | 2 | 0 | 8 | 19 | 0 | 0 |
| 16 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 72 | 1 | 0 | 0 | 0 | 22 | 0 | 0 |
| 17 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 8 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 18 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 152 | 12 | 0 | 13 | 3 | 42 | 0 | 0 |
| 19 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 9 | 0 | 0 | 0 | 1 | 6 | 0 | 0 |
| 20 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 13 | 26 | 0 | 0 | 0 | 0 | 6 | 14 | 0 |
| 25 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 25 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 117 | 1 | 0 | 0 | 8 | 45 | 0 | 0 |
| 27 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 20 | 23 | 2 | 0 | 0 | 0 | 4 | 16 | 0 |
| 29 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 5 | 6 | 0 | 0 | 0 | 0 | 0 | 6 | 0 |
| 30 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 0 | 17 | 2 | 0 | 0 | 1 | 6 | 0 | 0 |
| 31 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/( | 20 | 77 | 4 | 0 | 0 | 0 | 17 | 18 | 0 |

Sheet1

# How would I use this CSV file in Rascal?

# How would I use this CSV file in Rascal?

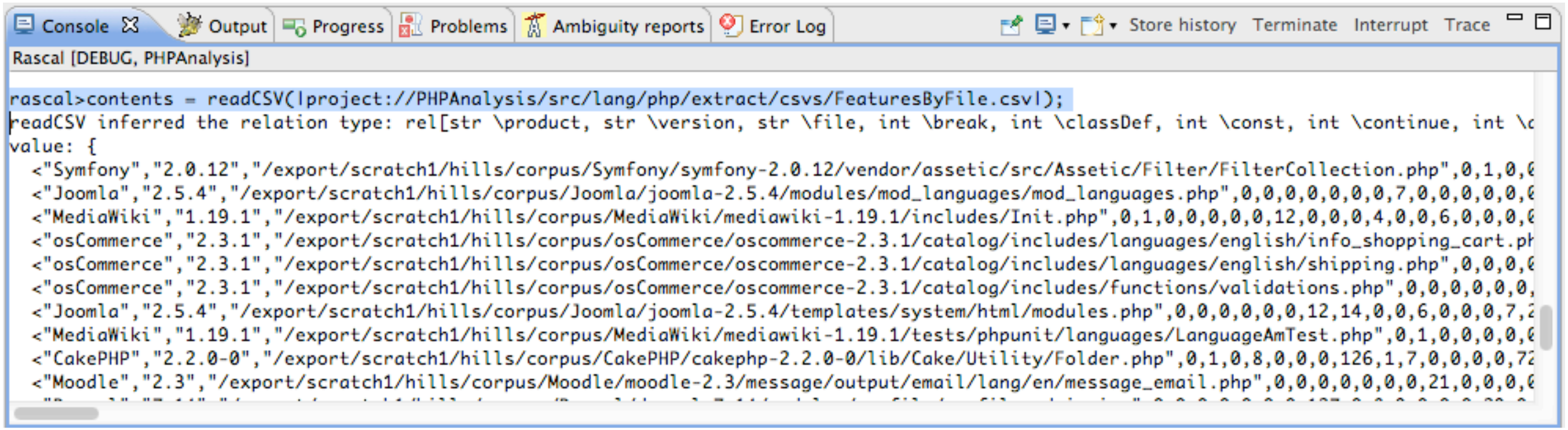
- Rascal has a parser and library for CSV files

# How would I use this CSV file in Rascal?

- Rascal has a parser and library for CSV files
- Easy to use, import the library, then just import the file!

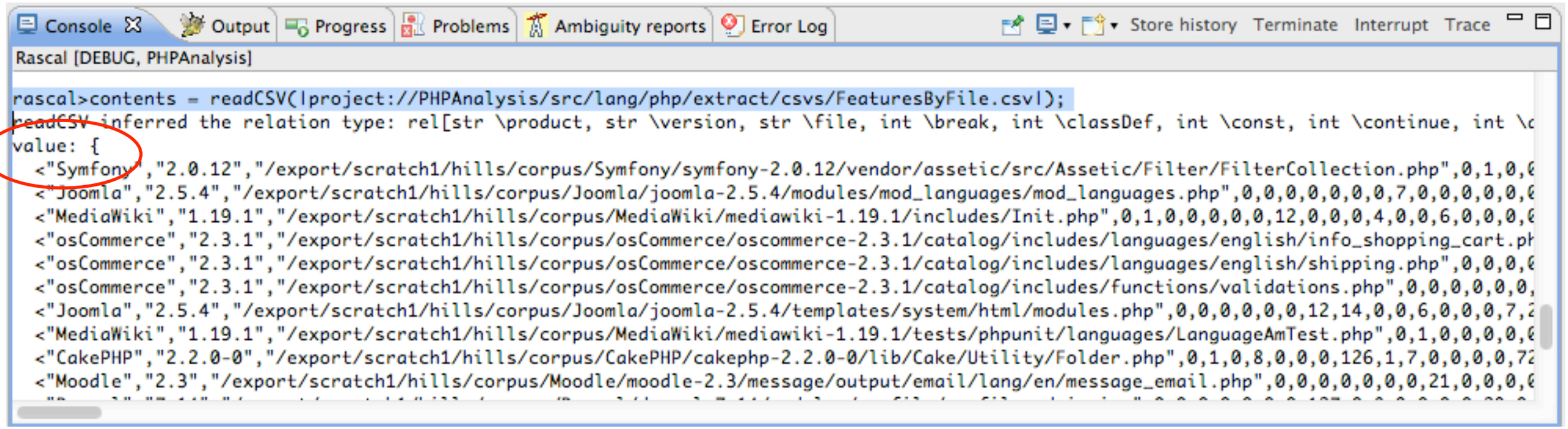
# How would I use this CSV file in Rascal?

- Rascal has a parser and library for CSV files
- Easy to use, import the library, then just import the file!



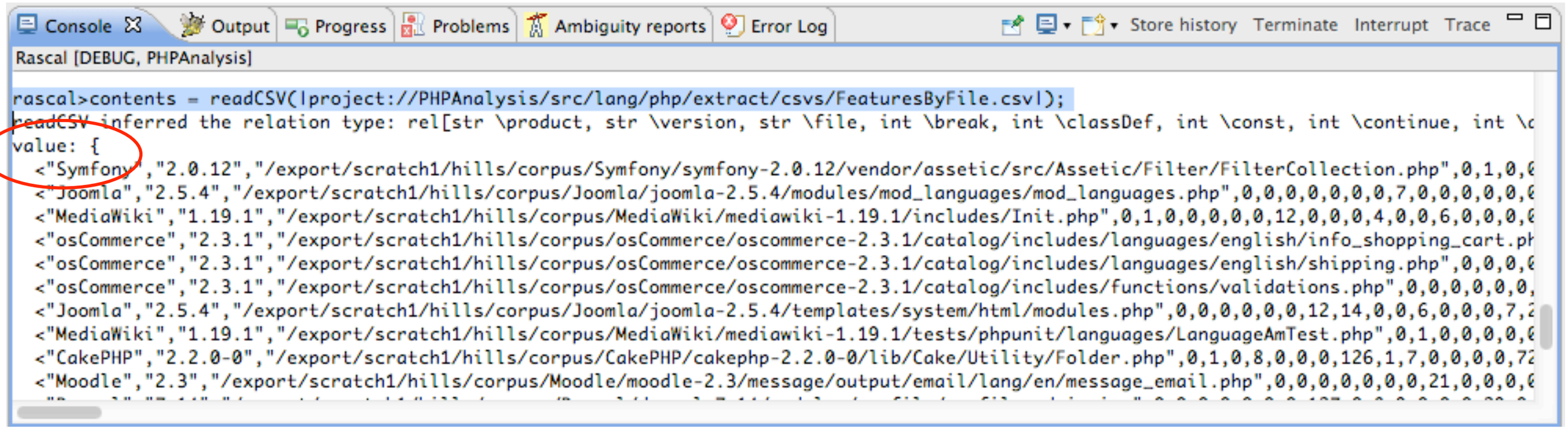
# How would I use this CSV file in Rascal?

- Rascal has a parser and library for CSV files
- Easy to use, import the library, then just import the file!



# How would I use this CSV file in Rascal?

- Rascal has a parser and library for CSV files
- Easy to use, import the library, then just import the file!




- But, the result is a value -- needs to be picked apart using pattern matching

# Why is that a problem?

- Does anyone want to manually write a 112-tuple?

Workbook1

Sheets | Charts | SmartArt Graphics | WordArt

| | CW | CX | CY | CZ | DA | DB | DC | DD | DE | DF | DG | DH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | \isSet | \print | \propertyFetch | \shellExec | \ternary | \fetchStaticProperty | \scalar | \var | \propertyDef | \classConstDef | \methodDef | \traitUse |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 18 | 0 | 1 | 0 | 68 | 50 | 5 | 0 | 6 | 0 |
| 4 | 2 | 0 | 0 | 0 | 0 | 0 | 45 | 56 | 0 | 0 | 0 | 0 |
| 5 | 7 | 0 | 16 | 0 | 1 | 0 | 187 | 121 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 12 | 0 | 0 | 0 | 44 | 65 | 0 | 0 | 6 | 0 |
| 7 | 14 | 0 | 2 | 0 | 3 | 0 | 129 | 74 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 19 | 0 | 1 | 0 | 56 | 41 | 1 | 0 | 14 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |

Sheet1

Normal View | Ready | Sum=0

- And what types would each tuple item be?

# Why not just look at the CSV file to get the types?

- Not practical for big files: 112 columns, 19860 rows

Workbook1

Sheets | Charts | SmartArt Graphics | WordArt

| | A | B | C | J | K | L | M | N | O | R | S | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 19831 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 18 | 0 | 0 | 0 | 0 | 8 | 0 | |
| 19832 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19833 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 6 | 0 | 0 | 0 | 0 | 4 | 0 | |
| 19834 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 43 | 0 | 1 | 0 | 0 | 9 | 0 | |
| 19835 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 6 | 0 | 1 | 0 | 0 | 1 | 0 | |
| 19836 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19837 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19838 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 19839 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 6 | 0 | 1 | 0 | 0 | 2 | 0 | |
| 19840 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19841 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 7 | 0 | 0 | 0 | 0 | 2 | 0 | |
| 19842 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19843 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19844 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19845 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 10 | 0 | 4 | 0 | 0 | 5 | 0 | |
| 19846 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19847 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 5 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 19848 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 12 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 19849 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 18 | 0 | 0 | 0 | 0 | 8 | 0 | |
| 19850 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 9 | 0 | 2 | 0 | 0 | 14 | 0 | |
| 19851 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19852 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 19853 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 19854 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 7 | 0 | 1 | 0 | 0 | 3 | 0 | |
| 19855 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 26 | 0 | 3 | 0 | 0 | 1 | 0 | |
| 19856 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 4 | 0 | 2 | 0 | 0 | 7 | 0 | |
| 19857 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | |
| 19858 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 14 | 0 | 0 | 0 | 0 | 6 | 0 | |
| 19859 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 10 | 0 | 0 | 0 | 0 | 2 | 0 | |
| 19860 | DoctrineORM | 2.2.2 | /export/scratch1/hills/corpus/l | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | |

Sheet1 +

Normal View | Ready | Sum=0

# Can I compute the type?

- Rascal functions can return *type literals*
- These literals can then be printed to get the Rascal type
- Then, printed literal used in typed IO functions
- Manual: Copying/pasting type strings

```
rascal>csvType = getCSVType(|project://PHPAnalysis/src/lang/php/extract/csvs/FeaturesByFile.csv|);
readCSV inferred the relation type: rel[str \product, str \version, str \file, int \break, int \classDef, int \const,
int \continue, int \declare, int \do, int \echo, int \expressionStatementChainRule, int \for, int \foreach, int
\functionDef, int \global, int \goto, int \haltCompiler, int \if, int \inlineHTML, int \interfaceDef, int \traitDef,
int \label, int \namespace, int \return, int \static, int \switch, int \throw, int \tryCatch, int \unset, int \use,
int \while, int \array, int \fetchArrayDim, int \fetchClassConst, int \assign, int \assignWithOperationBitwiseAnd,
int \assignWithOperationBitwiseOr, int \assignWithOperationBitwiseXor, int \assignWithOperationConcat, int
\assignWithOperationDiv, int \assignWithOperationMinus, int \assignWithOperationMod, int \assignWithOperationMul, int
\assignWithOperationPlus, int \assignWithOperationRightShift, int \assignWithOperationLeftShift, int \listAssign, int
\refAssign, int \binaryOperationBitwiseAnd, int \binaryOperationBitwiseOr, int \binaryOperationBitwiseXor, int
\binaryOperationConcat, int \binaryOperationDiv, ...
```

# Great! But, what if the type changes?

Workbook1

Sheets | Charts | SmartArt Graphics | WordArt

| | A | B | C | J | K | L | M | N | O | P | R | S | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | product | version | file | \echo | \expressio | \for | \foreach | \functionDef | \globa | \goto | \if | \inlineHTML | \interfa |
| 2 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 3 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 38 | 0 | 0 | 0 | 3 | 0 | 11 | 0 | |
| 4 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 8 | 18 | 2 | 0 | 0 | 0 | 0 | 10 | 6 | |
| 5 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 15 | 46 | 0 | 0 | 0 | 0 | 0 | 7 | 15 | |
| 6 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 16 | 3 | 0 | 0 | 2 | 0 | 5 | 0 | |
| 7 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 76 | 0 | 0 | 0 | 0 | 0 | 18 | 0 | |
| 8 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 22 | 0 | 0 | 0 | 3 | 0 | 6 | 0 | |
| 9 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 10 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 21 | 0 | 0 | 0 | 2 | 0 | 6 | 0 | |
| 11 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 178 | 8 | 2 | 0 | 9 | 0 | 43 | 0 | |
| 12 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 13 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 33 | 4 | 0 | 0 | 2 | 0 | 13 | 0 | |
| 14 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 17 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | |
| 15 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 61 | 2 | 2 | 0 | 8 | 0 | 19 | 0 | |
| 16 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 72 | 1 | 0 | 0 | 0 | 0 | 22 | 0 | |
| 17 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 8 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |
| 18 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 152 | 12 | 0 | 13 | 3 | 0 | 42 | 0 | |
| 19 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 9 | 0 | 0 | 0 | 1 | 0 | 6 | 0 | |
| 20 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 21 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 22 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 23 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 24 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 13 | 26 | 0 | 0 | 0 | 0 | 0 | 6 | 14 | |
| 25 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 25 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 26 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 117 | 1 | 0 | 0 | 8 | 0 | 45 | 0 | |
| 27 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 28 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 20 | 23 | 2 | 0 | 0 | 0 | 0 | 4 | 16 | |
| 29 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 5 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | |
| 30 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 0 | 17 | 2 | 0 | 0 | 1 | 0 | 6 | 0 | |
| 31 | osCommerce | 2.3.1 | /export/scratch1/hills/corpus/ | 20 | 77 | 4 | 0 | 0 | 0 | 0 | 17 | 18 | |

Sheet1

# Evolving Sources of Data

- Type literal solution fixed one problem, introduces another
- Hardcoded type of CSV now needs to be kept in sync with type computed for CSV file contents
- CSV file changes require changes in our code, even if we don’t care about the new data elements

# Summary: External Resource Challenges (Partial)

- Type mapping not always trivial
  - Implicitly-typed data needs inference
- Data sources evolve -- type/code maintenance issues

# Summary: External Resource Challenges

- All libraries may have different ways of identifying where data is coming from, want something uniform
- Type mapping not always trivial
  - Explicitly-typed data can have complex mappings
  - Implicitly-typed data needs inference
- Data sources evolve -- type/code maintenance issues

# Proposed Solution in Rascal: Rascal Resources

# Proposed Solution in Rascal: Rascal Resources

- Uniform mechanism for naming resources

# Proposed Solution in Rascal: Rascal Resources

- Uniform mechanism for naming resources
- Static types ensure safe use of resources

# Proposed Solution in Rascal: Rascal Resources

- Uniform mechanism for naming resources
- Static types ensure safe use of resources
- Code generation at module import time provides resource specific modules, supports co-evolution

# Proposed Solution in Rascal: Rascal Resources

- Uniform mechanism for naming resources
- Static types ensure safe use of resources
- Code generation at module import time provides resource specific modules, supports co-evolution
- Use of Rascal code allows inspection and debugging

# Proposed Solution in Rascal: Rascal Resources

- Uniform mechanism for naming resources
- Static types ensure safe use of resources
- Code generation at module import time provides resource specific modules, supports co-evolution
- Use of Rascal code allows inspection and debugging
- Rascal to Java bridge allows interaction with wide variety of existing resources

# Resources for the Resource User

- Resources accessed as modules, using module import
- Imported module provides types and typed access functions
- Imported modules generated at load time, before any code in the module runs
- Signature of imported module used to check uses of resource for correctness
- Generated module available for inspection, debugging

# Example: Importing a CSV Resource

```
import lang::csv::IO;


import Versions = |csv+project://PHPAnalysis/src/lang/php/
extract/csvs/Versions.csv?funname=getVersions|;
```

# Example: Importing a CSV Resource

```
import lang::csv::IO;
```

Imports Resource Generator

```
import Versions = |csv+project://PHPAnalysis/src/lang/php/
extract/csvs/Versions.csv?funname=getVersions|;
```

# Example: Importing a CSV Resource

```
import lang::csv::IO;
```

Imports Resource Generator

```
import Versions = |csv+project://PHPAnalysis/src/lang/php/
extract/csvs/Versions.csv?funname=getVersions|;
```

Imports Resource

# Example: Importing a CSV Resource

```
import lang::csv::IO;

import Versions = |csv+project://PHPAnalysis/src/lang/php/
extract/csvs/Versions.csv?funname=getVersions|;
```

Imports Resource Generator

Imports Resource

Indicates CSV Resource

# Example: Importing a CSV Resource

```
import lang::csv::IO;

import Versions = |csv+project://PHPAnalysis/src/lang/php/
extract/csvs/Versions.csv?funname=getVersions|;
```

Imports Resource Generator

Imports Resource

Indicates CSV Resource

Specifies Accessor Function

# Example: Using the Imported Resource

```
ver = getVersions();

getVersionsType: {
  <"CodeIgniter","1.5.4","2007-07-15","4.3.2","">,
  <"Drupal","5.10","2008-08-14","4.3.5","">,
  <"CakePHP","1.2.3-0","2009-05-04","4.0.0","">,
  <"PEAR","1.4.6","2006-01-06","4.2.0","">, ... }

requiresPHP5 = { v | v <- ver, /^5/ := v.RequiredPHPVersion };

rel[str Product, str Version, str ReleaseDate,
    str RequiredPHPVersion, str Comments]: {
  <"Drupal","7.5","2011-07-27","5.2.4","">,
  <"ZendFramework","1.11.4","2011-03-03","5.2.4","">,
  <"ZendFramework","1.10.4","2010-04-28","5.2.4","">,
  <"ZendFramework","1.10.1","2010-02-09","5.2.4","">, ... }
```

# Example: Using the Imported Resource

Get Resource Data with Accessor

```
ver = getVersions();

getVersionsType: {
  <"CodeIgniter","1.5.4","2007-07-15","4.3.2","">,
  <"Drupal","5.10","2008-08-14","4.3.5","">,
  <"CakePHP","1.2.3-0","2009-05-04","4.0.0","">,
  <"PEAR","1.4.6","2006-01-06","4.2.0","">, ... }

requiresPHP5 = { v | v <- ver, /^5/ := v.RequiredPHPVersion };

rel[str Product, str Version, str ReleaseDate,
    str RequiredPHPVersion, str Comments]: {
  <"Drupal","7.5","2011-07-27","5.2.4","">,
  <"ZendFramework","1.11.4","2011-03-03","5.2.4","">,
  <"ZendFramework","1.10.4","2010-04-28","5.2.4","">,
  <"ZendFramework","1.10.1","2010-02-09","5.2.4","">, ... }
```

# Example: Using the Imported Resource

```
ver = getVersions();

getVersionsType: {
  <"CodeIgniter","1.5.4","2007-07-15","4.3.2","">,
  <"Drupal","5.10","2008-08-14","4.3.5","">,
  <"CakePHP","1.2.3-0","2009-05-04","4.0.0","">,
  <"PEAR","1.4.6","2006-01-06","4.2.0","">, ... }

requiresPHP5 = { v | v <- ver, /^5/ := v.RequiredPHPVersion };

rel[str Product, str Version, str ReleaseDate,
    str RequiredPHPVersion, str Comments]: {
  <"Drupal","7.5","2011-07-27","5.2.4","">,
  <"ZendFramework","1.11.4","2011-03-03","5.2.4","">,
  <"ZendFramework","1.10.4","2010-04-28","5.2.4","">,
  <"ZendFramework","1.10.1","2010-02-09","5.2.4","">, ... }
```

Get Resource Data with Accessor

Result, as Relation

# Example: Using the Imported Resource

```
ver = getVersions();

getVersionsType: {
  <"CodeIgniter","1.5.4","2007-07-15","4.3.2","">,
  <"Drupal","5.10","2008-08-14","4.3.5","">,
  <"CakePHP","1.2.3-0","2009-05-04","4.0.0","">,
  <"PEAR","1.4.6","2006-01-06","4.2.0","">, ... }

requiresPHP5 = { v | v <- ver, /^5/ := v.RequiredPHPVersion };

rel[str Product, str Version, str ReleaseDate,
    str RequiredPHPVersion, str Comments]: {
  <"Drupal","7.5","2011-07-27","5.2.4","">,
  <"ZendFramework","1.11.4","2011-03-03","5.2.4","">,
  <"ZendFramework","1.10.4","2010-04-28","5.2.4","">,
  <"ZendFramework","1.10.1","2010-02-09","5.2.4","">, ... }
```

Get Resource Data with Accessor

Result, as Relation

Filter, Using Field Names

# Resources for the Resource Designer

- Designer adds a resource by writing a standard Rascal function with a special tag

- Function gets information needed to access resource from a Rascal source location

- Code generation used to generate a module that loads the resource

- Generated code uses Rascal libraries, includes type information to provide typed access

# Example: The CSV Resource Handler

```
@resource{csv}
public str generate(str moduleName, loc uri) {
    map[str,str] options = uri.params;
    str funname = "resourceValue";
    if ("funname" in options) {
        funname = options["funname"];
        options = domainX(options,{"funname"});
    }
    type[value] csvType = getCSVType(uri, options);
    mbody = "module <moduleName>
            'import lang::csv::IO;
            'alias <funname>Type = <csvType>;
            'public <funname>Type <funname>() {
            '   return readCSV(#<csvType>, <uri>, <options>);
            '}
            '";
    return mbody;
}
```

# Example: The CSV Resource Handler

Name of Module to Generate

```
@resource{csv}
public str generate(str moduleName, loc uri) {
    map[str,str] options = uri.params;
    str funname = "resourceValue";
    if ("funname" in options) {
        funname = options["funname"];
        options = domainX(options,{"funname"});
    }
    type[value] csvType = getCSVType(uri, options);
    mbody = "module <moduleName>
            'import lang::csv::IO;
            'alias <funname>Type = <csvType>;
            'public <funname>Type <funname>() {
            '   return readCSV(#<csvType>, <uri>, <options>);
            '}
            '";
    return mbody;
}
```

# Example: The CSV Resource Handler

Name of Module to Generate

Location with Access Info

```
@resource{csv}
public str generate(str moduleName, loc uri) {
    map[str,str] options = uri.params;
    str funname = "resourceValue";
    if ("funname" in options) {
        funname = options["funname"];
        options = domainX(options,{"funname"});
    }
    type[value] csvType = getCSVType(uri, options);
    mbody = "module <moduleName>
            'import lang::csv::IO;
            'alias <funname>Type = <csvType>;
            'public <funname>Type <funname>() {
            '   return readCSV(#<csvType>, <uri>, <options>);
            '}
            '";
    return mbody;
}
```

# Example: The CSV Resource Handler

```
@resource{csv}
public str generate(str moduleName, loc uri) {
    map[str,str] options = uri.params;
    str funname = "resourceValue";
    if ("funname" in options) {
        funname = options["funname"];
        options = domainX(options,{"funname"});
    }
    type[value] csvType = getCSVType(uri, options);
    mbody = "module <moduleName>
            'import lang::csv::IO;
            'alias <funname>Type = <csvType>;
            'public <funname>Type <funname>() {
            '   return readCSV(#<csvType>, <uri>, <options>);
            '}
            '";
    return mbody;
}
```

Name of Module to Generate

Location with Access Info

Type Computation

# Example: The CSV Resource Handler

```
@resource{csv}
public str generate(str moduleName, loc uri) {
    map[str,str] options = uri.params;
    str funname = "resourceValue";
    if ("funname" in options) {
        funname = options["funname"];
        options = domainX(options,{"funname"});
    }
    type[value] csvType = getCSVType(uri, options);
    mbody = "module <moduleName>
            'import lang::csv::IO;
            'alias <funname>Type = <csvType>;
            'public <funname>Type <funname>() {
            '   return readCSV(#<csvType>, <uri>, <options>);
            '}
            '";
    return mbody;
}
```

Name of Module to Generate

Location with Access Info

Type Computation

Code Generation

# Example: The Generated PHP Features CSV Module

```
@generated
module Feats
import lang::csv::IO;
alias getFeatsType = rel[str \product,str \version,str \file,int \break,int \classDef,int \const,int \continue,int \declare,int \do,int
\echo,int \expressionStatementChainRule,int \for,int \foreach,int \functionDef,int \global,int \goto,int \haltCompiler,int \if,int
\inlineHTML,int \interfaceDef,int \traitDef,int \label,int \namespace,int \return,int \static,int \switch,int \throw,int \tryCatch,int
\unset,int \use,int \while,int \array,int \fetchArrayDim,int \fetchClassConst,int \assign,int \assignWithOperationBitwiseAnd,int
\assignWithOperationBitwiseOr,int \assignWithOperationBitwiseXor,int \assignWithOperationConcat,int \assignWithOperationDiv,int
\assignWithOperationMinus,int \assignWithOperationMod,int \assignWithOperationMul,int \assignWithOperationPlus,int
\assignWithOperationRightShift,int \assignWithOperationLeftShift,int \listAssign,int \refAssign,int \binaryOperationBitwiseAnd,int
\binaryOperationBitwiseOr,int \binaryOperationBitwiseXor,int \binaryOperationConcat,int \binaryOperationDiv,int \binaryOperationMinus,int
\binaryOperationMod,int \binaryOperationMul,int \binaryOperationPlus,int \binaryOperationRightShift,int \binaryOperationLeftShift,int
\binaryOperationBooleanAnd,int \binaryOperationBooleanOr,int \binaryOperationGt,int \binaryOperationGeq,int \binaryOperationLogicalAnd,int
\binaryOperationLogicalOr,int \binaryOperationLogicalXor,int \binaryOperationNotEqual,int \binaryOperationNotIdentical,int
\binaryOperationLt,int \binaryOperationLeq,int \binaryOperationEqual,int \binaryOperationIdentical,int \unaryOperationBooleanNot,int
\unaryOperationBitwiseNot,int \unaryOperationPostDec,int \unaryOperationPreDec,int \unaryOperationPostInc,int \unaryOperationPreInc,int
\unaryOperationUnaryPlus,int \unaryOperationUnaryMinus,int \new,int \castToInt,int \castToBool,int \castToFloat,int \castToString,int
\castToArray,int \castToObject,int \castToUnset,int \clone,int \closure,int \fetchConst,int \empty,int \suppress,int \eval,int \exit,int
\call,int \methodCall,int \staticCall,int \include,int \instanceOf,int \isSet,int \print,int \propertyFetch,int \shellExec,int \ternary,int
\fetchStaticProperty,int \scalar,int \var,int \propertyDef,int \classConstDef,int \methodDef,int \traitUse];
public getFeatsType getFeats() {
   return readCSV(#rel[str \product,str \version,str \file,int \break,int \classDef,int \const,int \continue,int \declare,int \do,int \echo,int
\expressionStatementChainRule,int \for,int \foreach,int \functionDef,int \global,int \goto,int \haltCompiler,int \if,int \inlineHTML,int
\interfaceDef,int \traitDef,int \label,int \namespace,int \return,int \static,int \switch,int \throw,int \tryCatch,int \unset,int \use,int
\while,int \array,int \fetchArrayDim,int \fetchClassConst,int \assign,int \assignWithOperationBitwiseAnd,int \assignWithOperationBitwiseOr,int
\assignWithOperationBitwiseXor,int \assignWithOperationConcat,int \assignWithOperationDiv,int \assignWithOperationMinus,int
\assignWithOperationMod,int \assignWithOperationMul,int \assignWithOperationPlus,int \assignWithOperationRightShift,int
\assignWithOperationLeftShift,int \listAssign,int \refAssign,int \binaryOperationBitwiseAnd,int \binaryOperationBitwiseOr,int
\binaryOperationBitwiseXor,int \binaryOperationConcat,int \binaryOperationDiv,int \binaryOperationMinus,int \binaryOperationMod,int
\binaryOperationMul,int \binaryOperationPlus,int \binaryOperationRightShift,int \binaryOperationLeftShift,int \binaryOperationBooleanAnd,int
\binaryOperationBooleanOr,int \binaryOperationGt,int \binaryOperationGeq,int \binaryOperationLogicalAnd,int \binaryOperationLogicalOr,int
\binaryOperationLogicalXor,int \binaryOperationNotEqual,int \binaryOperationNotIdentical,int \binaryOperationLt,int \binaryOperationLeq,int
\binaryOperationEqual,int \binaryOperationIdentical,int \unaryOperationBooleanNot,int \unaryOperationBitwiseNot,int \unaryOperationPostDec,int
\unaryOperationPreDec,int \unaryOperationPostInc,int \unaryOperationPreInc,int \unaryOperationUnaryPlus,int \unaryOperationUnaryMinus,int
\new,int \castToInt,int \castToBool,int \castToFloat,int \castToString,int \castToArray,int \castToObject,int \castToUnset,int \clone,int
\closure,int \fetchConst,int \empty,int \suppress,int \eval,int \exit,int \call,int \methodCall,int \staticCall,int \include,int
\instanceOf,int \isSet,int \print,int \propertyFetch,int \shellExec,int \ternary,int \fetchStaticProperty,int \scalar,int \var,int
\propertyDef,int \classConstDef,int \methodDef,int \traitUse], |project://PHPAnalysis/src/lang/php/extract/csvs/FeaturesByFile.csv?
funname=getFeats|, ());
}
```

# Example: The Generated PHP Versions CSV Module

```
@generated
module Versions
import lang::csv::IO;

alias getVersionsType =
   rel[str \Product,str \Version,str \ReleaseDate,
   str \RequiredPHPVersion,str \Comments];

public getVersionsType getVersions() {
    return readCSV(
   #rel[str \Product,str \Version,str \ReleaseDate,
      str \RequiredPHPVersion,str \Comments],
   |project://PHPAnalysis/src/lang/php/extract/csvs/Versions.csv?
      funname=getVersions|, ());
}
```

# Example: The Generated PHP Versions CSV Module

```
@generated
module Versions
import lang::csv::IO;

alias getVersionsType =
   rel[str \Product,str \Version,str \ReleaseDate,
   str \RequiredPHPVersion,str \Comments];

public getVersionsType getVersions() {
    return readCSV(
   #rel[str \Product,str \Version,str \ReleaseDate,
      str \RequiredPHPVersion,str \Comments],
   |project://PHPAnalysis/src/lang/php/extract/csvs/Versions.csv?
      funname=getVersions|, ());
}
```

Computed Type of Resource

# Example: The Generated PHP Versions CSV Module

```
@generated
module Versions
import lang::csv::IO;

alias getVersionsType =
   rel[str \Product,str \Version,str \ReleaseDa
   str \RequiredPHPVersion,str \Comments];

public getVersionsType getVersions() {
    return readCSV(
   #rel[str \Product,str \Version,str \ReleaseDate,
      str \RequiredPHPVersion,str \Comments],
   |project://PHPAnalysis/src/lang/php/extract/csvs/Versions.csv?
      funname=getVersions|, ());
}
```

Computed Type of Resource

Typed Resource Import

# Example: The Generated PHP Versions CSV Module

```
@generated
module Versions
import lang::csv::IO;

alias getVersionsType =
   rel[str \Product,str \Version,str \ReleaseDa
   str \RequiredPHPVersion,str \Comments];

public getVersionsType getVersions() {
    return readCSV(
   #rel[str \Product,str \Version,str \ReleaseDate,
      str \RequiredPHPVersion,str \Comments],
   |project://PHPAnalysis/src/lang/php/extract/csvs/Versions.csv?
      funname=getVersions|, ());
}
```

Computed Type of Resource

Typed Resource Import

Returns Resource Type, Statically Checkable Code

# This Seems to Work Well, Any Limitations?

- Generated module still needs to be type checked
- Schema could change “out from under” resource
- Manipulation of resources is completely “in core”, not realistic for large resources
- Resource connection information needs to be expressible as a URI, could be cumbersome in some cases, cannot build URI dynamically in code

# Related Work

- Dynamic languages
- Database/XML data access (ORM, LINQ, etc)
- F# Type Providers
  - We provide modules, bigger granularity
  - Eager vs lazy generation of provided types
  - Code generation versus reflection

# Future Work


- Can we check the safety of the generated code at generation time?

- How can we work with resources that are too large to load into memory?

- Write more resources!

# Thank you!
# Any Questions?

- Rascal: http://www.rascal-mpl.org
- SEN1: http://www.cwi.nl/sen1
- Me: http://www.cwi.nl/~hills