LinqToWeb

Content

[Introduction 3](#_Toc254084739)

[Language 3](#_Toc254084740)

[Framework 3](#_Toc254084741)

[Compiler to C# 3](#_Toc254084742)

[Parallelism 4](#_Toc254084743)

[Conclusion 4](#_Toc254084744)

# Introduction

What? Why? Where? Extraction on the “live” web is performed. An abstraction over more websites is defined.

# Language

* Used for …
* Language grammar
* Elements (classes, methods, blocks, foreach, contexts, method call, literals, variables declaration, type cast, adding elements to lists, operators, no IF, no RETURN)
* Examples (everywhere)
* Variables: Extraction-Object and Value-Type
* Reason: minimalistic, more declarative, able to transform extracted data into any unified form, looks like simple synchronous extraction algorithm
* Methods declaration – main method = no name, same signature, same name = call them all
* Properties(public extracted data) – extraction-object arguments of main method
* Extraction arguments – value arguments main method

# Framework

* How does it work? (Diagram – framework, application, compiled code)
* ExtractionContext (public properties, methods, cache)
* Base objects – classes and lists (actions, parent)
* Extraction on request, diagram
* Getting data from objects, Actions list, DoNextAction method, parent action
* invoking best next action (GetNextAction) based on MethodDecl analysis
* ActionItem, Extraction method delegate (current data context, arguments)
* Calling actions, modifying and filtering arguments (extracting only requested data, replace object with the ghost that cannot obtain actions – does nothing, does not invoke method calls, see list)
* if method is called and some argument already processed this method (means it does not have this action in ActionList), remember him to do not add new actions within this method call
* ExtractionList, enumeration, synchronizing ActionList with the listContainer, if method is called and argument is a list and thius list is not enumerated now – replace it with its empty ghost (see modifying arguments)
* VariableList
* DataContext, dynamic creating (the current DataContext type is known only at runtime, dynamically using reflections get the proper method to create the new context on the current context – because every DataContext can contain different methods to create new DataContexts)
* ScopeStack
* StorageBase (knowledge base)
* Foreach methods, and dynamic variables

# Compiler to C#

* Element to what C# construct
* Analysis of MethodDecl, affected variables and how deep they are
* Adding elements to list in another object (a.list[] = x; // Convert to method call, list is special object, elements must be added only within methods that have this list as an argument – so user is enumerating him, otherwise only action can be added into the list … when enumerating list, actions are used to get items – see Framework/ExtractionList)
* Current context
* Context creating
* Foreach methods, “dynamic” variables, method signature, where they are placed
* Method call

# Parallelism

More contexts can be created on different threads, using the same cache object the data are safely and properly shared but the code is running on separated threads. (More actions are called at the same time, but the knowledge base is shared – the knowledge base is that slow thing)

# Conclusion

* Only requested data extracted
* able to be updated automatically
* does not require long extraction processes
* define common abstraction over various web resources