# dotNetRDF Design Document

## dotNetRDF Version 0.4.2

Library: dotNetRDF.dll  
Version: 0.4.2  
Target Date: July 2011  
Author: Rob Vesse  
Proposed Implementer: Rob Vesse  
Last Updated:

## Required Features

* SPARQL Engine Refactoring to improve extensibility
* RDFa Parser rewrite
* Additional in-memory structures for Triples

## Time Permitting Features

* SPARQL Optimisations
* Configuration API additions

## Known Issues/Bugs to Fix

* None at present

# Design

## SPARQL Engine Refactoring (Completed)

Refactor the Engine (particularly the Query side) so that it is much easier to extend the engine to work in new ways. Add an additional property on SparqlEvaluationContext which registers a ISparqlQueryAlgebraProcessor<SparqlEvaluationContext,BaseMultiset>. Then all existing Evaluate methods call an Apply() method on the context object which applies the registered processor (if any) OR calls the Evalute() method of the Algebra if not.

Also define a formal Extend() operator and implement its usage – deprecate LET support and move towards using Extend() instead of BindPattern inside a BGP.

Add additional methods to IUnaryOperator and IAbstractJoin which allow an algebra to be transformed more easily by optimisers. The Transform(IAlgebraOptimiser) method should basically return the current operator with the given optimiser applied to the inner algebra. Add a ITerminalOperator interface to mark non-BGP terminals e.g. Service()

## RDFa Parser rewrite

Based on the RDFa 1.1 specification it is clear that it would be best if the RDFa parser was rewritten from scratch in a modular manner that allows it to apply to a host language of choice. Provide modules for using it with HTML, XHTML, XML and SVG with the means to add additional ones in the future.

Rewriting the parser will also allow for efficiency and memory improvements to be made. The base parser architecture should be based upon RDFa 1.1 core and support extension points to allow for the extensions supported by different host languages.

## SPARQL Optimisations

### 1 – Parallelise Union() evaluation

Investigate whether Union() evaluation can be safely parallelised

### 2 – Improve Contains(String var, INode value) on Multiset (Completed)

Cache allowable values in a HashSet internally. Also consider exposing this publicly so that TriplePattern can use this as a possibly faster way of determining what pre-bound values to insert during evaluation

## Configuration API Additions (In-Progress)

Add to the Configuration API the ability to specify parsers and writers plus automatically associate them with specific MIME types, file extensions etc