101companies

http://101companies.org/

Ralf Lämmel (Software Languages Team, University of Koblenz-Landau)
101companies is a software chrestomathy

- Chrestomathy; Greek vs. English
- chresto — useful
- mathein — to learn
- A chrestomathy collecting software systems
Chrestomathies in linguistics and philology

A collection of literary passages in typically one language from one or more authors compiled by one or more chrestomathy authors as an aid in learning a language.
Program and software chrestomathies

- The Evolution of a Haskell programmer
- 99 Bottles of Beer
- OO Shapes
- Rosetta Code
- Beautiful Code
- The Computer Language Benchmarks Game
- 101companies (The `101' project)

http://softlang.uni-koblenz.de/chrestomathy/

© 2010-16 101companies Project
“101” is short for
“101project” which is short for
“101companies Project”

What’s 101?

“101companies” refers to
“101 ways of building a
Human-resources management system
for a conceived company”.

© 2010-16 101companies Project
What's 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...
What’s 101?

101 = 101system + 101contributions + 101repo + 101wiki + 101worker + 101explorer + 101people + ...

A conceived human resources management system (HRMS) to be modeled and implemented time and again, thereby collecting community knowledge about software languages, technologies, and concepts.
What’s 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Data model with companies, departments, and employees. Functional requirements such as “salary total”. Non-functional requirements such as “distribution”.

© 2010-16 101companies Project
What's 101?

Company

<table>
<thead>
<tr>
<th>name : String</th>
</tr>
</thead>
</table>

Department

<table>
<thead>
<tr>
<th>name : String</th>
</tr>
</thead>
</table>

sub-departments

employees

| 0..* |

Employee

<table>
<thead>
<tr>
<th>name : String</th>
</tr>
</thead>
<tbody>
<tr>
<td>address : String</td>
</tr>
<tr>
<td>salary : Double</td>
</tr>
</tbody>
</table>

manager

+ ...
What’s 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Mostly, these are implementations of the 101system; they vary in the following dimensions:

Used programming languages
Used libraries/frameworks
Implemented features sets
Applied design patterns
...

© 2010-16 101companies Project
What's 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Company X:
Swing + JDBC

Company Y:
SWT + Hibernate

Company Z:
GWT + MongoDB

...
What's 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

A confederated (GitHub-based) open-source repository which holds all code artifacts:

- Code for contributions
- Other illustrative code
- Infrastructure of 101
- Version history of 101wiki
- ...

© 2010-16 101companies Project
What’s 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Different features
Different designs
Different technologies
Different languages
What's 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101explorer
+ 101people
+ ...

Technologies / Languages
What's 101?

101

= 101system
+ 101contributions
+ **101repo**
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...
What's 101?

101 = 101system + 101contrib + 101repo + 101wiki + 101worker + 101explorer + 101people + ...
What's 101?

101 = 101system + 101contributions + 101repo + 101wiki + 101worker + 101explorer + 101people + ...

A highly structured, semantically enriched, and linked-data enabled wiki for text and triples along different namespaces:

- Docs for contributions
- Feature model of 101system
- Entries for sw languages
- Entries for sw technologies
- Entries for sw concepts
- ...

© 2010-16 101companies Project
101wiki’s namespaces

- **Namespace**: all namespaces including those listed below.
- **Language**: software languages such as Haskell, XML, or UML.
- **Technology**: software technologies such as JUnit, GitHub, or Ruby on Rails.
- **Concept**: software concepts such as parsing, abstraction, or visitor pattern.
- **Feature**: features (requirements) of the 101system such as Cut or Total.
- **Contribution**: implementations, models, etc. of the 101system.
- **Contributor**: open-source developers and wiki authors contributing to the 101project.
- **Theme**: themes (collections) of contributions addressing stakeholder perspectives.
- **Vocabulary**: vocabularies of software concepts, e.g., for Software engineering.
- **Course**: open online courses leveraging resources of the 101project.
- **Script**: scripts for individual lectures, labs, etc. in courses.
- **Module**: modules of the 101worker deriving resources and dumps.
What's 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

A wiki page for a contribution

Contribution: antlrObjects

- Headline
- Characteristics
- Illustration
- Relationships
- Architecture
- Usage
- Metadata
What's 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Top 2 sections of the earlier page

Headline

Object/Test mapping for Java with Technology:ANTLR for parsing.

Characteristics

An Technology:ANTLR-based parser for a concrete syntax of companies is provided. The underlying grammar is LL(1). The parser translates text into company objects. To this end, a straightforward object model for companies is used. (In fact, the object model is very similar to the one of Contribution:JavaComposition. The mapping is described by the semantic actions within the Technology:ANTLR-based parser description. Pretty printing is also supported in the manner that objects can be exported in the concrete syntax. Pretty printing provides an implementation of Feature:Unparsing. Basic operations are implemented on top of the object model in basic OO programming style.
What’s 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

The metadata section of the earlier page:

- this `developedBy` Contributor: Ralf Lämmel
- this `implements` Feature: Cut
- this `implements` Feature: Hierarchical company
- this `implements` Feature: Parsing
- this `implements` Feature: Total
- this `implements` Feature: Unparsing
- this `instanceOf` Namespace: Contribution
- this `instanceOf` Theme: ANTLR
- this `instanceOf` Theme: Java mapping
- this `uses` Language: Java
- this `uses` Technology: ANTLR
- this `uses` Technology: Gradle
- this `uses` Technology: JUnit
What’s 101?

101

= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

An open, extensible, computational infrastructure for analyzing 101repo&wiki and synthesizing information:

- Dump of 101wiki
- Facts extracted from 101repo
- Inferred metadata of 101repo
- ...

© 2010–16 101companies Project
What's 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Derived metrics data for a source file

{  
  size: 769,
  loc: 25,
  ncloc: 26,
  relevance: "system"
}

© 2010-16 101companies Project
What's 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

A linked data layer on top of 101 to surface all entities in a systematically discoverable and connected manner:

- List of all namespaces
- List of namespace members
- List of subfolders and files
- Links to GitHub, 101wiki, ...
- Support for JSON, RDF, HTML
- ...

© 2010–16 101companies Project
What’s 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Members of contributions namespace

101explorer

Summary
- Headline
- The namespace for @contributions
- Name contributions
- Namespace Namespace
- Classifier Namespace

Parts
- Members
  - FsharpClasses
  - antlrAcceptor
  - antlrLexer
  - antlrObjects
  - antlrParser
  - antlrTrees
  - argoUML
  - aspectJ
  - atl
  - atlCutPlugin
  - atlPluginUsage
  - atlTotalPlugin
- basicLb
- callref-rascal
- clojure
- clojureRefs
- cobol
- coq
- cppMpl
- csharp
- csharpAspnetMvc
- csharpDryadLinq
- csharpLinqToXml
- dom

© 2010-16 101companies Project
What's 101?

101 = 101 system + 101 contributions + 101 repo + 101 wiki + 101 worker + 101 explorer + ...

The root folder of a contribution...
What is 101?

101 = 101 system + 101 contributions + 101 repo + 101 wiki + 101 worker + 101 explorer + ...
What’s 101?

101
= 101system
+ 101contributions
+ 101repo
+ 101wiki
+ 101worker
+ 101explorer
+ 101people
+ ...

Various stakeholders:

- Contributors (development)
- Technologists (demonstration)
- Ontologists (formalization)
- Teachers (utilization)
- Learners (utilization)
- ...

© 2010-16 101companies Project
What's 101 good for?
What’s 101 good for?

- The guide through the software galaxy
- Technology comparison
- Programming education
- Test-harness for language technology
- Feature detection
- Fact extraction
- ...

© 2010-16 101companies Project
What is 101 good for?

The Hitchhiker's Guide to the Software Galaxy

Wannabe Wikipedia for Software Developers. :-)

© 2010-16 101companies Project
What is 101 good for?

Too much software technologies.
Too much software languages.
Too little time.

101 supports examples, abstractions, and analogies.
101 is a knowledge resource for technological space travel.

101 is a knowledge resource for technological space travel.
101 is a knowledge resource for technology comparison.
101 is a knowledge resource for programming education.

- Lecture First steps
- Lecture Basic software engineering
- Lecture Searching and sorting
- Lecture Basic data modeling
- Lecture Higher-order functions
- Lecture Type-class polymorphism
- Lecture Functors and friends
- Lecture Monads
- Lecture Parsing and unparsing
- Dry run for final
- Lecture Generic functions
- Final

A functional programming course

© 2010-16 101companies Project
101 is a knowledge resource for programming education.

<table>
<thead>
<tr>
<th>Contribution/Term</th>
<th>haskellList (4/10)</th>
<th>haskellStarter (3/2/2)</th>
<th>haskellLambda (2/3/1)</th>
<th>haskellMonad (2/2/0)</th>
<th>haskellParsec (2/2/0)</th>
<th>haskellAcceptor (2/0/0)</th>
<th>haskellPicker (2/0/2)</th>
<th>haskellEngineer (0/5/0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous function (2/10)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Map function (2/10)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td></td>
</tr>
<tr>
<td>Parsing (2/8)</td>
<td>⬤ ⬤</td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recursion (2/2)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fold function (2/1)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parser combinator (2/1)</td>
<td>⬤ ⬤</td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function application (1/1)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lambda abstraction (1/1)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>String (1/1)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤ ⬤</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type class (1/1)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type-class instance (1/0)</td>
<td>⬤ ⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function (0/5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬤ ⬤</td>
<td></td>
</tr>
</tbody>
</table>

Concept coverage of contributions
Q&A

😊 Thanks!