Microdata and schema.org

Basics

- Microdata is a simple semantic markup scheme that's an alternative to RDFa
- Developed by WHATWG* and supported by major search companies (Google, Microsoft, Yahoo, Yandex)
- Like RDFa, it uses HTML tag attributes to host metadata
- It can also be expressed as JSON-LD
- Vocabularies are controlled and hosted at schema.org

* Web Hypertext Application Technology Working Group

Microdata

- The microdata effort has two parts:
  - A markup scheme
  - A set of vocabularies/ontologies
- The markup is similar to RDFa in providing ways to identify subjects, types, properties & objects

  Also a standard way to encode Microdata as RDFa
- Sanctioned vocabularies at schema.org and include a small number of very useful ones: people, movies, events, recipes, etc.

An example

```
<div>
 <h1>Avatar</h1>
 <span>Director: James Cameron (born 1954) </span>
 <span>Science fiction</span>
 <a href=”avatar-trailer.html">Trailer</a>
</div>
```

An example: itemscope

- An *itemscope* attribute identifies a content *subtree* that is the subject about which we want to say something

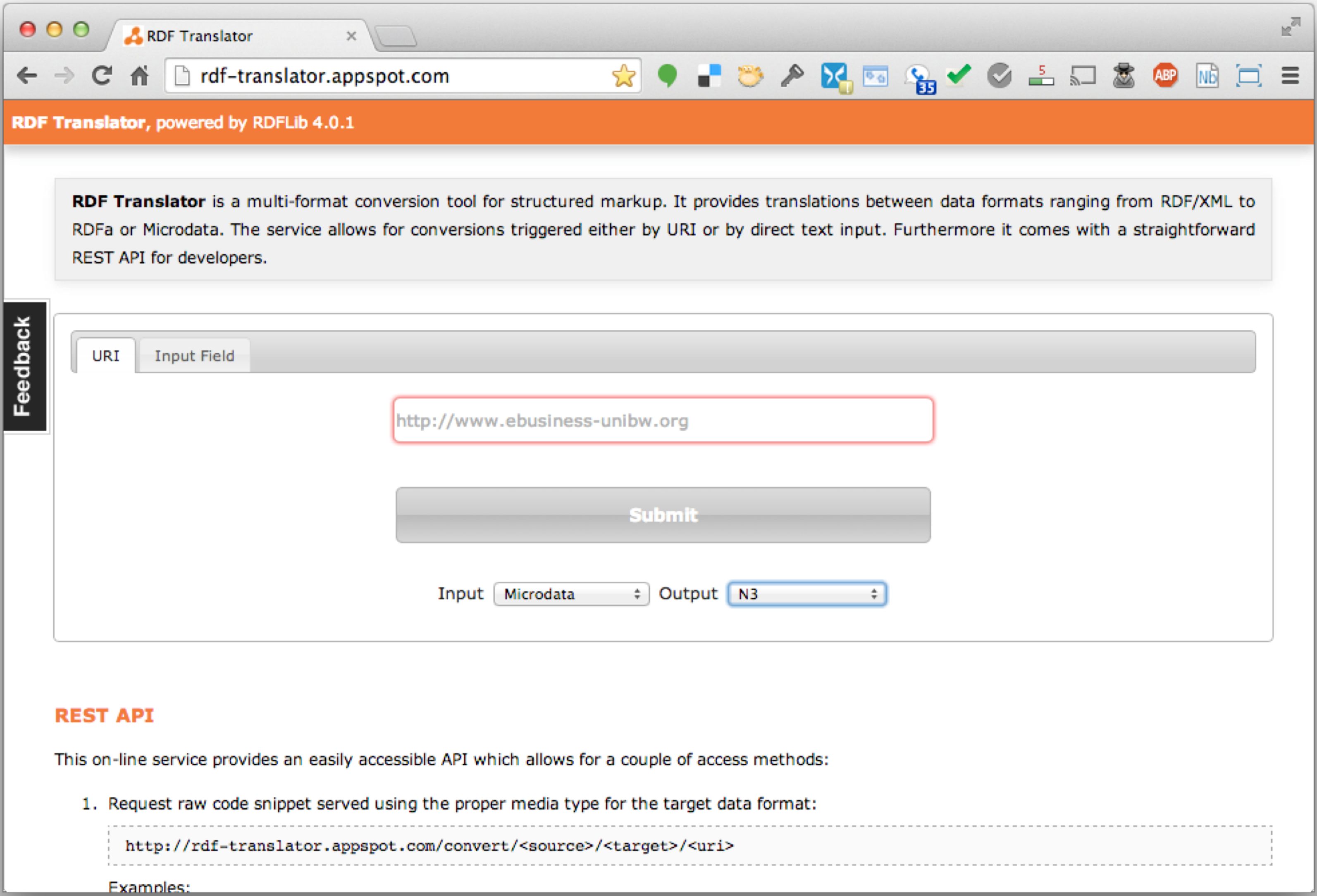
```
<div itemscope >
  <h1>Avatar</h1>
  <span>Director: James Cameron (born 1954) </span>
  <span>Science fiction</span>
  <a href=”avatar-trailer.html">Trailer</a>
</div>
```

An example: itemtype

- An *itemscope* attribute identifies a content *subtree* that is the subject about which we want to say something
- The *itemtype* attribute specifies the subject’s type

```
<div itemscope itemtype="http://schema.org/Movie">
  <h1>Avatar</h1>
  <span>Director: James Cameron (born 1954) </span>
  <span>Science fiction</span>
  <a href=”avatar-trailer.html">Trailer</a>
</div>
```

Microdata <-> RDF



RDF Translator, powered by RDFLib 4.0.1

RDF Translator is a multi-format conversion tool for structured markup. It provides translations between data formats ranging from RDF/XML to RDFa or Microdata. The service allows for conversions triggered either by URI or by direct text input. Furthermore it comes with a straightforward REST API for developers.

http://www.ebusiness-unibw.org

Input Microdata Output N3

REST API

This on-line service provides an easily accessible API which allows for a couple of access methods:

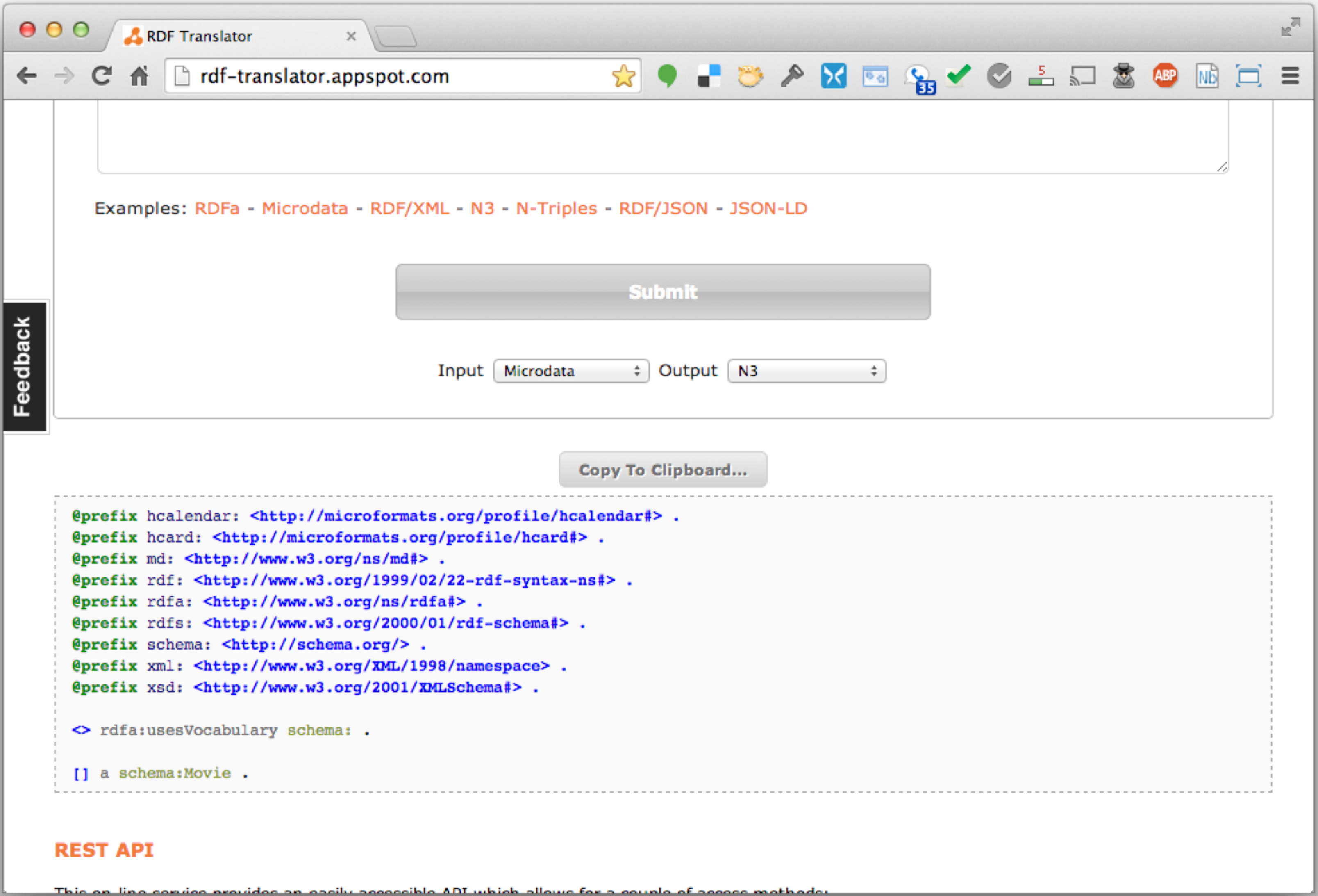
1. Request raw code snippet served using the proper media type for the target data format:

```
http://rdf-translator.appspot.com/convert/<source>/<target>/<uri>
```

Examples:

http://rdf-translator.appspot.com/

Microdata <-> RDF



Examples: RDFa - Microdata - RDF/XML - N3 - N-Triples - RDF/JSON - JSON-LD

Input Microdata Output N3

```
@prefix hcalendar: <http://microformats.org/profile/hcalendar#> .
@prefix hcard: <http://microformats.org/profile/hcard#> .
@prefix md: <http://www.w3.org/ns/md#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<> rdfa:usesVocabulary schema: .

[] a schema:Movie .
```

REST API

This on-line service provides an easily accessible API which allows for a couple of access methods:

http://rdf-translator.appspot.com/

An example: itemtype

- An *itemscope* attribute identifies content *subtree* that is the subject about which we want to say something
- The *itemtype* attribute specifies the subject’s type

```
[ ] a schema:Movie .
```

```
<div itemscope itemtype="http://schema.org/Movie">
  <h1>Avatar</h1>
  <span>Director: James Cameron (born 1954) </span>
  <span>Science fiction</span>
  <a href=”avatar-trailer.html">Trailer</a>
</div>
```

An example: itemprop

- An *itemscope* attribute identifies a content *subtree* that is the subject about which we want to say something
- The *itemtype* attribute specifies the subject's type
- An *itemprop* attribute gives a property of that type

```
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <span>Director: James Cameron (born 1954) </span>
  <span itemprop="genre">Science fiction</span>
  <a href=”avatar-trailer.html” itemprop="trailer">Trailer</a>
</div>
```

An example: itemprop

- An *itemscope* attribute identifies a content *subtree* that is the subject about which we want to say something
- The *itemtype* attribute specifies the subject's type
- An *itemprop* attribute gives a property of that type

```
[ ] a schema:Movie ;
    schema:genre "Science fiction" ;
    schema:name "Avatar" ;
    schema:trailer <avatar-trailer.html> .
```

```
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <span>Director: James Cameron (born 1954) </span>
  <span itemprop="genre">Science fiction</span>
  <a href=”avatar-trailer.html” itemprop="trailer">Trailer</a>
</div>
```

An example: embedded items

- An *itemprop* immediately followed by another *itemscope* makes the value an object

```
<div itemscope itemtype="http://schema.org/Movie">
 <h1 itemprop="name">Avatar</h1>
   <div itemprop="director"
          itemscope itemtype="http://schema.org/Person">
      Director: <span itemprop="name">James Cameron</span>
      (born  <span itemprop="birthDate">1954</span>)
  </div>
 <span itemprop="genre">Science fiction</span>
 <a href="avatar-trailer.html" itemprop="trailer">Trailer</a>
</div>
```

An example: embedded items

- An itemprop immediately followed by another itemscope makes the value an object

```
[ ] a schema:Movie ;
    schema:director [ a schema:Person ;
           schema:birthDate "1954" ;
           schema:name "James Cameron" ] ;
    schema:genre "Science fiction" ;
    schema:name "Avatar" ;
    schema:trailer <avatar-trailer.html> .
```

```
<div itemscope itemtype="http://schema.org/Movie">
 <h1 itemprop="name">Avatar</h1>
   <div itemprop="director"
         itemscope itemtype="http://schema.org/Person">
      Director: <span itemprop="name">James Cameron</span>
      (born  <span itemprop="birthDate">1954</span>)
  </div>
 <span itemprop="genre">Science fiction</span>
 <a href="avatar-trailer.html" itemprop="trailer">Trailer</a>
</div>
```

schema.org vocabulary

- Full type hierarchy in one file
- 605 classes, 911 properties (Nov ‘18)
- **Data types:** Boolean, Date, DateTime, Number, Text, Time
- **Objects:** Rooted at Thing with two ‘metaclasses’ (Class and Property) and eight subclasses
- See github repo for examples and code

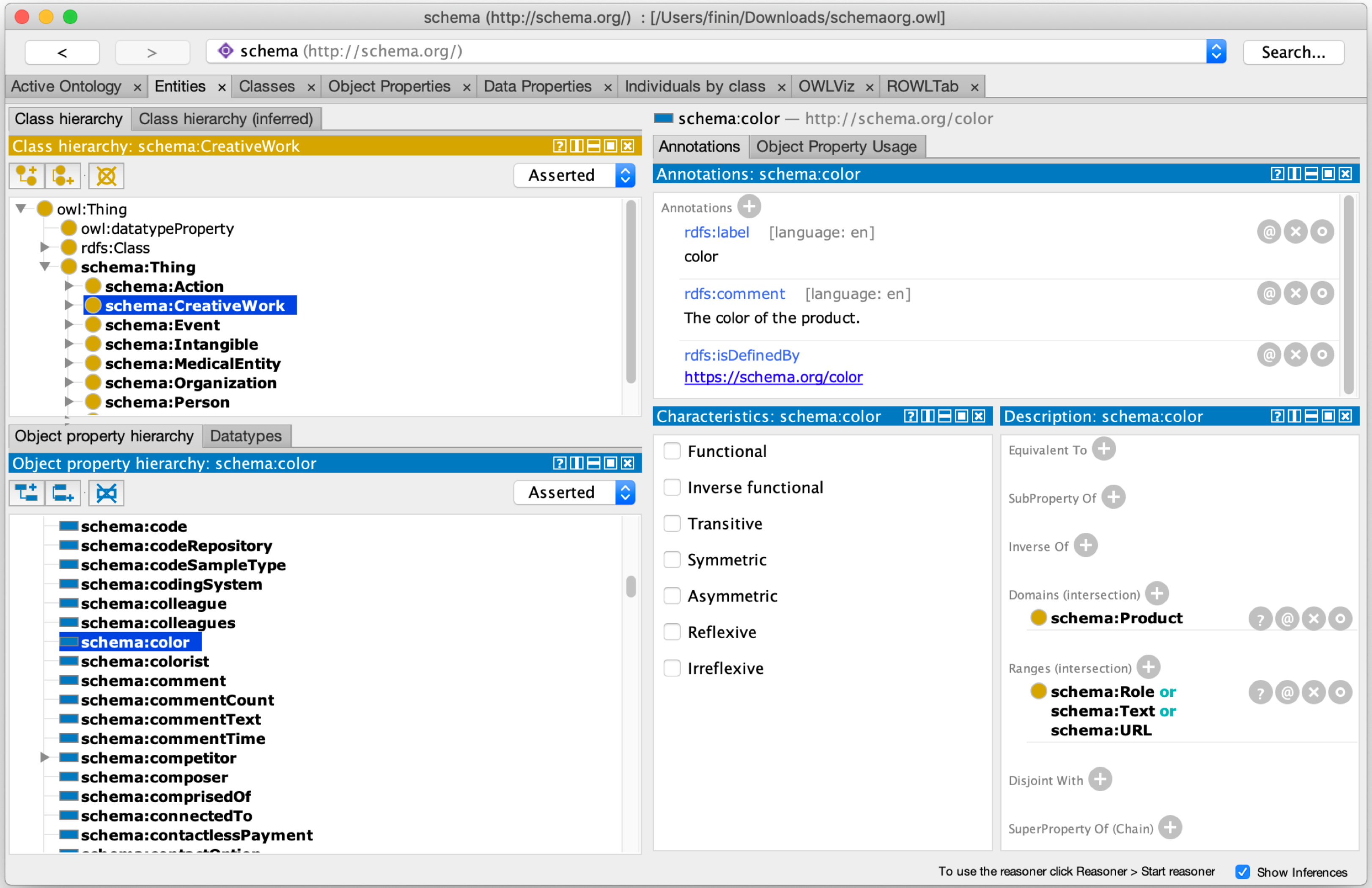
DataType
- Boolean
- Date
- DateTime
- Number
  - Float
  - Integer
- Text
  - URL
- Time

More specific types

- Class
- CreativeWork
- Event
- Intangible
- MedicalEntity
- Organization
- Person
- Place
- Product
- Property

Schemas as rdfs and owl?

See the schema.org developer page



schema (http://schema.org/) : [/Users/finin/Downloads/schemaorg.owl]

schema (http://schema.org/)

Class hierarchy: schema:CreativeWork

- owl:Thing
  - owl:datatypeProperty
  - rdfs:Class
  - schema:Thing
    - schema:Action
    - **schema:CreativeWork**
    - schema:Event
    - schema:Intangible
    - schema:MedicalEntity
    - schema:Organization
    - schema:Person

Object property hierarchy: schema:color

- schema:code
- schema:codeRepository
- schema:codeSampleType
- schema:codingSystem
- schema:colleague
- schema:colleagues
- **schema:color**
- schema:colorist
- schema:comment
- schema:commentCount
- schema:commentText
- schema:commentTime
- schema:competitor
- schema:composer
- schema:comprisedOf
- schema:connectedTo
- schema:contactlessPayment

schema:color — http://schema.org/color

Annotations: schema:color

Annotations

rdfs:label [language: en]
color

rdfs:comment [language: en]
The color of the product.

rdfs:isDefinedBy
https://schema.org/color

Characteristics: schema:color

- [ ] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

Description: schema:color

Equivalent To

SubProperty Of

Inverse Of

Domains (intersection)
schema:Product

Ranges (intersection)
schema:Role or
schema:Text or
schema:URL

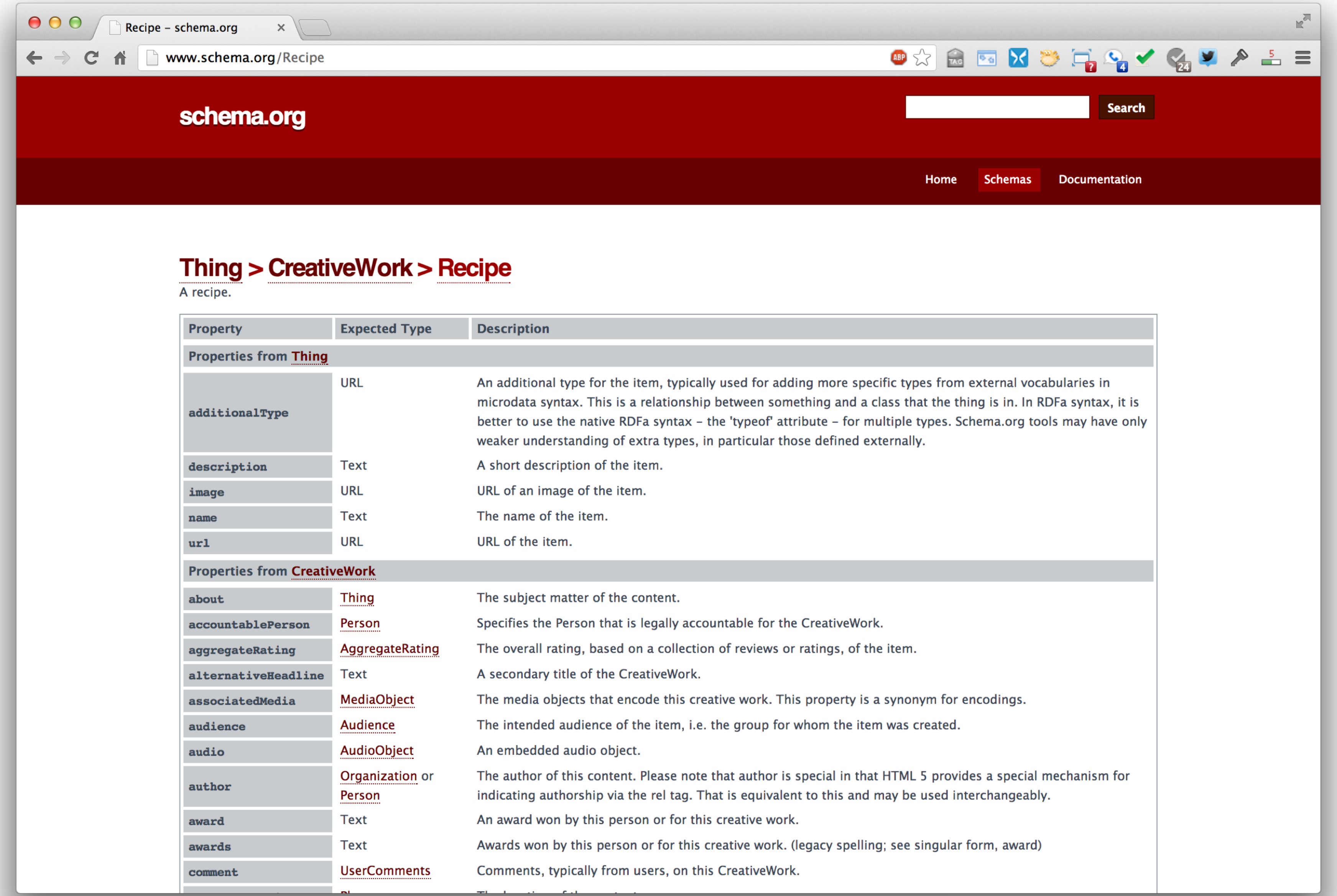
Disjoint With

SuperProperty Of (Chain)

To use the reasoner click Reasoner > Start reasoner

[✓] Show Inferences

http://www.schema.org/Recipe

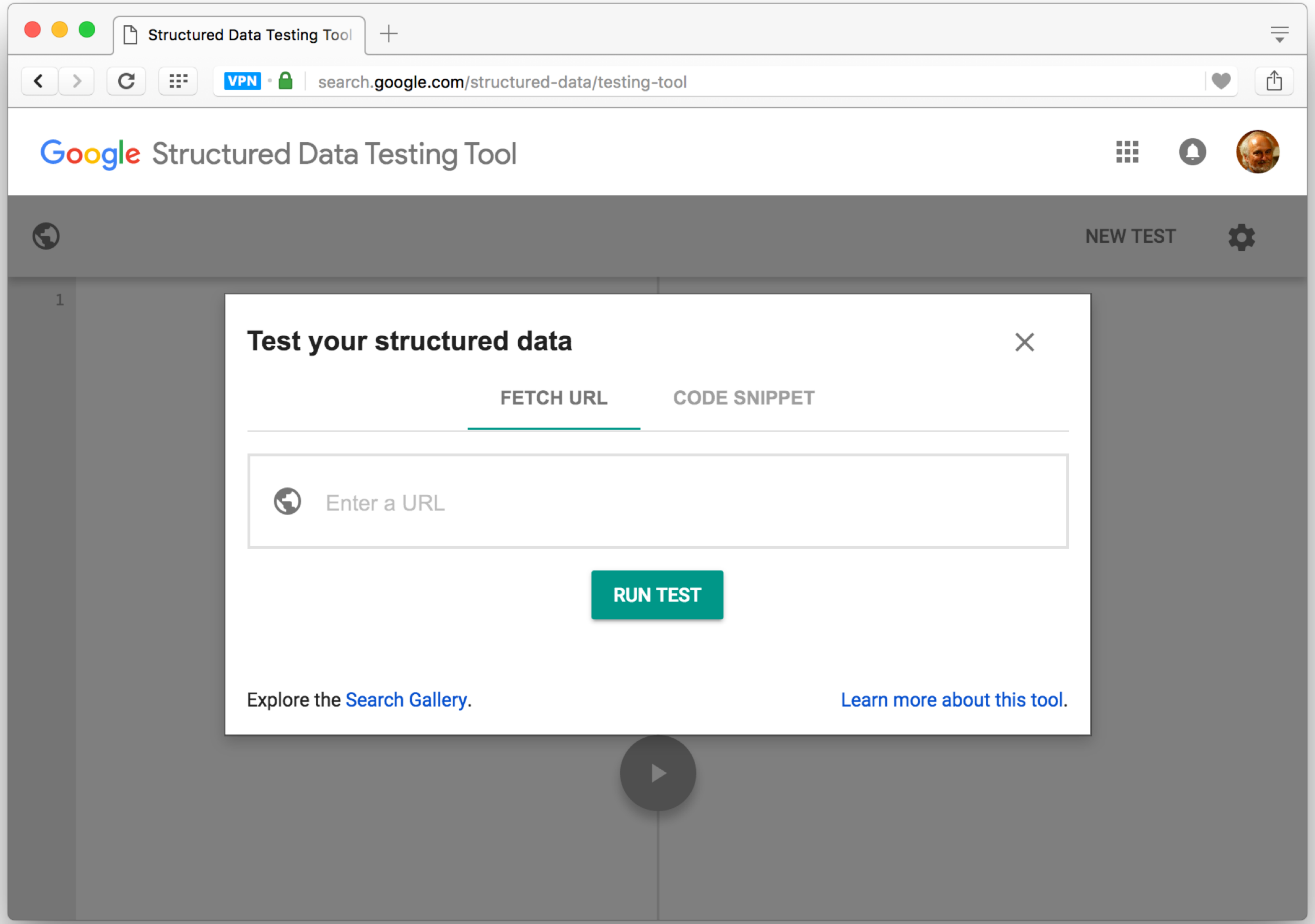


Thing > CreativeWork > Recipe

A recipe.

Property	Expected Type	Description
Properties from Thing		
additionalType	URL	An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. In RDFa syntax, it is better to use the native RDFa syntax – the 'typeof' attribute – for multiple types. Schema.org tools may have only weaker understanding of extra types, in particular those defined externally.
description	Text	A short description of the item.
image	URL	URL of an image of the item.
name	Text	The name of the item.
url	URL	URL of the item.
Properties from CreativeWork		
about	Thing	The subject matter of the content.
accountablePerson	Person	Specifies the Person that is legally accountable for the CreativeWork.
aggregateRating	AggregateRating	The overall rating, based on a collection of reviews or ratings, of the item.
alternativeHeadline	Text	A secondary title of the CreativeWork.
associatedMedia	MediaObject	The media objects that encode this creative work. This property is a synonym for encodings.
audience	Audience	The intended audience of the item, i.e. the group for whom the item was created.
audio	AudioObject	An embedded audio object.
author	Organization or Person	The author of this content. Please note that author is special in that HTML 5 provides a special mechanism for indicating authorship via the rel tag. That is equivalent to this and may be used interchangeably.
award	Text	An award won by this person or for this creative work.
awards	Text	Awards won by this person or for this creative work. (legacy spelling; see singular form, award)
comment	UserComments	Comments, typically from users, on this CreativeWork.

Testing Structured Data in HTML


Structured Data Testing Tool
search.google.com/structured-data/testing-tool
Google Structured Data Testing Tool
NEW TEST

Test your structured data

FETCH URL

CODE SNIPPET

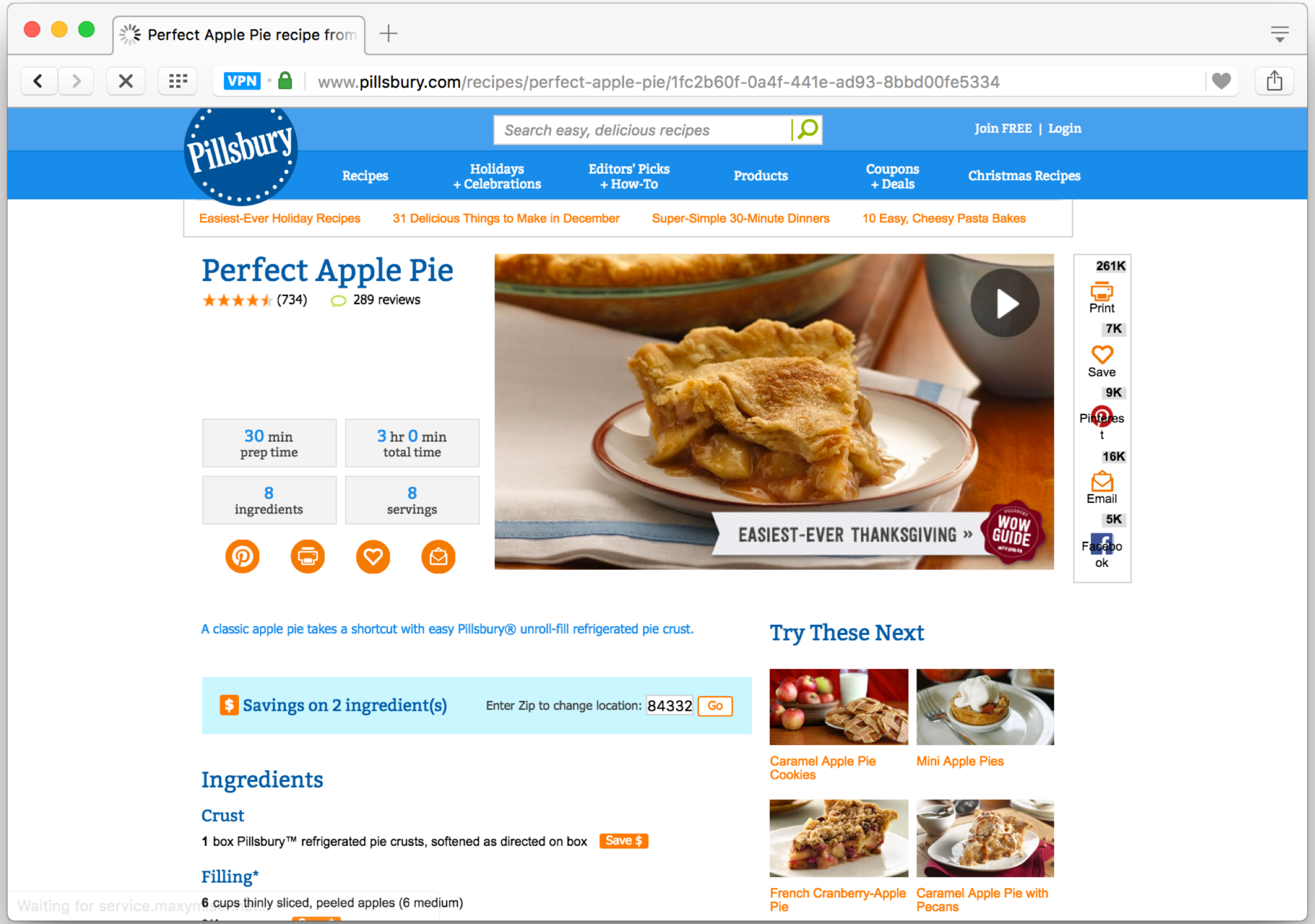
Enter a URL

RUN TEST

Explore the Search Gallery.

Learn more about this tool.

Testing Structured Data in HTML



Perfect Apple Pie recipe from

www.pillsbury.com/recipes/perfect-apple-pie/1fc2b60f-0a4f-441e-ad93-8bbd00fe5334

Search easy, delicious recipes

Join FREE | Login

Recipes | Holidays + Celebrations | Editors' Picks + How-To | Products | Coupons + Deals | Christmas Recipes

Easiest-Ever Holiday Recipes | 31 Delicious Things to Make in December | Super-Simple 30-Minute Dinners | 10 Easy, Cheesy Pasta Bakes

Perfect Apple Pie

★★★★½ (734) 289 reviews

30 min prep time	3 hr 0 min total time
8 ingredients	8 servings

A classic apple pie takes a shortcut with easy Pillsbury® unroll-fill refrigerated pie crust.

Savings on 2 ingredient(s) Enter Zip to change location: 84332 Go

Ingredients

Crust

1 box Pillsbury™ refrigerated pie crusts, softened as directed on box Save $

Filling*

6 cups thinly sliced, peeled apples (6 medium)

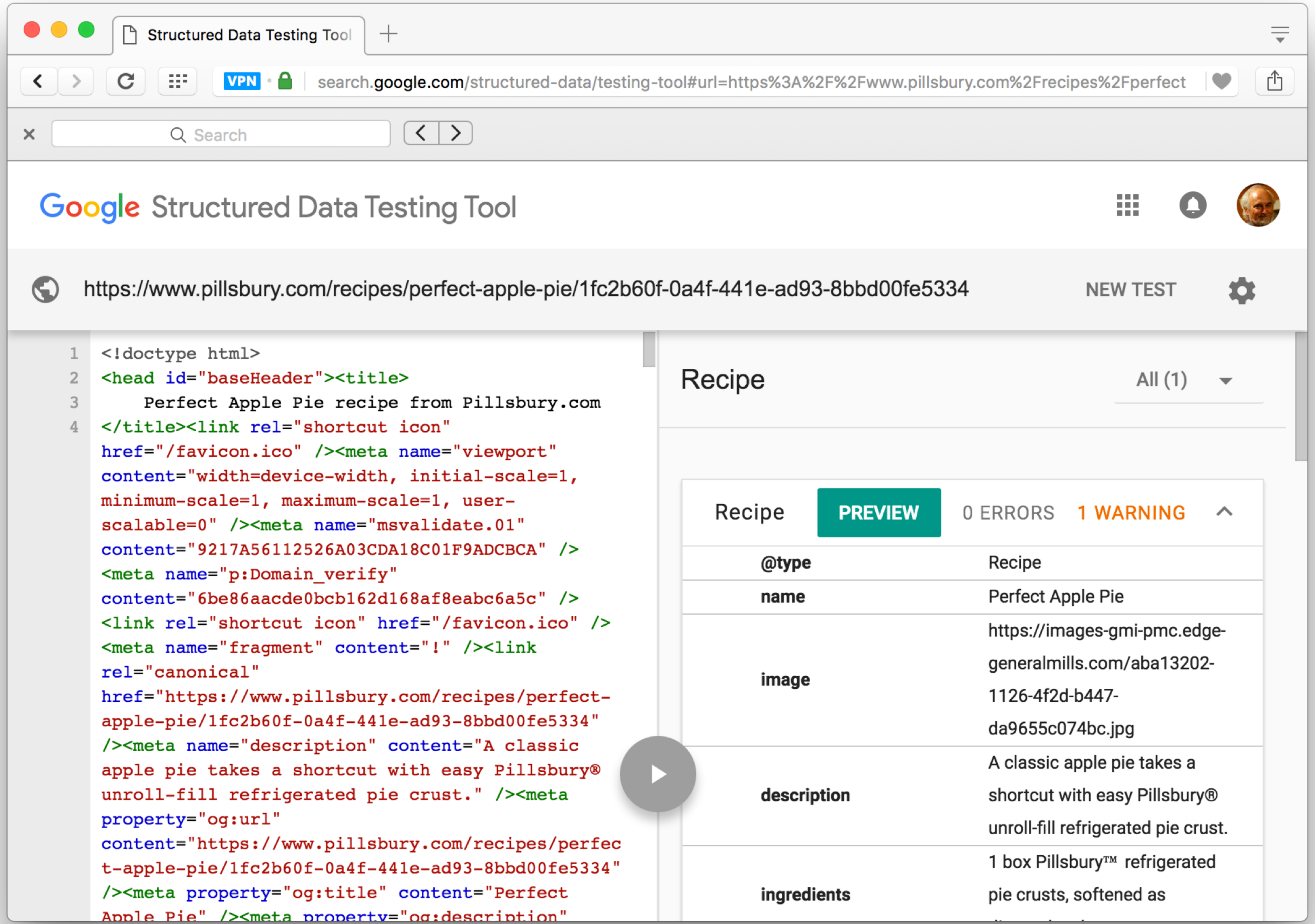
Try These Next

Caramel Apple Pie Cookies | Mini Apple Pies | French Cranberry-Apple Pie | Caramel Apple Pie with Pecans

Waiting for service.maxymiser.net...

https://www.pillsbury.com/recipes/perfect-apple-pie/1fc2b60f-0a4f-441e-ad93-8bbd00fe5334

Testing Structured Data in HTML


Structured Data Testing Tool
search.google.com/structured-data/testing-tool#url=https%3A%2F%2Fwww.pillsbury.com%2Frecipes%2Fperfect
Search
Google Structured Data Testing Tool
NEW TEST
PREVIEW

https://www.pillsbury.com/recipes/perfect-apple-pie/1fc2b60f-0a4f-441e-ad93-8bbd00fe5334

```
<!doctype html>
<head id="baseHeader"><title>
    Perfect Apple Pie recipe from Pillsbury.com
</title><link rel="shortcut icon"
href="/favicon.ico" /><meta name="viewport"
content="width=device-width, initial-scale=1,
minimum-scale=1, maximum-scale=1, user-
scalable=0" /><meta name="msvalidate.01"
content="9217A56112526A03CDA18C01F9ADCBCA" />
<meta name="p:Domain_verify"
content="6be86aacde0bcb162d168af8eabc6a5c" />
<link rel="shortcut icon" href="/favicon.ico" />
<meta name="fragment" content="!" /><link
rel="canonical"
href="https://www.pillsbury.com/recipes/perfect-
apple-pie/1fc2b60f-0a4f-441e-ad93-8bbd00fe5334"
/><meta name="description" content="A classic
apple pie takes a shortcut with easy Pillsbury®
unroll-fill refrigerated pie crust." /><meta
property="og:url"
content="https://www.pillsbury.com/recipes/perfec
t-apple-pie/1fc2b60f-0a4f-441e-ad93-8bbd00fe5334"
/><meta property="og:title" content="Perfect
Apple Pie" /><meta property="og:description"
```

Recipe   All (1)

Recipe   0 ERRORS   1 WARNING

@type	Recipe
name	Perfect Apple Pie
image	https://images-gmi-pmc.edge-generalmills.com/aba13202-1126-4f2d-b447-da9655c074bc.jpg
description	A classic apple pie takes a shortcut with easy Pillsbury® unroll-fill refrigerated pie crust.
ingredients	1 box Pillsbury™ refrigerated pie crusts, softened as

Microdata as a KR language

- More than RDF, less than RDFS
- Properties have an *expected* type (range)
  - Can be a list of types, **any** of which are OK
  - Might be a string for many properties (*"some data better than none"*)
- Properties attached ≥ 1 types (domain)
- Classes can have multiple parents and inherit (properties) from all of them
- No axioms (e.g., disjointness, cardinality, etc.)
- No relation like subPropertyOf

Mixing vocabularies

- Microdata is intended to work with just one vocabulary: the one at schema.org
- Advantages: simple and controlled
  - Simple, organized, well designed
  - Controlled by the schema.org people
- Disadvantages: too simple, too controlled
  - Too simple, narrow, mono-lingual
  - Controlled by the schema.org people

Extending schema.org ontology

- Extensions: hosted vs. external
  - Hosted: managed & published by schema.org project
- You can subclass existing classes
  - Person/Engineer
  - Person/Engineer/ElectricalEngineer
- Subclass existing properties
  - musicGroupMember/leadVocalist
  - musicGroupMember/leadGuitar1
  - musicGroupMember/leadGuitar2

Hosted Extensions 11/18

- auto.schema.org
- bib.schema.org
- health-lifesci.schema.org
- iot.schema.org
- meta.schema.org
- pending.schema.org

Extension Problems

- Hard to establish agreed upon meaning
  - Through axioms supported by the language (e.g., equivalence, disjointness, etc.)
  - No place for documentation (annotations, labels, comments)
- With no namespace mechanism, your Person/Engineer and mine can be confused and might mean different things
  - Is a Computer Scientist an engineer?

Serialization

- Schema.org has a data model and serializations
  - Microdata is the original, native serialization
  - RDFa is more expressive and works with the RDF stack
  - Everyone agrees that *RDFa Lite* is a good encoding: as simple as Microdata but more expressive
  - JSON-LD is an increasingly popular accepted encoding
- Search engines look for Microdata, RDFa and JSON-LD
- Schema.org considers RDFa to be the “canonical machine representation of schema.org”
- Bur Google recommends using JSON-LD

Conclusions

- Microdata is an effort by search companies to use a simple, controlled semantic language
- Its semantics is pragmatic
  - e.g., expected types: a string is accepted where a thing is expected – "some data is better than none"
- The real value is in
  - Supported vocabularies and
  - their use by Search companies

=> Immediate motivation for using semantic markup