

# Code Reuse

CS 695 / SWE 699: Programming Tools

Fall 2023



# Today

- Part 1 (Lecture)(~45 mins)
- Part 2: Tech Talks (60 mins)
  - Two tech talks
  - 10 min break
  - Two tech talks
- Part 3: (Short In-Class Activity)(30 mins)

# Logistics

- HW 4 checkpoint due 11/1
- HW 4 due 11/29
- Tech talks should now all be on the dates originally scheduled

# Overview

- What is code reuse, how developers do it, and what makes it hard?
- Tools to support code reuse

# What is reuse?

- Making use of previously written code rather than writing new code
- Often, reuse takes form of reusing a *library* or a *framework*
- Once made choice to reuse a library or framework, need to understand how to achieve specific behavior with library or framework
- Often finding code *snippets* that achieve desired behavior

# Reuse of Uses

- Developers rely extensively on *examples* to understand how to instantiate objects

Reuse Activity	Specific Strategies Observed
<i>Finding a Usage Context</i>	Find senders of messages defined for target class, focusing on “interesting” ones
<i>Evaluating a Usage Context</i>	
Executing the Context	Look for references to application data objects in the openOn: method.
Assessing Similarity	Open example application “on” a basic data object from the project.
Studying Bits of Context	Reason by analogy from familiar syntactic construction, e.g., button1Down:
Deciding to Subclass	Look for use of unmappable instance variables or many messages to “self.”
<i>Debugging a Usage Context</i>	
Getting an Instance Running	Focus first on the openOn: code for starting up a window.
Borrowing the Context	Use multiple browsers to work from related pieces of context. Carry out step-by-step replacement of message parameters. Edit what does not compile. Develop a method to substitute one data object for another.
Analysis by Testing	Adapt or develop the method identified in the notification “message not understood.”

# Some possible reuse strategies

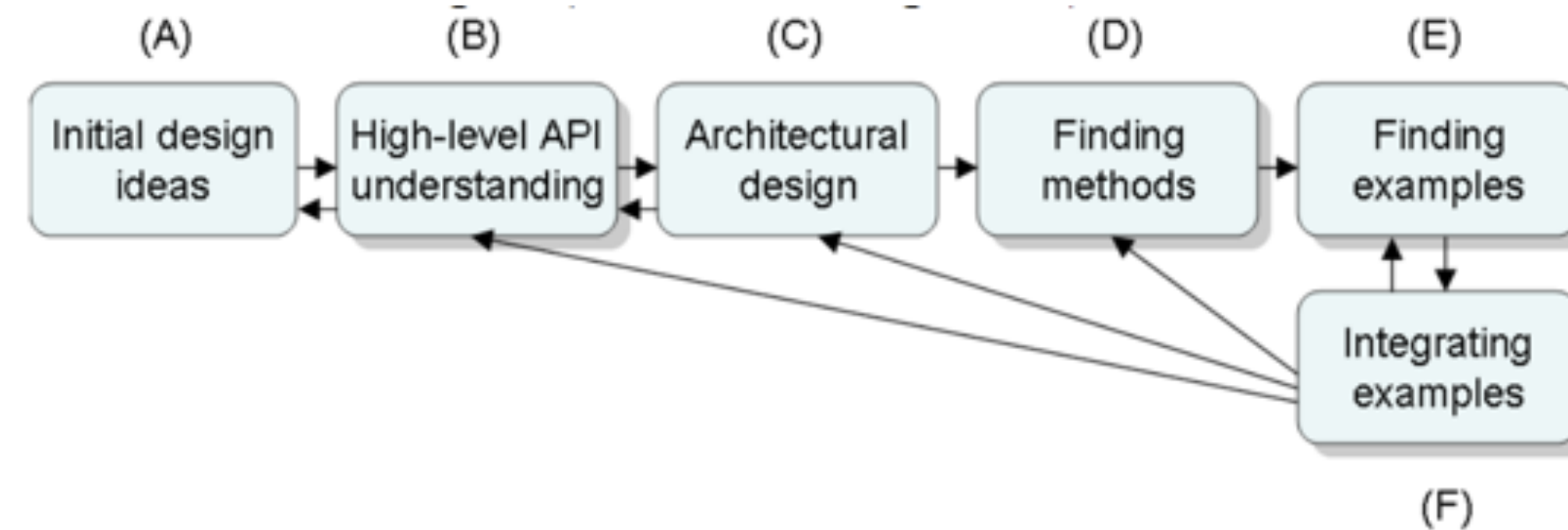
- Read the documentation
- Find tutorials
- Find StackOverflow snippets
- Find similar code in your own codebase
- Call API functions, see what they return

# Opportunistic vs. systematic

- Opportunistic developers more likely to start with example code
- Systematic developers more likely to read the documentation first



# Example reuse process



B: read tutorials, articles, projects to understand domain

D: searched Google, often seeking descriptions in API of specific classes & methods to use

E: looked for examples of how to use specific methods

# Types of reuse

WEB SESSION INTENTION:	LEARNING	CLARIFICATION	REMINDER
Reason for using Web	Just-in-time learning of unfamiliar concepts	Connect high-level knowledge to implementation details	Substitute for memorization ( <i>e.g.</i> , language syntax or function usage lookup)
Web session length	Tens of minutes	~ 1 minute	< 1 minute
Starts with web search?	Almost always	Often	Sometimes
Search terms	Natural language related to high-level task	Mix of natural language and code, cross-language analogies	Mostly code ( <i>e.g.</i> , function names, language keywords)
Example search	“ajax tutorial”	“javascript timer”	“mysql_fetch_array”
Num. result clicks	Usually several	Fewer	Usually zero or one
Num. query refinements	Usually several	Fewer	Usually zero
Types of webpages visited	Tutorials, how-to articles	API documentation, blog posts, articles	API documentation, result snippets on search page
Amount of code copied from Web	Dozens of lines ( <i>e.g.</i> , from tutorial snippets)	Several lines	Varies
Immediately test copied code?	Yes	Not usually, often trust snippets	Varies

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems (CHI '09)*, 1589-1598.

# Types of reuse

- Learning—relies on selecting highest quality tutorials tutorials
  - e.g., “update web page without reloading php”
- Clarification—learning syntax based on exiting understanding of the domain concepts
  - e.g., reminding use of syntax of HTML forms
  - Often search by analogy to domain concepts in other languages / frameworks
    - e.g., Perl has a function to format dates as strings, what’s the one for PHP?
- Reminder—using web as external memory aid
  - e.g., forgot a word in a long function name
  - e.g., 6 lines of code necessary to connect and disconnect from MySQL database copied hundreds of times by individual

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems (CHI '09)*, 1589-1598.

# Design implications

- Web tutorials used for just in time learning
  - —> Tutorials should be tightly coupled to code, where developers can play in sandbox then read tutorial content to understand problems when do not work
- Web search used as translator from intention to terminology & syntax
  - —> tools could compare code from search results to users code to automatically locate errors
  - —> search should be integrated into autocomplete
- Developers delay testing, esp for routine functionality
  - —> Tools should assist with adaption by highlighting variables and literals in reused snippets & provide link back to original source

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems (CHI '09)*, 1589-1598.

# Challenges with reuse

- Mapping an abstract conceptual solution into the appropriate elements
  - *“How do I create a rectangle? Why is there no Rectangle tool?”*
- Understanding control & data flow, hidden dependencies due to run-time binding or inheritance, between classes in the API
  - *“I’m over-riding SelectionTool, and in particular mouseDown() so that when the figure is clicked the box is drawn. This bit works, however when trying to drag the figure, if I do something similar the rectangle flickers like mad.”*
- Understanding how functionality works
  - *“How does ... work?”, “What does ... do?” or, “Where is ... defined/created/called?”*
- Making changes consistent w/ architectural constraints of API
  - Violating constraints of MVC architecture by passing references in prohibited ways

Douglas Kirk, Marc Roper, and Murray Wood. 2007. Identifying and addressing problems in object-oriented framework reuse. *Empirical Softw. Eng.* 12, 3 (June 2007), 243-274.

# Challenges with reuse

- **Design** barriers—inherent cognitive difficulties of the programming problem, separate from notation used
  - I don't know what I want the computer to do
- **Selection** barriers—finding programming interfaces available to achieve a particular behavior
  - I don't know what to use
- **Coordination** barriers—constraints governing how languages & libraries can be combined
  - I don't know how to make them work together
- **Use** barriers—determining how API how to use API
  - I don't know how to use it
- **Understanding** barriers—environment properties such as compile & runtime errors that prevent seeing behavior
  - It didn't do what I expected
- **Information** barriers—environment properties that prevent understanding runtime execution state
  - I think I know why didn't behave as expected, but don't know how to check

# Vocabulary problem

- Developers are familiar with concepts using one set of terminology.
- API, tutorials, or other resources use different terminology
- How do developers find the right concepts with alternative terms?

# Challenges may vary by context

- Size of desired snippet
  - Reusing a line of code? A whole algorithm?
- Alternatives
  - How many alternatives are there? How important is it to find the best alternative?
- Integration
  - What libraries or frameworks does a snippet require? How can they be integrated?



# Challenges working with API documentation

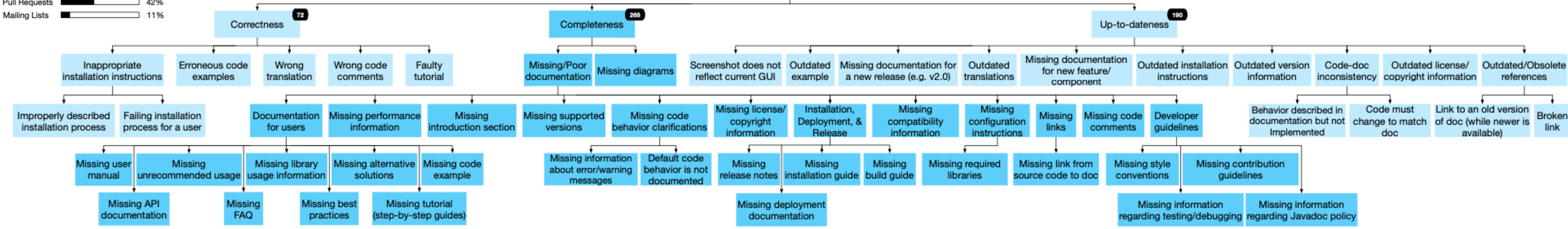
- Goal: Parse a Java source file w/ Eclipse
- Answer:

```
IFile file = ...;  
ICompilationUnit cu =  
    JavaCore.createCompilationUnitFrom(file);  
ASTNode ast = AST.parseCompilationUnit(cu, false);
```

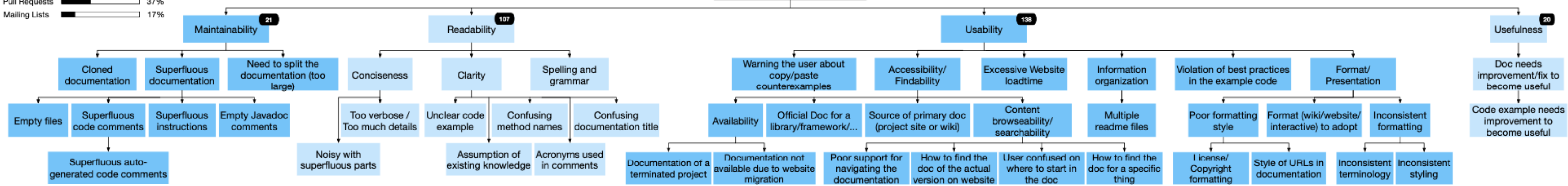
- Challenges
  - Want to work with files and ASTNodes, but key class is JavaCore
  - No connection from what you might know about ASTNode and IFile to JavaCore



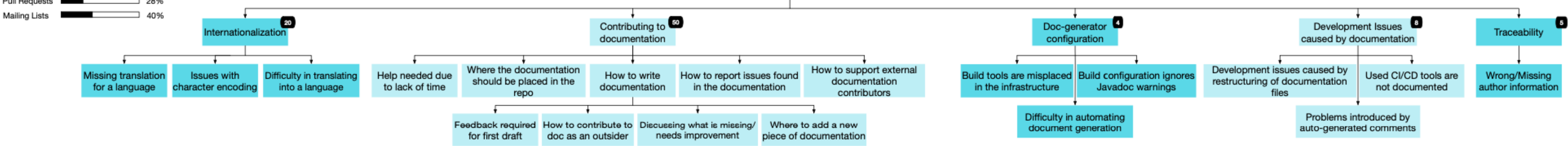
## Information Content (What) 485



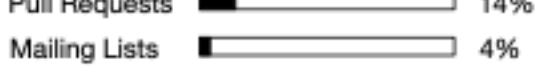
## Information Content (How) 255



## Process Related 81



Emad Aghajani, Csaba Nagy, Olga Lucero Vega-Márquez, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, and Michele Lanza. 2019. Software documentation issues unveiled. In Proceedings of the 41st International Conference on Software Engineering (ICSE '19). IEEE Press, Piscataway, NJ, USA, 1199-1210. DOI: <https://doi.org/10.1109/ICSE.2019.00122>



# Some techniques for supporting reuse

- New ways to start code search
  - Browse API documentation to find right class for the task
  - Support searching for examples across existing codebases
  - Find the right sequence of methods to complete some task you already started
- Helping understand code examples

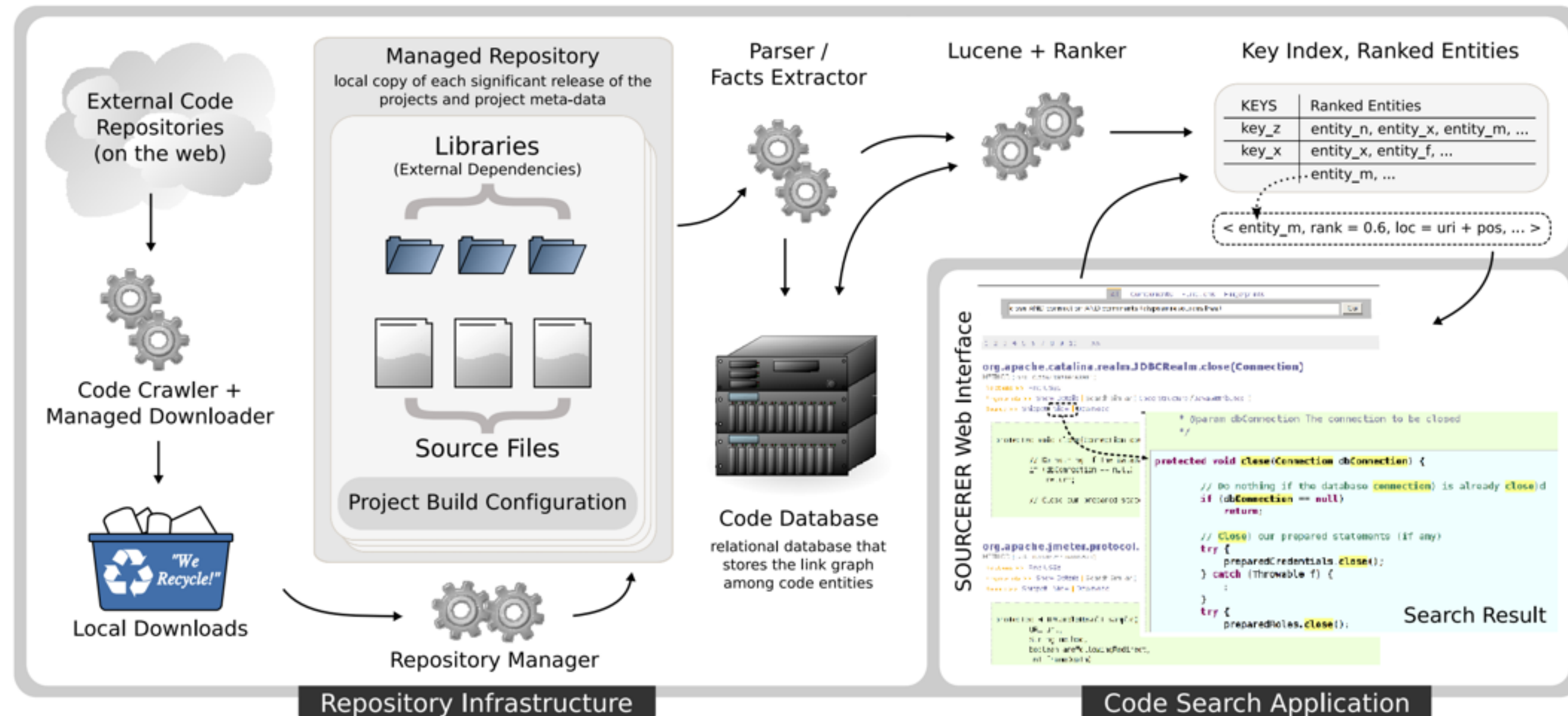
# Filtering & browsing documentation

## Apatite: A New Interface for Exploring APIs

Daniel S. Eisenberg, Jeffrey Stylos,  
and Brad A. Myers

Carnegie Mellon University

# Indexing OSS projects for code search



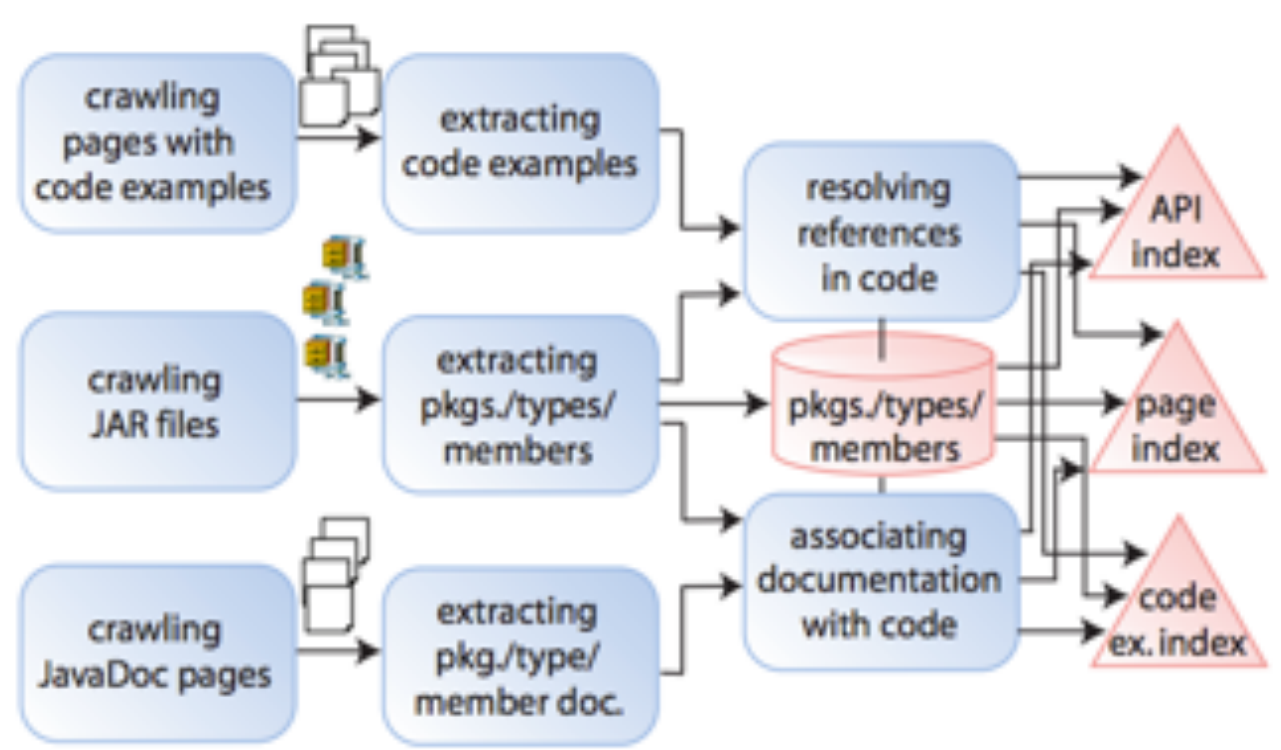
Sushil Bajracharya, Trung Ngo, Erik Linstead, Yimeng Dou, Paul Rigor, Pierre Baldi, and Cristina Lopes. Sourcerer: a search engine for open source code supporting structure-based search. In OOPSLA '06: Companion to the 21st ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, pages 681–682, New York, NY, USA, 2006. ACM Press.

# Grouping diverse search results

The screenshot shows the Assieme web search interface. At the top, there is a search bar with the text 'output acrobat' and a 'Search' button. Below the search bar, there is a table of search results. The table has columns for 'Packages', 'Types', and 'Members'. The results are grouped by package, with example counts and links to Javadoc. A filter is applied to 'com.lowagie.text.pdf'. Below the search results, there are several search results with code examples and library links. A context-sensitive menu is shown over a code example, revealing the fully qualified name 'com.lowagie.text.Phrase' and links to 'Examples', 'Javadoc', and 'Java Archive Files'. Annotations on the left side of the screenshot explain the interface features:

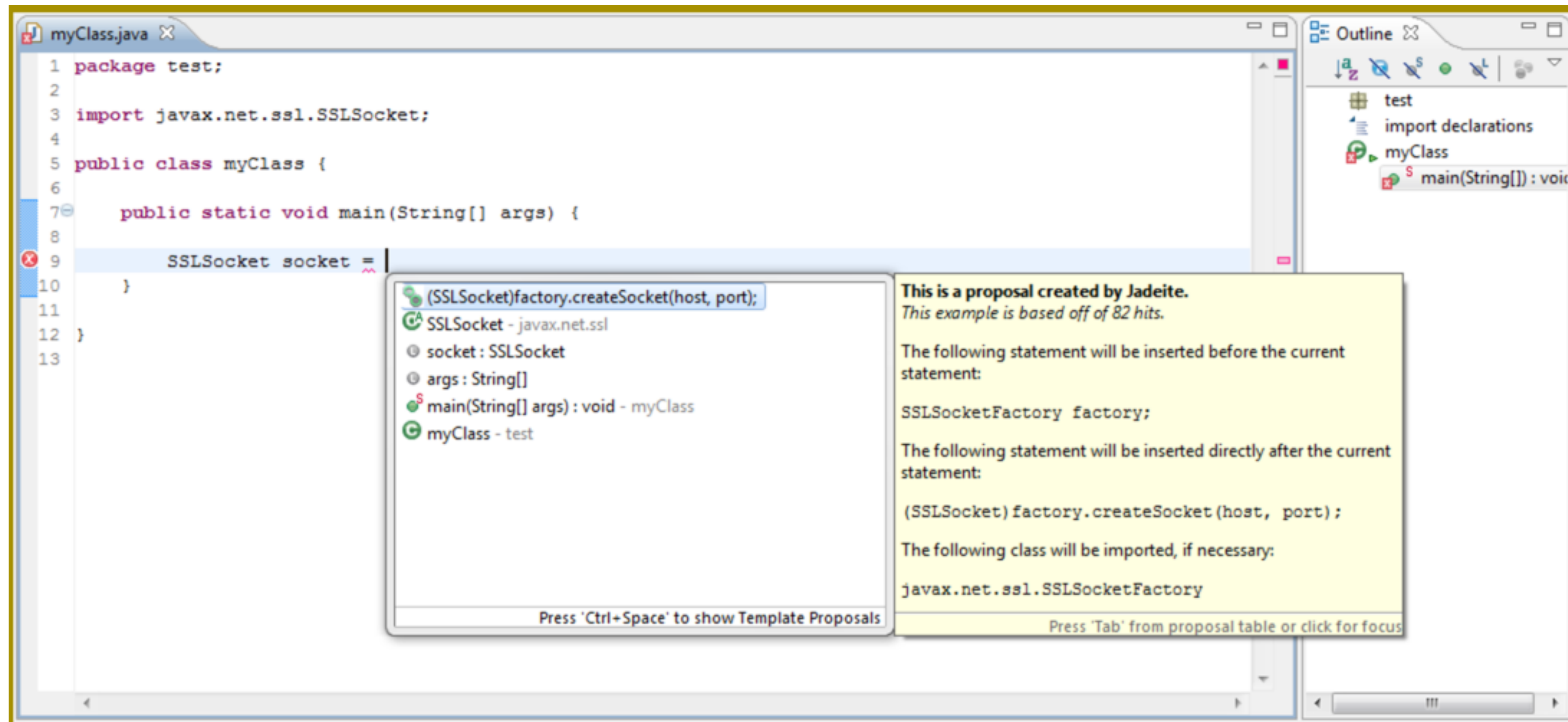
- Browse packages, types, and methods related to keywords
- Judge relevancy by example counts
- Filter pages with code examples by package, type, or member
- See and download required JAR files
- Page summaries show Java types used in code examples
- Hover over page titles to see code example previews
- Context-sensitive menus at highlighted types (see Figure 2)

Figure 2: A context-sensitive menu reveals the fully qualified names of elements appearing in Java code and provides links to additional information.



Raphael Hoffmann, James Fogarty, and Daniel S. Weld. 2007. Assieme: finding and leveraging implicit references in a web search interface for programmers. *Symposium on User interface software and technology (UIST '07)*, 13-22.

# Calcite: Offering virtual methods matching user expectations



Mathew Mooty, Andrew Faulring, Jeffrey Stylos and Brad Myers. "Calcite: Completing Code Completion for Constructors using Crowds," 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, Madrid, Spain, 21–25 September 2010. pp. 15–22.

# Searching by inputs and outputs

Name	Test Cases	
	Input	Output
Simple Tokenizer	"this is a test"	[ "this", "is", "a", "test" ]
Quote Tokenizer	"this is a test"	[ "this", "is", "a", "test" ]
	"this is a 'test with' quoted \"string types\" in it"	[ "this", "is", "a", "test with", "quoted", "string types", "in", "it" ]
Robots.txt	"http://www.cs.brown.edu/people/spr"	true
	"http://www.cnn.com/topics"	true
	"http://www.nytimes.com/college/students"	false
Log2	0	RuntimeException
	1	0
	4	2
	32	5
To Roman	13	xiii
From Roman	VIII	8
	xxvi	26
Primes	5	true
	39	false
	59	true
Perfect Numbers	6	true
	12	false
	28	true
Day of Week	"08/07/08"	"Thursday"
Easter	2008	new Date(108,2,23)



S6 Search Page - Iceweasel

File Edit View History Bookmarks Tools Help

http://conifer.cs.brown.edu:8180/S6Search/s6search.html

Most Visited Getting Started Latest Headlines

# S<sup>6</sup>

Look for:  In  Archives Using

Description:   
(keywords)

Method

Declaration:

Tests:

<input type="text" value="(17"/>	<input type="text" value="=="/>	<input type="text" value="XVII"/>	<input type="text" value="CALL"/>
<input type="text" value=""/>	<input type="text" value="=="/>	<input type="text" value=""/>	<input type="text" value="CALL"/>

**Find it!**

---

## Results:

Order By:  Format Using:

Source: [programs/roman-numerals/bio98q1.java @ http://www.olympiad.org.uk](http://www.olympiad.org.uk/programs/roman-numerals/bio98q1.java)

```
static String [] hundreds = {"", "c", "cc", "ccc", "cd", "d", "dc", "dcc", "dccc", "cm" };

static String [] tens = {"", "x", "xx", "xxx", "xl", "l", "lx", "lxx", "lxxx", "xc" };
static String [] thousands = {"", "m", "mm", "mmm" };
static String [] units = {"", "i", "ii", "iii", "iv", "v", "vi", "vii", "viii", "ix"};

public static String convert(int n)
{
    return (thousands[(n/1000)] + hundreds[(n/100)% 10] + tens[(n/10)% 10] +units[(n)% 10]).toUpperCase();
}
```

Source: [XQuisitor/saxonb8-4+Folder/net/sf/saxon/number/Numberer\\_en.java @ http://www.cafeconleche.org/xquisitor/xquisitor-1.0a5.zip](http://www.cafeconleche.org/xquisitor/xquisitor-1.0a5.zip)

# Searching by input and output *types*

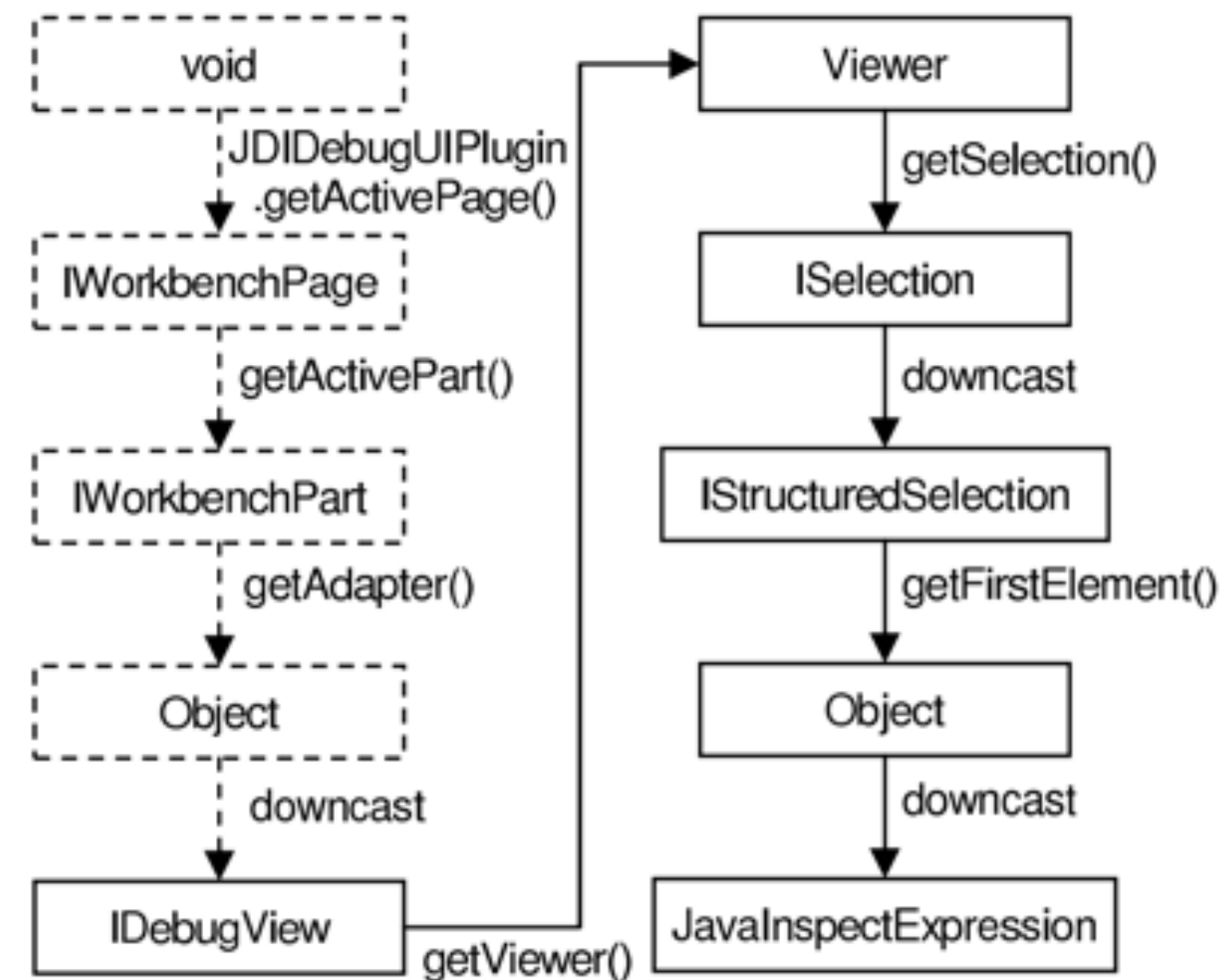
## Programming problem

Read lines from an input stream (Tester)  
 Open a named file for memory-mapped I/O (Almanac)  
 Get table widget from an Eclipse view (FAQs)  
 Get the active editor (Eclipse FAQs)  
 Retrieve canvas from scrolling viewer (Author)  
 Get window for MessageBox (Author)  
 Convert legacy class (Author)

$\tau_{in}$   
 InputStream  
 String  
 TableViewer  
 IWorkbench  
 ScrollingGraphicalViewer  
 KeyEvent  
 Enumeration

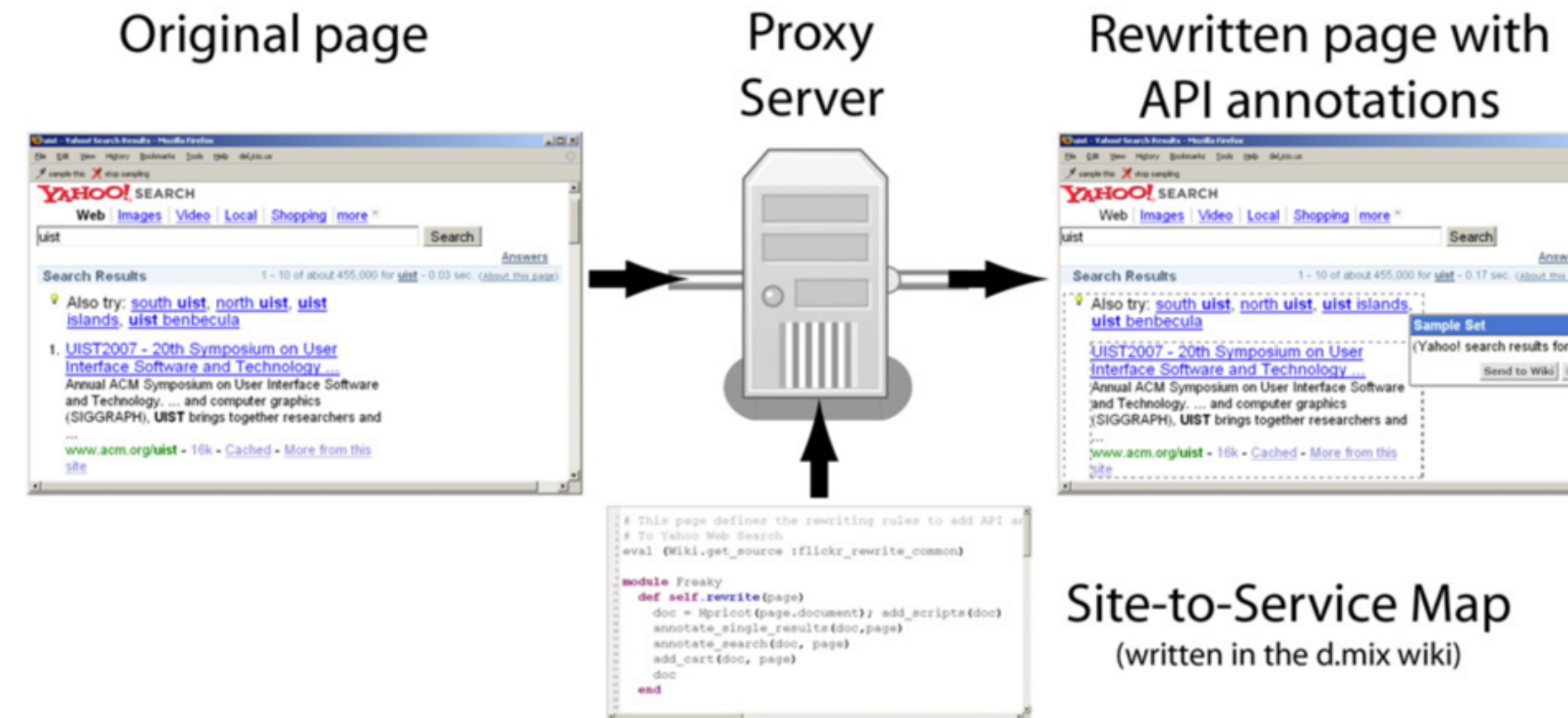
$\tau_{out}$   
 BufferedReader  
 MappedByteBuffer  
 Table  
 IEditorPart  
 FigureCanvas  
 Shell  
 Iterator

Mine *Jungoloids* describing paths by which types can be converted



David Mandelin, Lin Xu, Rastislav Bodík, and Doug Kimelman. 2005. Jungloid mining: helping to navigate the API jungle. *Conference on Programming language design and implementation (PLDI '05)*, 48-61.

# Searching by output

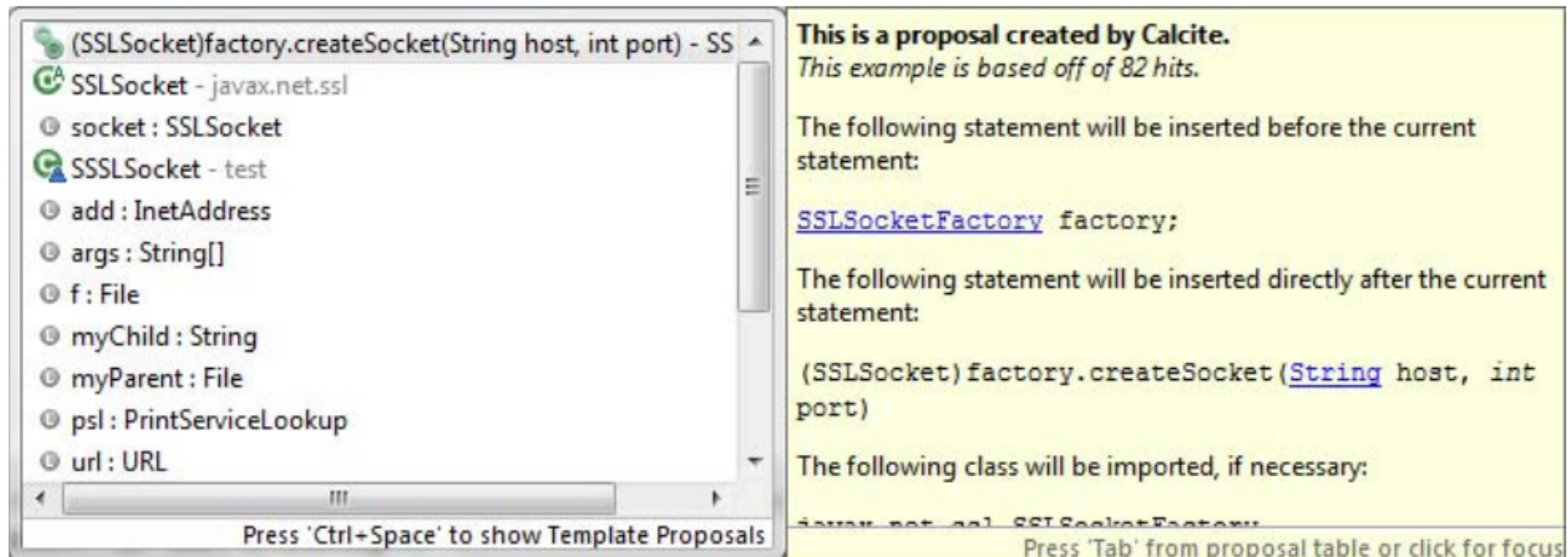


**Figure 1.** With d.mix, users browse web sites through a proxy that marks API-accessible content. Users select marked elements they wish to copy. Through a site-to-service map, d.mix composes web service calls that yield results corresponding to the user's selection. This code is copied to the d.mix wiki for editing and hosting.

<http://dl.acm.org/citation.cfm?doid=1294211.1294254>

# Searching for instantiation snippets

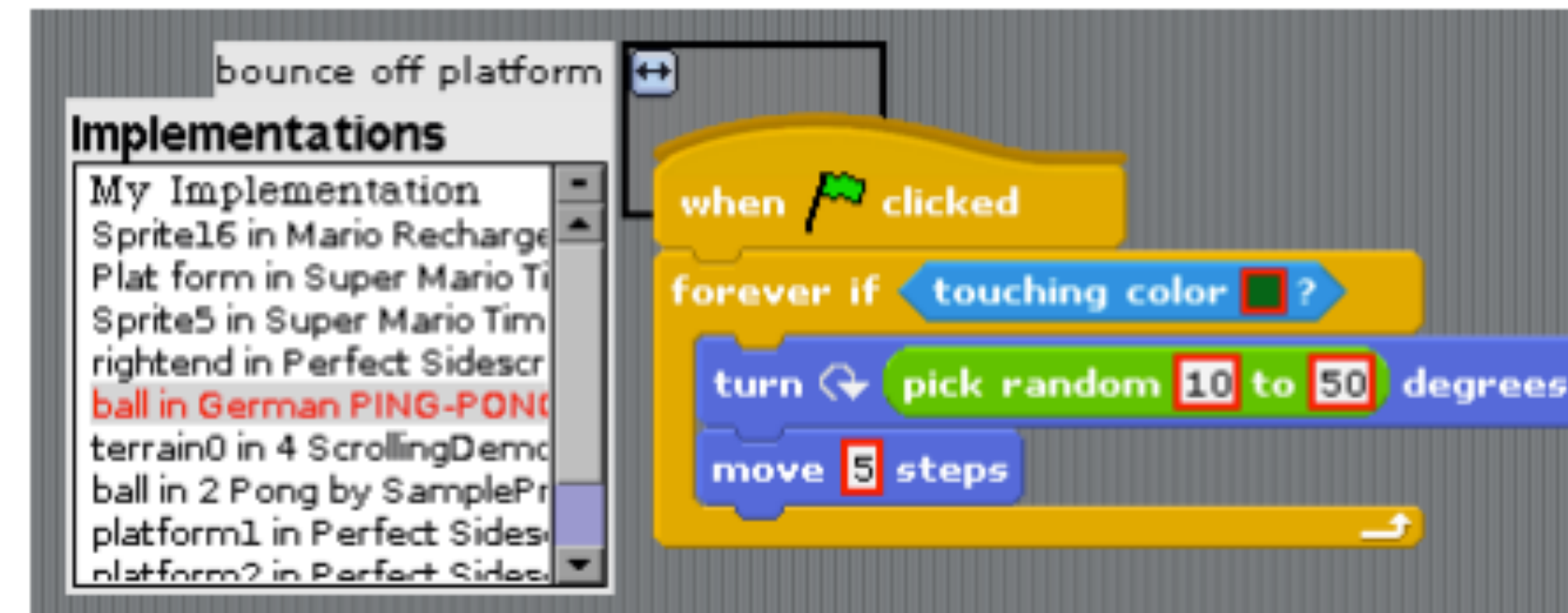
- Classes are often created through factories rather than constructors, making construction snippets harder to find
- Integrate construction snippet search into autocomplete



M. Mooty, A. Faulring, J. Stylos and B. A. Myers, "Calcite: Completing Code Completion for Constructors Using Crowds," *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, Leganes, 2010, pp. 15-22.

# Labeling snippets with keywords

- Problem: how do you ensure that there's high quality labels explaining the intention of code snippets?
- Idea: enable search from keywords to code **and** from code to keywords
- Log associations to support future queries



**Figure 2.** Given a purpose statement, the Zone sidebar (left) shows code that might fulfill it. Selecting an implementation from the list on the left shows its code on the right. As a simulation of future functionality, red boxes surround values that vary among otherwise similar code, highlighting what might need to be changed.



**Figure 3.** The Zone sidebar can suggest possible purpose statements for a code fragment (simulated for this illustration).

Kenneth C. Arnold and Henry Lieberman. 2010. Managing ambiguity in programming by finding unambiguous examples. *Conference on Object oriented programming systems languages and applications*, 877-884.

# Adapt snippets

## SnipMatch Demonstration

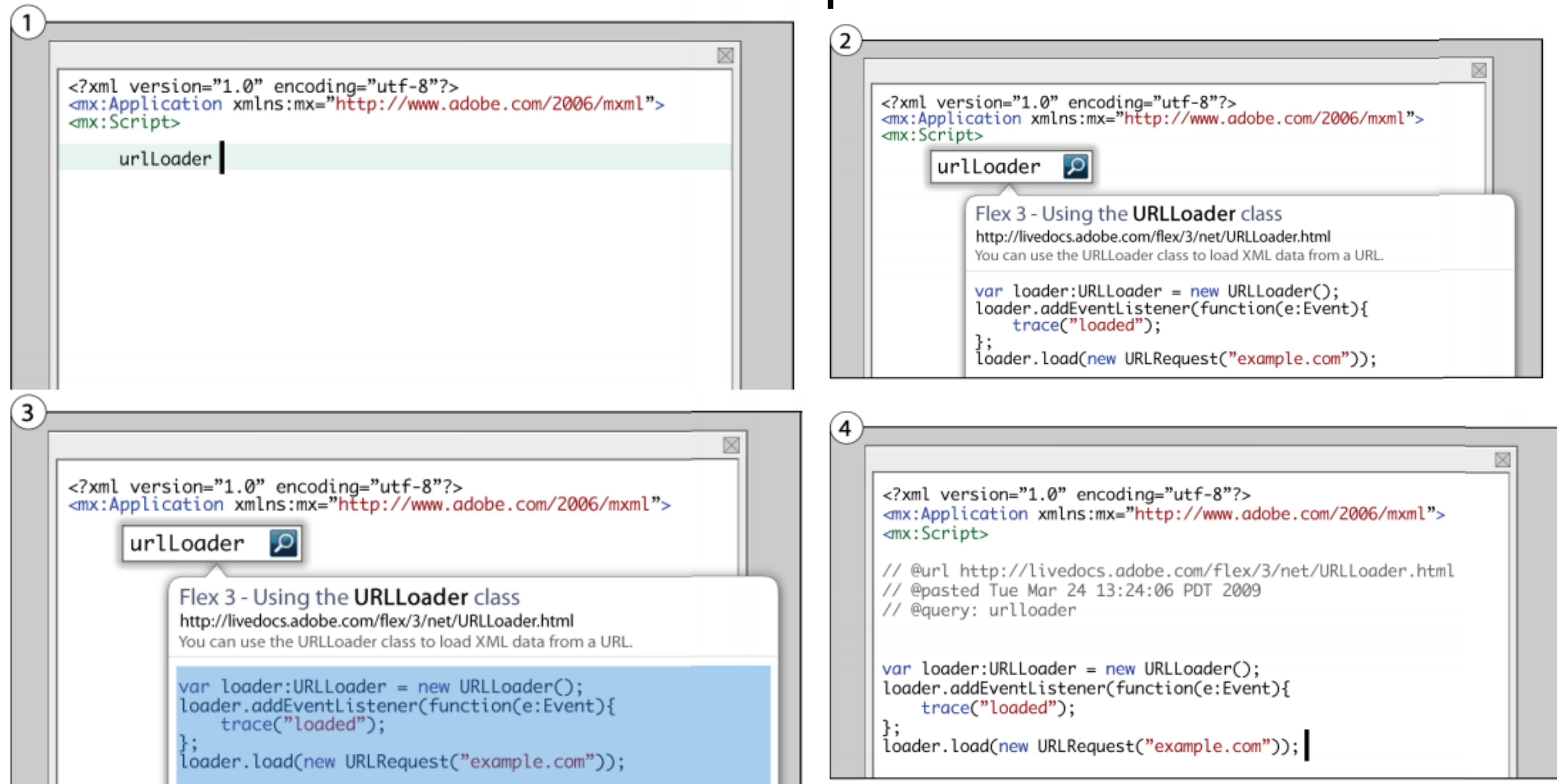
**Doug Wightman<sup>1</sup>, Zi Ye<sup>1</sup>, Joel Brandt<sup>2</sup>, Roel Vertegaal<sup>1</sup>**

<sup>1</sup>Human Media Lab, Queen's University  
Kingston, ON, K7L 3N6, Canada  
{wightman, zi, roel}@cs.queensu.ca

<sup>2</sup>Advanced Technology Labs, Adobe  
San Francisco, CA 94103  
joel.brandt@adobe.com

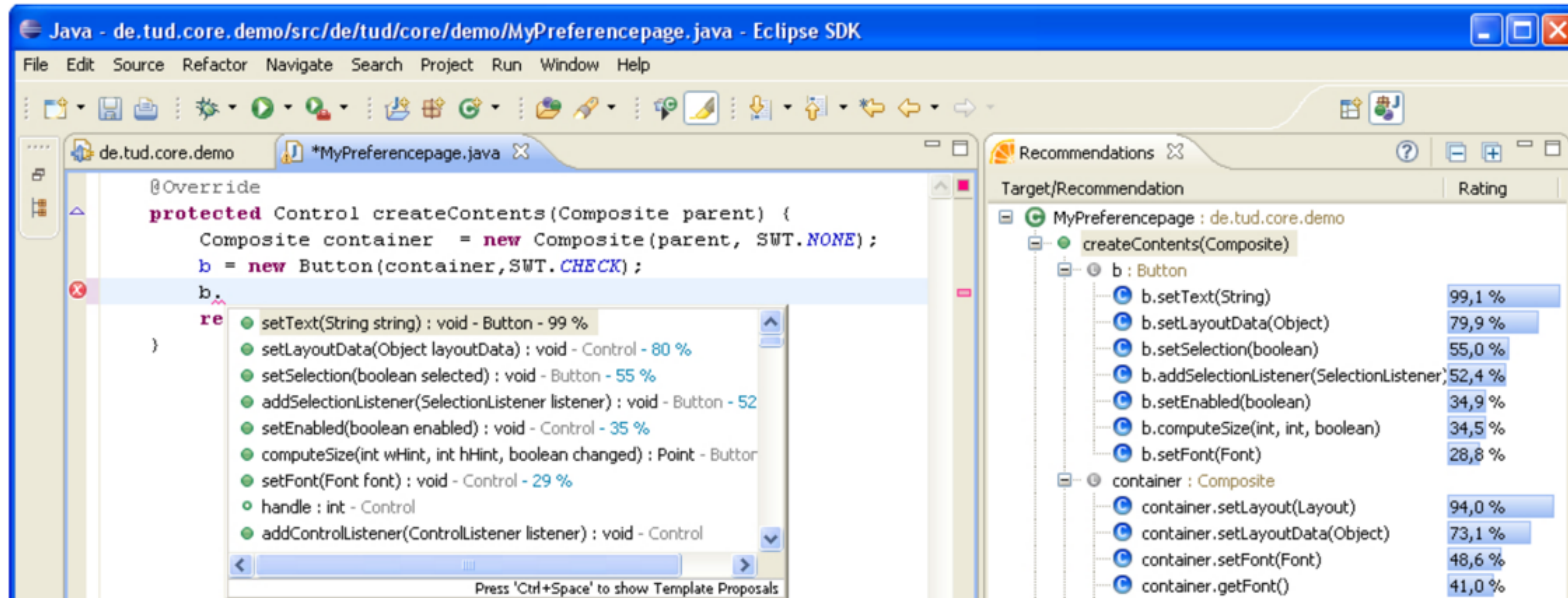
Doug Wightman, Zi Ye, Joel Brandt, and Roel Vertegaal. 2012. SnipMatch: using source code context to enhance snippet retrieval and parameterization. *Symposium on User interface software and technology*, 219-228.

# Integrating code search into autocomplete



Commercial tool, extension built for Adobe Flex Builder

# Offering autocomplete suggestions based on most frequent completions



Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric programming: integrating web search into the development environment. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). ACM, New York, NY, USA, 513-522. DOI: <https://doi.org/10.1145/1753326.1753402>



Helping understand  
code examples

```
video_data['speakers'] = [a.get_text() for a in soup.select('div#sidebar a[href^=/speaker]')]
video_data['youtube_url'] = soup.select('div#sidebar a[href^=/speaker]')[0].get('href')
```

A few things to note about this function:

- The URLs returned from the scraping
- The session title is obtained from the `<h1>` because the `select()` call returns a list
- The speaker names and YouTube links

Now all that remains is to scrape the video URL. This is very simple to write as a continuation of the previous code. We can also scrape the likes and dislikes.

You found a CSS selector.  
The selector `'div#sidebar a[href^=/speaker]'` chooses links with URLs starting with `/speaker` from a container with the ID `'sidebar'`.  
*If you haven't seen them before*, selectors pick sections of HTML pages by their names or properties. Once you've 'grabbed' elements with a selector, you can manipulate them, like changing their appearance or text.  
Here's an example of what this selector will find:

```
<div id="sidebar">
  <a href="/speaker">
  </a>
</div>
```

```
def get_video_data(video_page_url):
```

(a) A micro-explanation of a CSS selector with an automatically generated natural language explanation and demonstration of an HTML element it matches.

Use `m` suffix for megabytes (`--limit-rate=1m`). The above command will limit the retrieval rate to 50KB/s. It is also possible to specify disk quota for automatic retrievals to avoid disk DoS attack. The following command will be aborted when the quota is (100MB+) exceeded.

```
$ wget -cb -o /tmp/download.log -i /tmp/download.txt --quota=100m
```

You found a `wget` command.

`wget` is a Terminal command you run to download a page from the Internet. Here, it downloads content from URLs from the file `'/tmp/download.txt'`.

It uses these options:

- `--continue (-c)`: resume getting a partially-downloaded file.
- `--background (-b)`: go to background after startup.
- `--output-file (-o)`: log messages to `/tmp/download.log`.
- `--input-file (-i)`: download URLs found in local or external `/tmp/download.txt`.
- `--quota (-Q)`: set retrieval quota to 100m.

(b) A micro-explanation describing the high-level intent and low-level argument values of a `wget` command.

## Stack Overflow

```
if (value == null) {
```

```
    return def;
```

```
} else {
```

```
    try {
```

```
        return Integer.parseInt(value.trim());
```

```
        ...
```

```
    } catch (NumberFormatException nf) {
```

```
        ...
```

```

15 String QUERY = "SELECT id, title, year, num_pages FROM table WHERE
16 int COLUMN_INDEX_ID = 0;
17 int COLUMN_INDEX_TITLE = 1;
18 int COLUMN_INDEX_YEAR = 2;
19 int COLUMN_INDEX_NUM_PAGES = 3;
20 boolean DEBUG = true;
21
22 Database database = new Database("lou", "PA$$WORD", "https://acme-
23 Cursor cursor = database.cursor();
24 Booklist booklist = new Booklist();
25 List titles = new ArrayList();
26
27 try {
28
29     cursor.execute(QUERY);
30     boolean finished = false;
31
32     if (cursor.rowCount() > 0) {
33
34         int rowNumber = 0;
35         while (finished == false) {
36
37             int rowCount = cursor.rowCount();
38
39             for (int i = 0; i < Math.min(rowCount, maxBooks); ++i)
40
41                 cursor.fetchone();
42                 int id = cursor.getInt(COLUMN_INDEX_ID);
43                 String title = cursor.getString(COLUMN_INDEX_TITLE);
44                 int year = cursor.getInt(COLUMN_INDEX_YEAR);
45                 int num_pages = cursor.getInt(COLUMN_INDEX_NUM_PAGES);
46                 Book book = new Book(id, title, year, num_pages);
47
48                 if (title != null) {
49                     titles.add(title);
50                 }
51                 if (id != -1) {
52                     boolean bestseller = isBestseller(book.getId())

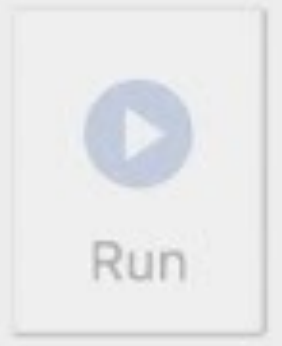
```

```

1 public class ExtractedExample {
2
3     public static void main(String[] args) {
4
5         Cursor cursor = database.cursor();
6         List titles = new ArrayList();
7         try {
8             cursor.execute(QUERY);
9             cursor.fetchone();
10            String title = cursor.getString(COLUMN_INDEX_TITLE);
11            titles.add(title);
12        } catch (Connecti
13        }
14
15    }
16
17 }

```

← Do you want any of those uses of "cursor"? **No**



# Tech Talks