

Semantic Web Tools

Based on the slides by Dipsy Kapoor

Outline

Semantic Web Tools

- Piggy Bank/Solvent
- Jena/ARQ

Piggy Bank



- “Semantic Web Browser” by SIMILE project at MIT
- Available as a Firefox plug-in at <http://simile.mit.edu/piggy-bank/>
- Can ingest RDF data directly from websites that contain meta information, and can execute screen-scrapers to extract content from normal websites and then convert that data to RDF

Warning: Piggy Bank works with Firefox 2.0.x only.

Piggy Bank

- Features of the browser:
 - **Users can find, collect, annotate RDF**
 - **Search and faceted browse of local RDF**
 - Export data in RDF/N3

But this tool is **REALLY USELESS**

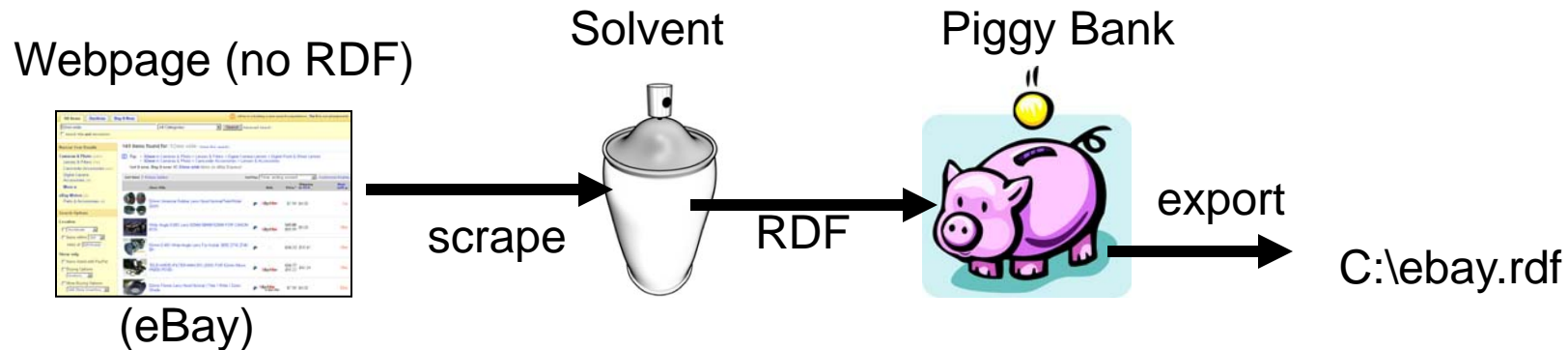
when websites have no RDF describing objects in them!

Solvent

<http://simile.mit.edu/wiki/Solvent>



- Screen-Scraper (javascript-based) developed by the SIMILE project
- An extension of Piggy Bank
- Extracts data and exports it as RDF/n3



Piggy Bank and Solvent Demo

Scrape data from <http://boston.craigslist.org/aap/> and export to RDF

The data

[boston craigslist](#) > [all apartments](#)

[\[help \]](#)

[all boston](#) [boston/camb/brook](#) [northwest/merrimack](#) [metro west](#) [north shore](#) [south shore](#)

search for: in: [all apartments](#) only search titles
rent: min max 0+ BR cats dogs has image

[Mon, 25 Feb 21:04:24]

[[stating a discriminatory preference in a housing post is illegal](#)] [[housing forum](#)] [[success story?](#)]

[\[AVOIDING SCAMS & FRAUD \]](#) | [\[PERSONAL SAFETY TIPS \]](#)

Mon Feb 25

[\\$1360 / 1br - 1.5 bed Porter Square All Util included+off street parking - \(Porter Sq Somerville\)](#) <<[apts broker fee](#)

[\\$4500 / 3br - ► Fully Furnished 3 Bed HOUSE◀ - \(Cambridge/Harvard Square\)](#) [pic](#) [img](#) <<[apts broker fee](#)

[\\$2900 / 2br - ++DOGS OKAY UP TO 35LBS + NO FEE LUXURY APARTMENT ++ - \(SEAPORT, WATERFRONT\)](#) [img](#) <<[apts broker no fee](#)

[\\$1295 / 2br - Large. New kitchen. Hardwood floors. Perfect location. - \(Allston / Pacakard's Corner\)](#) [img](#) <<[apts broker no fee](#)

[\\$1495 / 2br - AWESOME UNIT IN AWESOME BUILDING!! HEAT&HTO WATER INCL. SEPT 1ST - \(BOSTON/BU WEST/ALLSTON/AID\)](#) [pic](#) [img](#)
<<[apts broker no fee](#)

[\\$1499 / 2br - ◆MEDFORD - WELLINGTON T! LUXURY LIVING! BIG 1BR, NO FEE!! SEE PICS! - \(Medford\)](#) [img](#) <<[apts broker no fee](#)

[\\$2025 / 1br - RENOVATED 1 BEDROOM APARTMENT - \(BACK BAY\)](#) [img](#) <<[apts broker no fee](#)

Objective: turn this data to RDF file

Activate Solvent

The image shows a screenshot of the Solvent workspace. At the top left, there is a small toolbar with icons for a globe, a magnifying glass, and a refresh button. A large black arrow points from this toolbar to the workspace window. The workspace window is divided into two main sections. The top section is a browser window titled "boston all apartments classifieds - craigslist - Mozilla Firefox". The browser's address bar shows "http://boston.craigslist.org/aap/". The page content displays a list of apartment listings under the heading "Mon Feb 25". The listings include details such as price, number of bedrooms, and location. The bottom section of the workspace window is a "Solvent Workspace" window. It has a toolbar with "URLs", "Code: 1", "Insert", "Run", and "Stop" buttons. Below the toolbar, there are two panes: "Item" and "Variable". The "Item" pane is currently empty. The "Variable" pane is also empty. The status bar at the bottom of the workspace window shows "Done".

URL/code window

Solvent Workspace

Capture/result window

Select rows to scrape

The screenshot shows a Mozilla Firefox browser window displaying a Craigslist page for Boston apartments. The page lists several apartment listings, each highlighted in yellow. A text box is overlaid on the page, stating: "Just crop the first row, then Solvent will do the rest (based on DOM structure)". Below the browser window, the Solvent tool interface is visible, showing a list of captured items. The list includes items 1 through 9, with item 9 highlighted. A text box labeled "Captured items" is overlaid on the Solvent interface.

Mon Feb 25

[\\$1360 / 1br - 1.5 bed Porter Square All Util included+off street parking - \(Porter Sq Somerville\) <<apts broker fee](#)

[\\$4500 / 3br - Fully Furnished 3 Bed HOUSE - \(Cambridge/Harvard Square\) pic img <<apts broker fee](#)

[\\$2900 / 2br - ++DOGS OKAY UP TO 35LBS + NO FEE LUXURY APARTMENT ++ - \(SEAPORT, WATEFRONT\) img <<apts broker no fee](#)

[\\$1295 / 2br - Large. New kitchen. Hardwood floors. Perfect location. - \(Allston / Pacakard's Corner\) img <<apts broker no fee](#)

[\\$1495 / 2br - AWFUL WEST/ALLSTON/ALLSTON/BU](#)

[\\$1499 / 2br - +ME \(rd\) img <<apts broker no fee](#)

Just crop the first row, then Solvent will do the rest (based on DOM structure)

Done

URLs Code: 1 Insert Run Stop

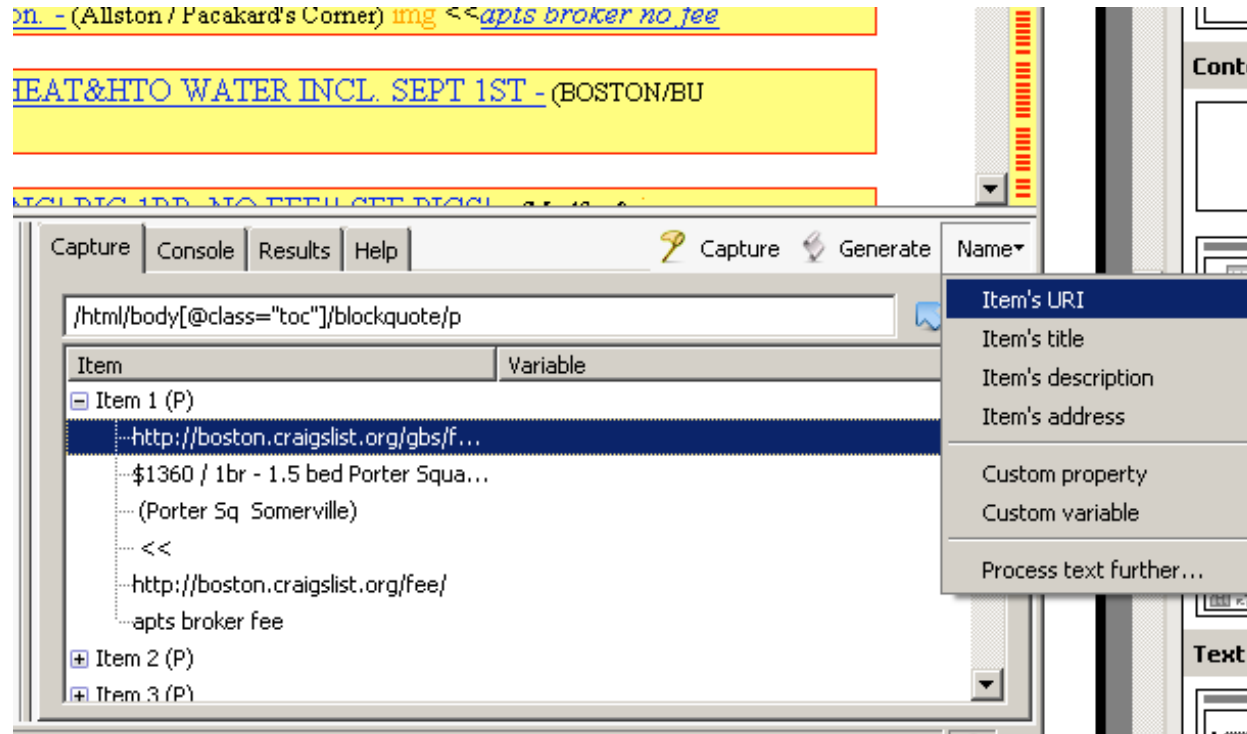
Capture Console Results Help Capture Generate Name

/html/body[@class="toc"]/blockquote/p

Item	Variable
Item 1 (P)	
Item 2 (P)	
Item 3 (P)	
Item 4 (P)	
Item 5 (P)	
Item 6 (P)	
Item 7 (P)	
Item 8 (P)	
Item 9 (P)	

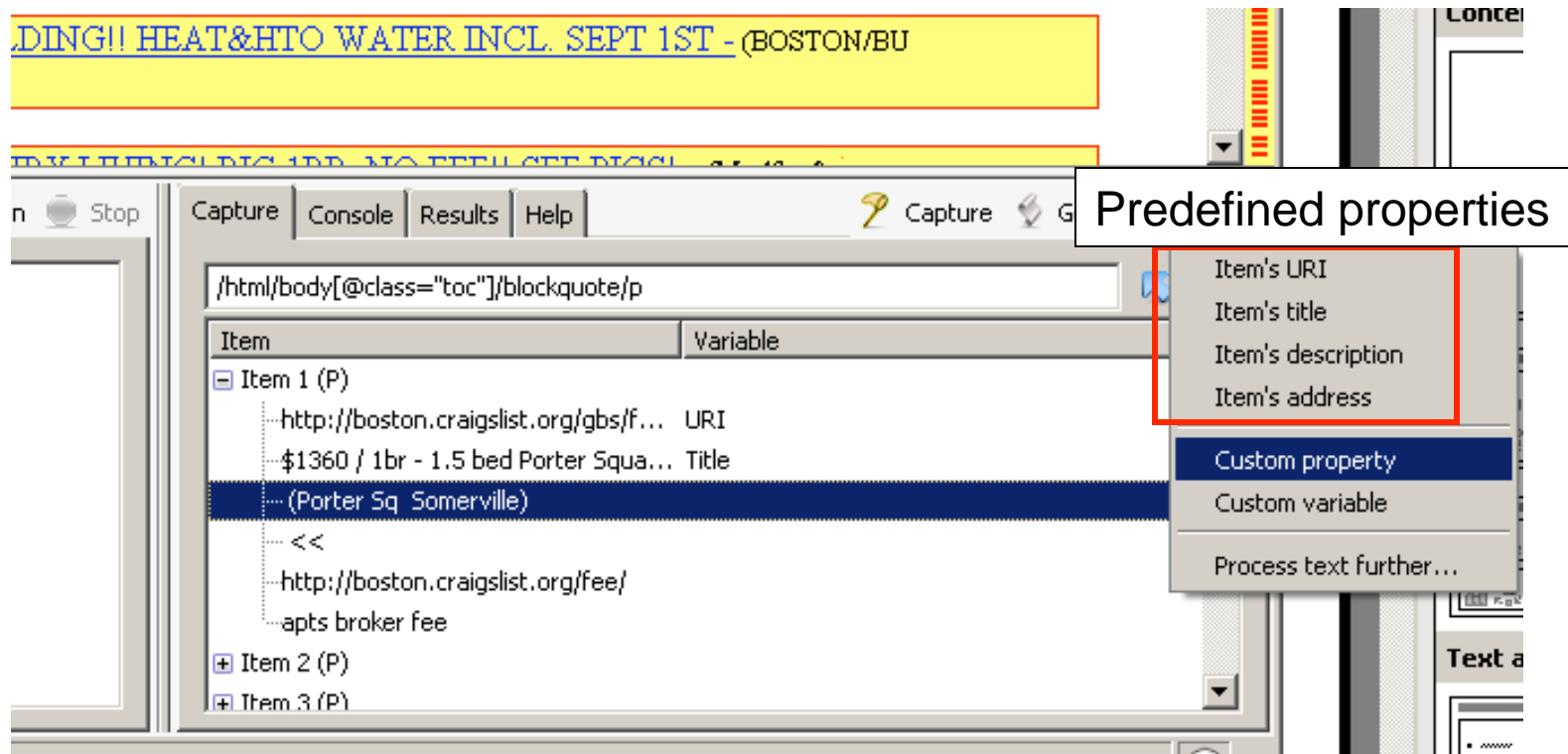
Captured items

Name each extracted attribute



Every item must have URI in URI format! (if not, Solvent will not scrape anything)

Custom properties



You can define custom properties.

Remember the name of a property should be in URI form

e.g. <http://boston.craiglist.org/app#area>

Generate the scraper

Mon Feb 25

[\\$1360 / 1br - 1.5 bed Porter Square All Util included+off street parking - \(Porter Sq Somerville\)](#) <<[apts broker fee](#)

[\\$4500 / 3br - Fully Furnished 3 Bed HOUSE - \(Cambridge/Harvard Square\)](#) [pic](#) [img](#) <<[apts broker fee](#)

[\\$2900 / 2br - ++DOGS OKAY UP TO 35LBS + NO FEE LUXURY APARTMENT ++ - \(SEAPORT, WATEFRONT\)](#) [img](#)
<<[apts broker no fee](#)

[\\$1295 / 2br - Large. New kitchen. Hardwood floors. Perfect location. - \(Allston / Pacakard's Corner\)](#) [img](#) <<[apts broker no fee](#)

[\\$1495 / 2br - AWESOME UNIT IN AWESOME BUILDING! HEAT&HOT WATER INCL SEPT 1ST - \(BOSTON/BU
WEST/ALLSTON/AID\)](#) [pic](#) [img](#)

Test if the code can scrape the page

The screenshot shows a web browser window displaying a list of apartment listings. Below the listings, there is a JavaScript console and a scraper tool interface. The JavaScript console contains the following code:

```
var rdf = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#';  
var dc = 'http://purl.org/dc/elements/1.1/';  
  
var namespace = document.documentElement.namespaceURI;  
var nsResolver = namespace ? function(prefix) {  
    return (prefix === 'x') ? namespace : null;  
} : null;  
  
var getNode = function(document, contextNode, xpath, nsRes  
    return document.evaluate(xpath, contextNode, nsResolver,  
    ]  
  
var cleanstring = function(s) {
```

The scraper tool interface shows a table of results for the XPath query `/html/body[@class="toc"]/blockquote/p`. The table has two columns: "Item" and "Variable". The results are as follows:

Item	Variable
Item 1 (P)	
http://boston.craigslist.org/gbs/f...	URI
\$1360 / 1br - 1.5 bed Porter Squa...	Title
(Porter Sq Somerville)	http://boston.craiglist.org/aap#area
<<	
http://boston.craigslist.org/fee/	
apts broker fee	http://boston.craiglist.org/aap#seller
Item 2 (P)	
Item 3 (P)	

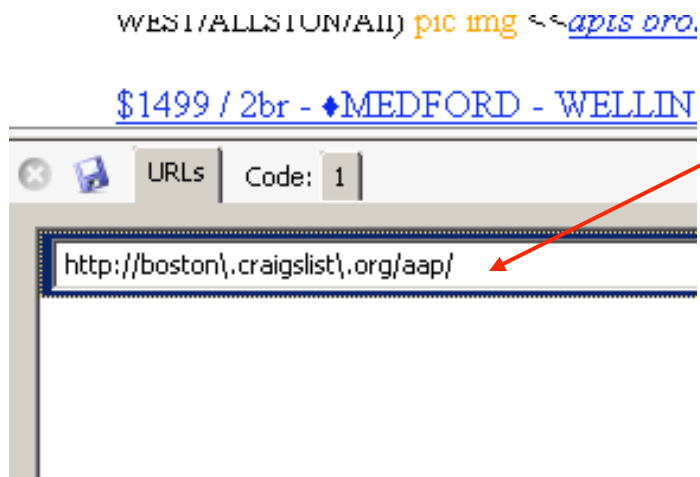
A red arrow points from the "Generate" button in the scraper tool to the JavaScript code in the console.

Javascript code for scrapping the page

Define type (row type) & URL pattern

```
data.addStatement(uri, rdf + 'type', 'http://simile.mit.edu/ns#Unknown',  
false); // Use your own type here
```

Change to the type of the row (e.g. Apartment)



Define URL pattern – so Piggy Bank will use this scraper when the present URL matches to the scraper' URL pattern.

Save your scraper & install it to Piggy Bank

Save Scraper

Scraper's Info | Author's Info | Files and URLs

Title
Craiglist Scraper

URI
http://me.anon.com#craiglistscraper

You need to enter a URI that will be used to identify your scraper. If you own a Web domain (e.g., http://somedomain.com/), you can coin a URI for your scraper as something like this: http://somedomain.com/scrapers/nameOfScraper.

Description

Browse to the metadata file after closing this dialog box

OK Cancel

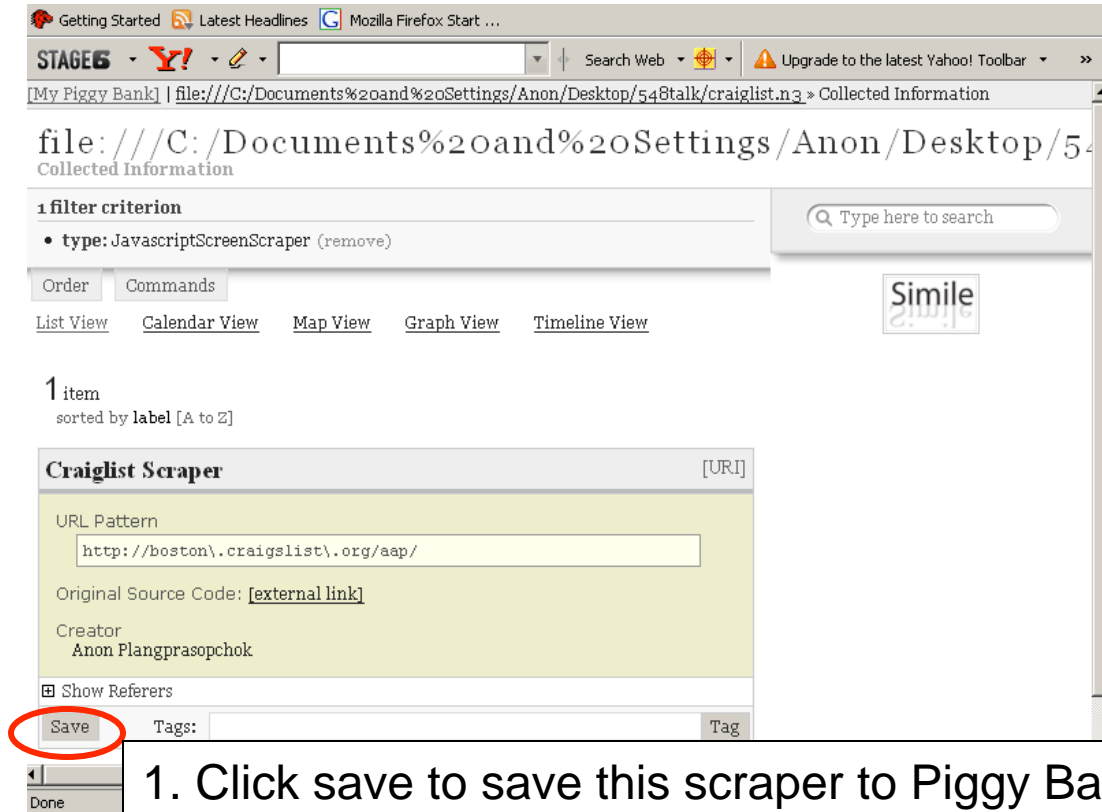
craiglist.js

craiglist.n3

Two files (scraper's code & metadata) are generated

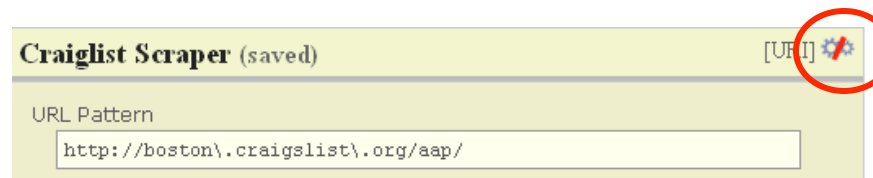
Then open the .n3 file (craiglist.n3) with Firefox.
Piggy Bank will load this scraper on its environment
(next slide).

Install the scraper to Piggy Bank



The screenshot shows the Piggy Bank interface in a Mozilla Firefox browser window. The address bar shows a file path: `file:///C:/Documents%20and%20Settings/Anon/Desktop/548talk/craigslist.n3`. The page title is "My Piggy Bank | Collected Information". The main content area shows a "filter criterion" section with a search bar and a list of items. One item is displayed, titled "Craiglist Scraper" with a "[URI]" link. The scraper details include a URL pattern of `http://boston\.craigslist\.org/aap/`, an original source code link, and a creator name "Anon Plangprasopchok". At the bottom of the scraper details, there is a "Show Referers" checkbox and a "Save" button, which is circled in red. A text box below the screenshot contains the instruction: "1. Click save to save this scraper to Piggy Bank repository".

2. Activate the scraper



The screenshot shows the "Craiglist Scraper (saved)" entry in the Piggy Bank interface. The entry is highlighted in a light green color. The URL pattern is `http://boston\.craigslist\.org/aap/`. A settings icon (a gear) is circled in red in the top right corner of the entry. A text box above the screenshot contains the instruction: "2. Activate the scraper".

Scrape the Data!

1. Visit boston.craigslist.org/aap again then click at the coin icon



You have to restart your browser to have your new scraper works properly!!

Scraping process might take time!
(don't panic)

[My Piggy Bank] | [boston all apartments classifieds - craigslist](#) » Collected Information

boston all apartments classifieds - craigslist

Collected Information

1 filter criterion

- type: Unknown (remove)

Order Commands

List View [Calendar View](#) [Map View](#) [Graph View](#) [Timeline View](#)

100 items
sorted by title [A to Z]

\$1(50) \$2(38) \$3(9) \$4(1) \$6(1) \$7(1)

« previous 1 2 3 4 5 6 7 8 9 10 next »

\$1000 / 1br - [1324!] - Near The TTTTTTTTTTTT! Big 1 Bedroom - [URI]

generatedBy	craigslistscraper
originTitle	boston all apartments classifieds - craigslist
originURL	http://boston.craigslist.org/aap/
seller	apts broker fee
title	\$1000 / 1br - [1324!] - Near The TTTTTTTTTTTT! Big 1 Bedroom -
type	Unknown

Show Referers

Save Tags: Tag

\$1000 / 2br -- 2 Bedroom \$1000 We Have Many Call for Info - [URI]

Search: Type here to search

▼ seller [click to expand]

▼ title [click to expand]

▼ area [click to expand]

Simile

Export the data to RDF

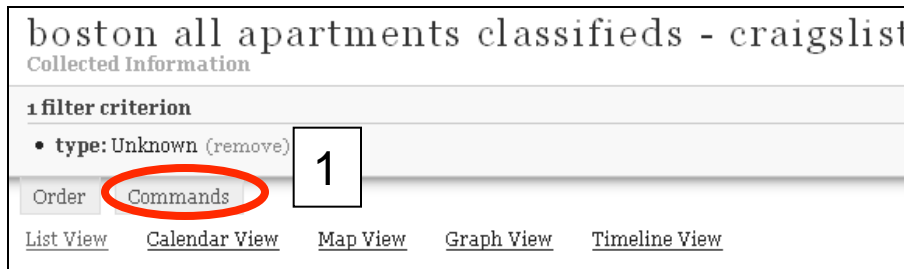
boston all apartments classifieds - craigslist
Collected Information

1 filter criterion

- type: Unknown (remove)

Order **Commands** 1

List View Calendar View Map View Graph View Timeline View



1 filter criterion

- type: Unknown (remove)

Order 2

Save All **Export** Tags:

Commands



Then save the RDF to somewhere on your machine

Jena

- A Semantic Web Framework that includes:
 - A RDF API
 - Reading and writing RDF in RDF/XML, N3 and N-Triples
 - An OWL API
 - In-memory and persistent storage
 - SPARQL query engine
- Documentation available at <http://jena.sourceforge.net/documentation.html>

ARQ

- ARQ is Jena's implementation of SPARQL.
- ARQ documentation is available at: <http://jena.sourceforge.net/ARQ/documentation.html>
- Can write queries using the Jena Java API or can write them in text files and execute them using ARQ's command line utility.

SPARQL Example

```
PREFIX SM: <http://simile.mit.edu/2005/05/ontologies/location#>
```

```
PREFIX DC: <http://purl.org/dc/elements/1.1/>
```

```
SELECT DISTINCT ?title ?address
```

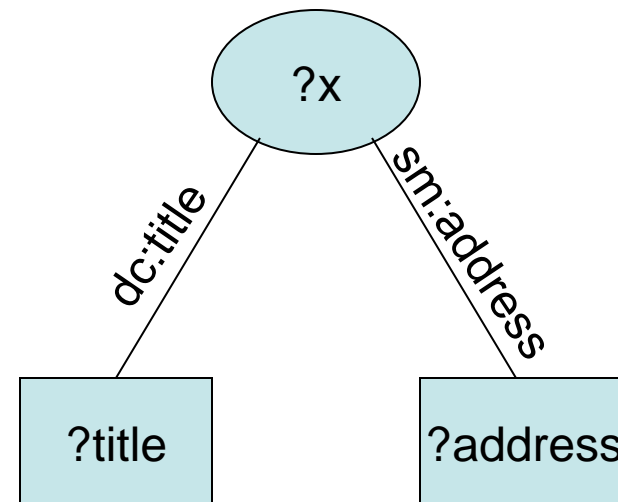
```
FROM <sources/starbucks.n3>
```

```
WHERE {
```

```
  ?x DC:title ?title .
```

```
  ?x SM:address ?address .
```

```
}
```



SPARQL Example

```
PREFIX SM: <http://simile.mit.edu/2005/05/ontologies/  
location#>
```

```
PREFIX DC: <http://purl.org/dc/elements/1.1/>
```

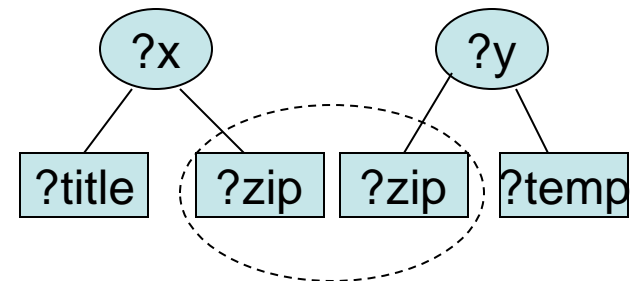
```
SELECT DISTINCT ?title ?address  
FROM <sources/starbucks.n3>  
WHERE {  
    ?x DC:title ?title .  
    ?x SM:address ?address .  
    FILTER (regex(?address, "Figueroa", "i")) .  
}
```

regex is a X-Path function. Other functions available at this tutorial -
<http://jena.sourceforge.net/ARQ/Tutorial/>

Querying multiple sources, Using simple JOIN

```
PREFIX SM: <http://simile.mit.edu/2005/05/ontologies/location#>
PREFIX DC: <http://purl.org/dc/elements/1.1/>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
```

```
SELECT DISTINCT ?title ?zip ?temperature
FROM NAMED <sources/starbucks.n3>
FROM NAMED <sources/weather.n3>
WHERE {
  GRAPH <sources/starbucks.n3> {
    ?x DC:title ?title .
    ?x DC:zipcode ?zip .
  } .
  GRAPH <sources/weather.n3> {
    ?y DC:zipcode ?zip .
    ?y DC:temperature ?temperature .
  } .
}
```



More examples (might be useful for your hw):

<http://www.ibm.com/developerworks/xml/library/j-sparql/>

Useful Links

- Piggy Bank – <http://simile.mit.edu/piggy-bank/>
- Solvent - <http://simile.mit.edu/solvent/>
- Jena Documentation - <http://jena.sourceforge.net/documentation.html>
- ARQ Documentation - <http://jena.sourceforge.net/ARQ/documentation.html>
- Article by IBM on ARQ - <http://www-128.ibm.com/developerworks/xml/library/j-sparql/>
- Portable firefox (in case you don't want to remove your firefox 3.0) - http://portableapps.com/apps/internet/firefox_portable