

Семантический Web: введение

The Semantic Web: An Introduction

[Sean B. Palmer](#), 2001-09

<http://infomesh.net/2001/swintro>

Этот документ задуман как простая, но понятная вводная публикация для всех, кто пытается войти в область Семантического Web'a: от новичков до опытных хакеров. В качестве предварительного чтения мы рекомендуем статью Сварца "[Семантический веб для всех](#)" ([the Semantic Web in Breadth](#)).

СОДЕРЖАНИЕ

Что такое семантический Web?.....	2
Простое моделирование данных: схемы.....	7
Онтологии, логический вывод и DAML.....	9
Мощь языков семантического Web'a.....	11
Доверие и доказательство.....	13
Повсеместная информация и SEM.....	16
Эволюция.....	18
Это работает?.....	20
Что теперь? Читать.....	21

Что такое семантический Web?

Семантический Web это сеть информационных узлов, связанных друг с другом таким образом, что имеющаяся информация может легко обрабатываться компьютером, причем с минимальными ограничениями. Вы можете рассматривать его как эффективный способ представления данных на World Wide Web'e, или как глобально связанную базу данных.

Семантический Web был придуман Тимом Бернерсом-Ли (Tim Berners-Lee), изобретателем WWW, URI, HTTP и HTML. В консорциуме [W3C](#) существует специальная группа людей, работающая над усовершенствованием, расширением и стандартизацией обсуждаемой системы; уже разработано большое количество языков, публикаций, средств и т.д. Впрочем, технологии семантического Web'a все еще не вышли из детского возраста, и хотя будущее этого проекта представляется блестящим, общего согласия относительно вероятных направлений и характеристик раннего семантического Web'a не наблюдается.

Каковы рациональные предпосылки разработки такой системы? Данные, которые обычно спрятаны в файлах HTML, в некоторых контекстах часто бывают полезны, а в других нет. Проблема на Web'e с большинством данных, хранящихся в этой форме, состоит в том, что их трудно использовать в полном объеме, так как не существует глобальной системы публикации данных таким образом, чтобы их было легко обрабатывать каждому. Например, обратим внимание: информация о местных спортивных событиях, прогнозах погоды, расписаниях полетов самолетов, статистиках бейсбольной лиги, телевизионных программах... всю эту информацию можно найти на бесчисленных сайтах, но вся эта информация в HTML. Проблема здесь в том, что в некоторых контекстах эту информацию трудно использовать так, как бы этого хотелось.

Поэтому на семантический Web можно смотреть как на очень солидное инженерное решение... но это узкое представление. Мы увидим, что чем легче публиковать данные в форме, пригодной не только для публикации, но и как сырья для обработки, тем больше людей захотят публиковать свои данные, после чего последует взрыв или эффект домино. Мы могли бы обнаружить, что имеется очень много приложений семантического Web'a, которые можно использовать для самых разных задач, увеличивая на Web'e модулярность приложений. Но достаточно субъективных рассуждений... займется вопросом, как эта задача может быть решена.

Как правило, семантический Web строят на синтаксисах, которые для представления данных используют URI, обычно в структурах, содержащих триплеты: т.е. на множестве триплетов данных URI, которые можно хранить в базах данных или которыми можно обмениваться на World Wide Web, используя набор определенных, разработанных специально для этой цели, синтаксисов. Эти синтаксисы называются синтаксисами "Общей схемы описания ресурсов" (Resource Description Framework, RDF).

URI - Универсальный идентификатор ресурсов (Uniform Resource Identifier)

URI это просто идентификатор, подобный строкам, начинающимся с "http:" или "ftp:", с которыми вы часто сталкиваетесь на World Wide Web'e. Создать URI может каждый, право на их использование делегируется просто, поэтому они образуют

идеальную базовую технологию, с помощью которой можно начинать строить глобальный Web. Существенной характеристикой World Wide Web'a является именно это: все, что имеет URI, – "на Web'e".

За синтаксисом URI аккуратно следит комитет IETF, опубликованный им документ [RFC 2396](#) является общей спецификацией URI. Консорциум W3C поддерживает [список схем URI](#).

RDF - Общая схема описания ресурсов (Resource Description Framework)

Триплет можно определить просто как вектор трех URI. Язык, который использует три URI таким образом, называется RDF: W3C разработал на основе XML сериализацию RDF, ее "синтаксис" можно найти в документе "[Рекомендации по модели и синтаксису RDF](#)". RDF XML считается стандартным форматом обмена для RDF на семантическом Web'e, хотя это и не единственный формат. Например, Нотация3 (которую мы рассмотрим позднее в этой статье) является прекрасной альтернативой сериализации обычного текста.

Как только информация представлена в форме RDF, она становится легко поддающейся обработке, так как RDF является обобщенным форматом, для которого уже написано много парсеров. Спецификация RDF XML достаточно многословна, и может потребоваться некоторое время, чтобы к ней привыкнуть (например, для хорошего понимания RDF XML прежде всего необходимо в некоторой степени быть знакомым с XML и пространством имен...). Но лучше всего сейчас же рассмотреть пример документа на RDF XML:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/0.1/foaf/" >
  <rdf:Description rdf:about="">
    <dc:creator rdf:parseType="Resource">
      <foaf:name>Sean B. Palmer</foaf:name>
    </dc:creator>
    <dc:title>The Semantic Web: An Introduction</dc:title>
  </rdf:Description>
</rdf:RDF>
```

В этом небольшом документе RDF в сущности говорится о том, что у этой статьи название "The Semantic Web: An Introduction" и что она написана человеком с именем "Sean B. Palmer". А вот триплеты, которые выработывает RDF:

```
<> <http://purl.org/dc/elements/1.1/creator> _:x0 .
this <http://purl.org/dc/elements/1.1/title> "The Semantic Web: An
Introduction" .
_:x0 <http://xmlns.com/0.1/foaf/name> "Sean B. Palmer" .
```

Этот формат называется "[Нотация3](#)", в дальнейшем мы к нему еще обратимся; он является обычной текстовой сериализацией RDF. Заметим, что некоторые предпочитают работать с XML RDF а не использовать нотацию3, но общепринято считать, что нотацией3 пользоваться легче, и, разумеется, в любом случае она может быть конвертирована обратно в XML RDF.

Почему RDF?

Когда люди сталкиваются с XML RDF в первый раз, у них обычно возникает два вопроса: "зачем пользоваться RDF, а не XML?", и "Используем ли мы вместе с RDF и схемы XML?"

Ответ на первый вопрос очень прост и имеет две грани. Во-первых, польза от составления текста в RDF состоит в том, что наша информация отображается *непосредственно и недвусмысленно* на модель, которая является децентрализованной и для которой уже имеется много обобщенных парсеров. Это означает, что если у вас есть RDF-приложение, то вы знаете, какие биты данных относятся к семантике приложения, а какие – лишь к синтаксическому оформлению. И это знаете не только вы, это знает *каждый*; часто это происходит подсознательно, без чтения спецификации, так как конструкции RDF хорошо известны. Вторая часть ответа заключается в нашей надежде на то, что RDF-данные станут частью семантического Web'a, поэтому пользу от ваших данных в RDF можно теперь сравнить с записью информации в HTML в дни ранней молодости Web'a.

Ответ на второй вопрос почти также краток. Схемы XML это язык ограничения *синтаксиса* XML-приложений. RDF уже встроен в BNF¹, который устанавливает правила использования применяемого языка, поэтому безусловным ответом является железное "нет". Впрочем, использование схем XML вместе с RDF *может* оказаться полезным при создании типов данных и так далее. Следовательно, ответ – "возможно" с предупреждением, что он практически не используется для управления синтаксисом RDF. Это общее недопонимание, длящееся уже слишком долго.

Трактовка экрана, и формы

Для того чтобы семантический Web реализовал свои потенции, большинству людей необходимо начать публиковать свои данные в RDF. Откуда мы собираемся брать эту информацию? Достаточно много этой информации может быть извлечено из большинства существующих на сегодняшний день публикаций данных с помощью процесса, называемого "трактовкой экрана" ("screen scraping"). Трактовка экрана это действие по буквальному извлечению данных из некоторого источника и преобразованию их в более удобную форму (например, RDF) с помощью имеющихся под рукой средств. Двумя полезными средствами для трактовки экрана является XSLT (язык преобразования XML) и RegExps (в Perl, Python и т.д.).

Впрочем, трактовка экрана часто оказывается скучным решением, другим путем является построение подходящих систем RDF, которые принимают входные данные от пользователя и затем записывают их непосредственно в RDF. Например, данные аналогичные тем, которые вы вводите, подписывая новый почтовый счет, покупая CD online или производя поиск подержанного автомобиля, могут быть сохранены на языке RDF и дальше быть использованы на семантическом Web'e.

¹ BNF, Backus-Naur Form. Функция, распознающая правильные последовательности языка путём анализа синтаксической структуры переданного ей лексическим анализатором множества лексем. BNF Впервые был использован в сообщении о языке ALGOL-60 (прим.перевод.)

Нотация3: приспособленный RDF

Как мы видели выше, XML RDF может показаться трудным для освоения, но к счастью имеются более простые для овладения формы RDF. Одна из них называется "Нотация3", она была разработана Тимом Бернерсом-Ли. Для N3 имеется некоторая документация, включая версию [спецификации](#) и прекрасный [учебник](#).

Критерии при разработке нотации3 были достаточно просты: дать простой, легкий для изучения и написания формат RDF, который был бы удобен для грамматического разбора и построения на его основе больших приложений. В нотации3 мы можем просто записывать URI в триплетах, ограничивая их скобками "<" и ">". Вот, например, простой триплет, состоящий из трех URI:

```
<http://xyz.org/#a> <http://xyz.org/#b> <http://xyz.org/#c> .
```

Если хотите использовать литеральные значения, просто заключите само значение в двойные кавычки:

```
<http://xyz.org/#Sean> <http://xyz.org/#name> "Sean" .
```

Если Вы не хотите чему-нибудь подразумеваемому присваивать URI, то можете прибегнуть к имеющемуся на этот случай средству (это подобно тому, когда Вы говорите "есть некто по имени...", но URI ему не даете). Вы просто пишете символы подчеркивания и двоеточие, и затем небольшую метку:

```
_:a1 <http://xyz.org/#name> "Sean" .
```

Это можно прочесть как "Есть что-то с именем Sean" или "a1 для некоторого значения имеет имя Sean". Эти конструкции называются анонимными узлами, потому что у них нет URI, а иногда на них ссылаются как на узлы, связанные квантором существования.

Теперь, зачем эта избыточность при использовании нами трижды в одном из приведенных выше примеров URI "http://xyz.org/#", причем каждый раз менялся только последний символ? Нотация3 предоставляет нам прекрасный способ сократить эту запись: заменяя части URI на псевдонимы. Вот как объявляется псевдоним в нотации3:

```
@prefix xyz: <http://xyz.org/#> .
```

Заметьте, что вы должны всегда объявлять псевдоним до его использования. Использовать псевдоним очень просто. Например, вместо того, чтобы писать

```
<http://xyz.org/#a> <http://xyz.org/#b> <http://xyz.org/#c> .
```

Мы можем написать

```
@prefix xyz: <http://xyz.org/#> .
:a :b :c .
```

Не имеет значения, какой псевдоним вы используете для URI, если вы используете один и тот же во всем документе. Вы можете также объявить несколько псевдонимов. Следующие фрагменты кода оба эквивалентны приведенному выше фрагменту кода:

```
@prefix blargh: <http://xyz.org/#> .  
blargh:a blargh:b blargh:c .
```

```
@prefix blargh: <http://xyz.org/#> .  
@prefix xyz: <http://xyz.org/#> .  
blargh:a xyz:b blargh:c .
```

Однако необходимо отметить, что мы часто используем некоторые псевдонимы в большой мере стандартно, так что когда разработчики семантического Web'a обмениваются кодами в обычном тексте, они префиксы опускают, и остальным нужно догадываться, о чем здесь идет речь. Отметим, что код *не* должен реализовывать эту особенность. Вот пример некоторых "стандартных" псевдонимов:

```
@prefix : <#> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@prefix daml: <http://www.daml.org/2001/03/daml+oil#> .  
@prefix log: <http://www.w3.org/2000/10/swap/log#> .  
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

Пустой псевдоним ":" часто используется для обозначения нового пространства имен, для которого автор еще не создал URI (ай-ай). Мы используем его в нашем введении.

В нотации³ имеется много других маленьких конструкций, содержащих контексты, списки DAML и альтернативные пути представления анонимных узлов, но нам не обязательно здесь ими заниматься. Отметим, что был разработан синтаксис, являющийся еще более простым подмножеством нотации³, называемый [N-триплетами](#), но в нем не используются префиксы, и поэтому многие из примеров в этой статье некорректны как N-триплеты, но корректны в нотации³.

Дан Конноли (Dan Connolly) однажды назвал нотацию³ так: "авторское средство RDF для бедняги" (источник: RDF IG logs,2001-06-01 03:55:12). По-видимому, она названа нотацией³, потому, что "первым был RDF M&S, вторым - огородное пугало RDF, а третьим является это " (источник: [RDF IG F2F 2001-02-26](#)).

CWM: логическая машина для XML RDF и нотации³

Хотя логические машины мы будем обсуждать в этой статье позднее, сейчас мы заметим, что большая часть обработки в RDF и на семантическом Web'e (на этом этапе часто лишь в качестве эксперимента и демонстрации) делается с использованием написанной на языке [Python](#) программы [CWM](#), или "машины закрытого мира" (Closed World Machine). Дополнительную информацию можно найти на сайте [SWAP](#).

Сейчас лучшей демонстрацией использования этой машины будет преобразование из XML RDF в нотацию³ и обратно. Чтобы преобразовать "a.n3" в "a.rdf" просто напишите в командной строке:

```
python cwm.py a.n3 -rdf > a.rdf
```

CWM очень мощный инструментальный пакет семантического Web'a, и в этой статье мы будем часто на него ссылаться.

Простое моделирование данных: схемы

Первым "слоем" семантического Web'a над только что обсужденным синтаксисом является простая модель типизации данных. "Схема" это просто документ или фрагмент кода, управляющий множеством терминов в другом документе или фрагменте кода. Это как главный контрольный список, или грамматика определений.

Схема RDF

Схема RDF ([Рекомендации кандидату в схему RDF](#)) была разработана как простая модель типизации данных для RDF. Используя схему RDF, мы можем сказать, что "Тузик"("Fido") типа "собака"("Dog") и является подклассом животного. Мы можем также создавать свойства и классы, а также производить и некоторые более "продвинутые" вещи, такие, как создание диапазонов и областей для свойств.

Все термины для схемы RDF начинаются с "http://www.w3.org/2000/01/rdf-schema#" что, как может быть вы уже заметили, встречается выше в нашем списке "стандартных" псевдонимов. Псевдоним "rdfs:" очень часто используется для схем RDF, и мы здесь эту традицию продолжим.

Три наиболее важных понятия, которые дает нам RDF и схема RDF – это "Ресурс" (rdfs:Resource), "Класс" (rdfs:Class) и "Свойство" (rdfs:Property). Эти понятия являются "классами" в том смысле, что этим классам могут принадлежать термины. Например, все термины являются типами ресурса. Объявляя, что нечто имеет "тип" чего-то другого, мы просто пользуемся свойством rdfs:type:

```
rdfs:Resource rdfs:type rdfs:Class .
rdfs:Class rdfs:type rdfs:Class .
rdfs:Property rdfs:type rdfs:Class .
rdfs:type rdfs:type rdfs:Property .
```

Здесь просто говорится, что "у ресурса – тип класс, у класса – тип класс, у свойства – тип класс и у типа – тип свойство". Все эти утверждения истинны.

Создавать свои собственные классы очень просто. Например, создадим класс с именем "Dog" ("собака"), который содержит всех собак мира:

```
:Dog rdfs:type rdfs:Class .
```

Теперь мы можем сказать, что "У Fido – тип Dog".

```
:Fido rdfs:type :Dog .
```

Мы также легко можем создавать свойства, сказав, то у термина тип rdfs:Property, и затем использовать это свойство в нашем документе RDF:

```
:name rdf:type rdf:Property .
:Fido :name "Fido" .
```

Для чего мы сказали, что у Fido имя "Fido"? Потому что термин ":Fido" это URI, и мы могли бы спокойно выбрать для Fido любое URI, включая ":Загогулина" или ":n508s0srh". Нам пришло в голову выбрать URI ":Fido" потому, что его легче запомнить. Впрочем, нам все-таки приходится сообщать машинам, что его имя Fido, потому что машины не могут, в отличие от людей, догадаться об этом, глядя на URI, (да они, надо полагать, и не должны).

В схеме RDF имеется также несколько других свойств, которыми мы могли бы воспользоваться: `rdfs:subClassOf` и `rdfs:subPropertyOf`. С их помощью мы можем сказать, что один класс или свойство является подклассом или подсвойством другого. Например, нам могло бы потребоваться сказать, что класс "dog" является подклассом класса "Animal" ("животное"). Для этого мы просто говорим:

```
:Dog rdfs:subClassOf :Animal .
```

Отсюда следует, что когда мы говорим, что Fido это dog, мы также говорим, что Fido это Animal. Мы также можем сказать, что имеются другие подклассы животных:

```
:Human rdfs:subClassOf :Animal .
:Duck rdfs:subClassOf :Animal .
```

И затем создать новые экземпляры этих классов:

```
:Bob rdf:type :Human .
:Quacky rdf:type :Duck .
```

А еще мы можем выдумывать дополнительные свойства, использовать их, и сообщать новую информацию ...

```
:owns rdf:type rdf:Property .
:Bob :owns :Fido .
:Bob :owns :Quacky .
:Bob :name "Bob Fleming" .
:Quacky :name "Quacky" .
```

И так далее. Вы видите, что схема RDF очень проста, и тем не менее она позволяет построить из данных базу знаний очень, очень быстро.

Следующими понятиями схемы RDF, о которых следует сказать несколько слов, это диапазоны (ranges) и домены(domains). Диапазоны и домены позволяют нам сказать, каким классам субъект и объект каждого свойства должен принадлежать. Например, мы могли бы сказать, что свойство ":bookTitle" всегда должно применяться к книге и иметь литеральное значение:

```
:Book rdf:type rdfs:Class .
:bookTitle rdf:type rdf:Property .
:bookTitle rdfs:domain :Book .
:bookTitle rdfs:range rdfs:Literal .
:MyBook rdf:type :Book .
:MyBook :bookTitle "My Book" .
```

`rdfs:domain` всегда определяет, какому классу принадлежит субъект триплета, использующего это свойство, а `rdfs:range` всегда определяет, какому классу принадлежит объект триплета, использующего это свойство.

Схема RDF содержит также группу свойств для аннотаций схем, введения комментариев, меток и тому подобного. Двумя свойствами, с помощью которых вы можете сделать это, являются `rdfs:label` и `rdfs:comment`. Вот пример их использования:

```
:bookTitle rdfs:label "bookTitle";
  rdfs:comment "the title of a book" .
```

Всегда ставить метки и комментировать ваши новые свойства, классы и другие термины, безусловно, очень хорошая привычка.

Онтологии, логический вывод и DAML

[DAML](#) это язык, созданный агентством [DARPA](#) на основе RDF, как язык онтологий и логического вывода. DAML делает шаг вперед в формировании RDF, глубже развивая понятия свойств и классов. DAML по сравнению со схемой RDF более выразителен и возвращает нас к прежней дискуссии о предоставлении простых терминов для создания выражений логического вывода.

DAML+OIL

DAML предоставляет нам метод использовать такие понятия, как инверсии, однозначные свойства, уникальные свойства, списки, ограничения, кардинальность, попарно непересекающиеся списки, типы данных и т.д. Мы сейчас рассмотрим поподробнее некоторые из них, но вооруженные знаниями, которые вы уже получили и получите при чтении этого введения (я надеюсь, что вы ничего не пропустили!), вы с большой пользой для себя прочтете также и [DAML+OIL Walkthru](#).

Одной из конструкций DAML, с которой мы будем разбираться, является `daml:inverseOfProperty`. Используя это свойство, мы можем сказать, что одно свойство является инверсией другого. Значения предикатов `rdfs:range` и `rdfs:domain` для предиката `daml:inverseOf` принадлежат `rdf:Property`. Вот пример использования `daml:inverseOf`:

```
:hasName daml:inverseOf :isNameOf .
:Sean :hasName "Sean" .
"Sean" :isNameOf :Sean .
```

Второй полезной конструкцией DAML, которую мы рассмотрим, является класс `daml:UnambiguousProperty`. Утверждение, что некоторое свойство принадлежит классу `daml:UnambiguousProperty` означает, что если объекты этого свойства одинаковы, то субъекты эквивалентны. Например:

```
foaf:mbox rdf:type daml:UnambiguousProperty .
:x foaf:mbox .
:y foaf:mbox .
```

означает, что:

```
:x daml:equivalentTo :y .
```

Не раздражайтесь, если это вам немножко слишком.... Для понимания семантического Web'a это не очень важно, но это полезно, так как многие приложения семантического Web'a содержат конструкции DAML. Впрочем, DAML лишь один из серии языков (и т.д.), которые теперь используются.

Логический вывод

Принцип "логического вывода" очень прост: возможность выводить новые данные из данных, которые вы уже знаете. В математическом смысле, выполнение запроса это одна из форм логического вывода (например, возможность вывести из массы данных некоторый результат поиска). Логический вывод является одним из ведущих принципов семантического Web'a, потому что он позволяет нам очень легко создавать SW (Semantic Web) приложения.

Для демонстрации мощи логического вывода мы приведем некоторые простые примеры. Про автомобиль мы можем сказать, что

```
:MyCar de:macht "160KW" .
```

В немецкий процессор семантического Web'a вполне может быть встроен термин ":macht"; и, хотя в английский процессор вполне может быть встроен где-то эквивалентный термин, английский процессор этот код не поймет, так как не понимает один из написанных нами терминов. Тогда, чтобы компьютеру ситуация стала более ясной, сообщим ему строчку данных для логического вывода:

```
de:macht daml:equivalentTo en:power .
```

Мы воспользовались свойством DAML "equivalentTo" для объяснения компьютеру, что "macht" в немецкой системе эквивалентен термину "power" в английской системе. Теперь, обратившись к машине логического вывода, клиент семантического Web'a может воспользоваться триплетом

```
:MyCar en:power "160KW" .
```

Это всего лишь очень простой пример логического вывода, но можно сразу же увидеть, какие большие потенциальные возможности скрываются за этим. Объединение баз данных становится простым делом фиксирования где-то в RDF того, что "Person Name" в вашей базе данных эквивалентно "Name" в моей базе данных и затем механического объединения информации; обо всем остальном пусть думает компьютер.

И на самом деле, это уже возможно с помощью средств семантического Web'a, имеющихся в нашем распоряжении на сегодняшний день: CWM. Высокие уровни логического вывода могут быть обеспечены лишь в языках "Логики предикатов первого порядка" (First Order Predicate Language, FOPL), а DAML, к сожалению, не является полностью языком FOPL.

Логика

Чтобы семантический Web стал достаточно выразительным и смог помогать нам в самых различных ситуациях, возникает необходимость построения мощного логического языка, поддерживающего логический вывод. Продолжаются бурные дискуссии относительно методов, и даже возможности выполнения этой задачи; обращается внимание на то, что в RDF недостаточны возможности квантификации, и что эта область определена недостаточно хорошо. Проблемы логики предикатов подробно рассмотрены в прекрасной монографии Джона Сова (John Sowa's) Математические предпосылки (логика предикатов) [Mathematical Background \(Predicate Logic\)](#)

В частности, разработка для модели RDF, делающая эту ситуацию менее напряженной, производится Патом Хайесом (Pat Hayes 2001-09), но пока в этом вопросе еще много неясности. Конечно, это не остановит нас от использования на семантическом Web'e приспособленную к Web'у версию KIF² или подобный ему логический язык.

В любом случае, в нашем распоряжении имеется достаточный набор средств для построения семантического Web'a: утверждения (т.е. "и"), и цитирование (материализация) в RDF, классы, свойства, области и документирование в схеме RDF, непересекающиеся классы, свойства однозначности и уникальности, типы данных, инверсии, эквивалентности, списки и кое-что еще в DAML+OIL.

Обратите внимание на то, что в нотации³ имеется конструкция "контекст", позволяющая объединять формулировки вместе в одну группу и квантифицировать эту группу, используя специально разработанный для этой цели [логический словарь](#). С помощью этого словаря, например, можно выразить "или", применив термин словаря NANDs:

```
{ { :Joe :loves :TheSimpsons } a log:Falsehood .
  { :Joe :is :Nuts } a log:Falsehood .
} a log:Falsehood .
```

Что можно прочесть как "не верно, что Джо(Joe) не любит Симпсонов (The Simpsons) и не является занудой (nuts)". Я не поддаюсь соблазну объявить Джо универсальной квантифицированной переменной.

Отметим, что приведенный выше пример не поддается "правильной" сериализации в XML RDF, потому что в XML RDF нет контекстной конструкции, которая обозначена в нашем примере фигурными скобками. Впрочем, аналогичным средством может явиться использование конкретизации и контейнеров.

Мощь языков семантического Web'a

Основная выдающаяся черта языков семантического Web'a это то, что каждый может создать свой семантический язык путем простого опубликования документа RDF, в котором описывается набор URI, что URI могут делать, и как их можно использовать. И мы уже видели, что схемы RDF и DAML являются очень мощными языками для создания других языков.

² KIF (Knowledge Interchange Format) является компьютерным языком обмена информацией между различными программами (прим. перевод.)

Поскольку в наших языках для каждого термина используются URI, мы можем публиковать новые языки, не боясь того, что они будут неправильно интерпретированы или украдены, и мы знаем, что каждый, у кого есть обобщенный RDF процессор, сможет их использовать.

Принцип минимальных ограничений

Семантический Web существует, основываясь на принципе минимальных ограничений: чем меньше правил, тем лучше. Это означает, что семантический Web предоставляет большую свободу высказывания, и, следовательно, кто угодно может говорить что угодно о чем угодно. Когда вы всмотритесь в то, что семантический Web пытается делать, становится очевидным, почему необходим именно такой уровень возможностей. Если бы мы начали людей ограничивать, они были бы не в состоянии строить весь желаемый диапазон приложений, и поэтому семантический Web оказался бы для многих бесполезным.

Слишком много это сколько?

Впрочем, нам объяснили, что предоставление такой большой свободы, безусловно, является слишком большим риском... не найдутся ли люди, которые при обработке на машине логического вывода своих коммерческих документов внезапно будут охвачены новым планом устранения войн или создания изумительной новой симфонии?

Наш ответ (может быть, к сожалению!) – нет. Хотя базовые части семантического Web'a, RDF и лежащие за ним понятия накладывают минимальные ограничения, построенные на семантическом Web'e приложения будут разрабатываться для выполнения конкретных задач и, как таковые, будут хорошо определены.

Например, рассмотрим простую серверную программу регистрации. Кто-то может захотеть записать какие-то серверные протоколы на RDF, а затем создать программу, собирающую статистику регистраций, которые принадлежат сайту: сколько было посетителей за неделю и т.д. Это не значит, что эта программа превратит ваш гибкий диск в тостер или что-то в этом роде; она будет всего лишь обрабатывать серверные протоколы. Возможности, которые вы приобретаете от публикации вашей информации в RDF, состоят в том, что опубликованная в открытом для всех домене, она может быть гораздо легче *перенацелена* (использована для чего-то другого). Так как RDF пользуется URI, информация полностью децентрализована. Вам не нужно обращаться за разрешением к какому-то центральному органу опубликовать ваш язык и все ваши данные... вы можете это сделать сами. Это система управления данными по принципу *сделай сам*.

Педантичный Web

К сожалению, сообщество семантического Web'a заражено академическим и корпоративным интеллектуальным снобизмом, что привело к появлению штампа "педантичный Web". Все больше распространяется неверной, эмоционально окрашенной и даже зловредной информации. Заметим, что цель этого документа показать людям ошибочность некоторых общих представлений о семантическом Web'e.

Например, почти все новички в RDF проходят некую фазу "кризиса идентичности", когда они путают людей с их именами, а документы с их названиями. Нередко приходится встречаться с такими утверждениями как:

```
<http://example.org/> dc:creator "Bob" .
```

Но Bob это всего лишь литеральная строка, разве литеральная строка может написать документ? А вот что автор в действительности имел в виду:

```
<http://example.org/> dc:creator _:b .
_:b foaf:name "Bob" .
```

то есть этот example.org был создан кем-то, чье имя "Bob". Примерчики такого типа постепенно собираются, и некоторые из них выставляются на учебном сайте [SWTips](#), который поддерживается проектом совместной разработки

Образование и распространение информации

Уход от "педантичного Web'a" является до некоторой степени движением, направленным на разъяснение людям мощности семантического Web'a. Необходимость этого задокументирована так:

[...] идея, что введенные таким образом URI дают нам путь к схеме, которая полностью описывает сам язык и является очень простой (только два {посчитайте сами – 2} возможных "утверждения"), и, тем не менее, выглядит как рецепт полета на Марс, слегка ошеломляет. Сама ее простота позволяет нам говорить о чем угодно – от документа к языку по правилам! Это фундаментальное средство семантического Web'a, оно наделяет "людей силой" говорить что угодно о чем угодно.

Гуру для простаков, Вильям Лафборо ([EARL for dummies](#) William Loughborough), май 2001.

Но RDF и DAML+OIL это языки, которыми просто так не овладеешь; что же делается для помощи тем, у кого нет времени и терпения листать учебники, а есть желание писать приложения для семантического Web'a? К счастью, большинство приложений семантического Web'a будут конечными приложениями пользовательского уровня, и вам не обязательно иметь больше знаний о RDF, чем редактор [Amaya](#)³ требует знаний об (X)HTML.

Доверие и доказательство

Следующим шагом в разработке семантического Web'a является доверие и доказательство. Об этом уровне написано очень мало; это недопустимо, так как в будущем он будет очень важен.

В суровой действительности наипростейшим образом это звучит так: если один человек говорит, что x – что-то голубое, а другой говорит, что x не голубое, то не следует ли из этого, что семантический Web развалится на части?

³ [Amaya](#) — свободный веб-браузер и редактор веб-страниц, разрабатываемый консорциумом [W3C](#). Доступен для операционных систем [Unix](#), [Windows](#), [Mac OS X](#) (прим. перевод.)

Разумеется, нет, так как а) приложения на семантическом Web'e вообще зависят от контекста, и б) приложения в будущем будут, как правило, содержать механизмы проверки доказательства и цифровые подписи.

Контекст

Приложения на семантическом Web'e будут учитывать контекст в целом для того, чтобы сообщать пользователям, должны ли они доверять текущим данным. Если я получаю от друга поток данных RDF об увиденной им кинокартине и об его оценке этой кинокартины, я знаю, что я буду доверять этой информации. Более того, я могу затем воспользоваться этой информацией, не сомневаясь в ее источнике. Далее я оставляю на свое собственное усмотрение насколько мне верить полученному критическому отзыву о фильме.

Над разделяемыми контекстами работают также и группы людей. Если некая группа разрабатывает на семантическом Web'e информационную службу для художников, каталогизируя людей, их имена и места, где находятся картины этих людей, то мое доверие к этой группе зависит от того, насколько я доверяю участвующим в этой группе людям.

Таким образом, контекст является хорошим подспорьем, потому что он позволяет нам работать в локальных средах, опираясь на интуицию, не обращаясь к сложным системам аутентификации и проверки. Но что случится, если мы знаем корреспондента, но не знаем способа удостовериться, что какой-то квант данных RDF пришел именно от него? Здесь нам на помощь приходят цифровые подписи.

Цифровые подписи

Цифровые подписи это небольшие фрагменты кода, которые можно использовать для однозначной проверки того, кто написал тот или иной документ. Большинство, вероятно, уже знакомо с этой технологией: это тот же самый ключ в стиле PGP⁴, который используется для шифровки и подписи сообщений. Мы просто прилагаем эту технологию к RDF.

Предположим, например, что у меня есть информация в RDF, содержащая ссылку на цифровую подпись:

```
this :signature <http://example.org/signature> .
:Jane :loves :Mary .
```

Чтобы удостовериться в том, что Jane действительно любит Mary, мы можем загрузить текст RDF в машину проверки надежности источника (в машину логического вывода со встроенной программой проверки цифровой подписи), и продолжать дальнейшую обработку информации, если мы доверяем источнику информации.

⁴ PGP (*Pretty Good Privacy*) — компьютерная программа, позволяющая выполнять операции шифрования (кодирования) и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде (прим. перевод.)

Языки проверки истинности

Язык проверки истинности это просто язык, который нам позволяет проконтролировать, является или нет утверждение истинным. Реализация языка проверки обычно состоит из списка "элементов" логического вывода, которые используются для получения искомой информации, а также для последующей проверки информации о доверии для каждого из этих элементов.

Например, мы хотим проверить, что Joe loves Mary. В процессе поиска информации обнаруживаем два документа на сайте, которому доверяем. Первый из них говорит, что ":Joe loves :MJS", а второй из них говорит, что ":MJS daml:equivalentTo :Mary". У нас также есть контрольные суммы персональных файлов от администратора сайта.

Для проверки этой информации мы может занести контрольные суммы в локальный файл и затем актуализировать определенные правила FOPL, которые утверждают, что "если файл 'a' содержит информацию Joe loves Mary и у него контрольная сумма md5:0qrhf8q3hfh, то зафиксировать SuccessA". "Если файл 'b' содержит информацию MJS is equivalent to Mary и у него контрольная сумма md5:0892t925h, то зафиксировать SuccessB" и "если SuccessA и SuccessB, то Joe loves Mary".

Соответствующий пример в нотации 3 можно найти в некоторых из авторских экспериментов по проверке истинности. Приводим файл с правилами проверки:

```
@prefix : <http://infomesh.net/2001/proofexample/#> .
@prefix p: <http://www.w3.org/2001/07/imaginary-proof-ontology#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

p:ProvenTruth rdfs:subClassOf log:Truth .

# Proof

{ { <a.n3> p:checksum <md5:blargh>;
      log:resolvesTo [ log:includes { :Joe :loves :MJS } ] } }
log:implies
{ :Step1 a p:Success } } a log:Truth .

{ { <b.n3> p:checksum <md5:test>;
      log:resolvesTo [ log:includes { :MJS = :Mary } ] } }
log:implies
{ :Step2 a p:Success } } a log:Truth .

{ { :Step1 a p:Success . :Step2 a p:Success }
log:implies
{ { :Joe :loves :Mary } a p:ProvenTruth } } a log:Truth .
```

Файл говорит сам за себя. При использовании для его обработки CWM он работает правильно, давая ожидаемый результат. CWM не обладает способностью автоматической проверки контрольных сумм или цифровых подписей, но скоро соответствующая машина контроля доверительности на семантическом Web'e будет написана.

Повсеместная информация и SEM

Мы немного касались вопроса объема информации, но рассмотрим подробнее, что мы в действительности имеем в виду, когда говорим о "локальных" и "глобальных" системах.

Говоря о масштабировании систем, мы, как правило, различаем системы малые и большие. Взаимодействие между системами этих двух типов будет, по всей видимости, составлять большую часть транзакций на семантическом Web'e. Определим, что мы понимаем под большими, средними и малыми системами.

Большие системы

Примером большой системы являются две компании, которые подвергаются слиянию и выполняют работу по объединению двух баз данных. Другим примером могло бы быть компилирование механизмов поиска для баз данных огромного объема. Большие системы на семантическом Web'e имеют дело, обычно, с большими базами данных, и для работы с этими базами требуется много очень точных правил вывода и соответствующих процессоров.

Средние системы

Средние системы семантического WEB'a пытаются воспользоваться возможностями больших семантических систем, или являются объединением аналогичных малых систем. В первом случае примером является компания, которая пробует частично разобраться в двух сложных форматах накладных для их дальнейшего совместного использования. Примером второго случая является работа двух групп по языкам адресных книг, пытающихся создать язык одной адресной супер-книги.

Малые системы

Малые системы семантического Web'a обсуждаются менее широко. Под такими системами мы понимаем языки, которые будут использоваться главным образом в режиме off line, или небольшие объемы данных, которыми будут обмениваться, например, друзья, отделы или даже две компании.

Разделение данных на локальном уровне это очень хороший пример того, как семантической Web может быть полезен в тысячах и тысячах ситуаций. В следующем разделе, посвященном эволюции, мы увидим, как взаимодействие между системами разного масштаба будет являться ключевым моментом семантического Web'a.

SEM – Семантическая память (SEmantic Memory)

Понятие семантической памяти (SEmantic Memory) было введено Сетом Русселлом (Seth Russell), предложившим, чтобы RDF-дампы личных баз данных, собранных из "остатков" семантического Web'a (род семантического облака), являлись руководящей основой поддержания корректного понимания данных. Другими словами, SEM может мыслиться как группы данных, существенных для всего семантического WEB'a (то есть, схемы всех главных языков, таких, как XML RDF, схема RDF, DAML+OIL, и т.д.). Сюда же относятся локальные данные, важные для всех исполняемых приложений

семантического Web'a (например, информация о логическом пространстве имен для CWM, встроенных на текущий момент), и данные, непосредственно используемые и публикуемые, или которые были введены в базовый контекст SEM каким-либо иным способом.

Внутренняя структура SEM не будет, по всей видимости, ограничиваться обычной триплетной структурой RDF, возможно придется использовать структуру квартлетов (quads) и даже пентлетов (pents). Дополнительные поля нужны для ссылок на контекст (StID) и, возможно, на последовательности. Другими словами, это способ объединения информации в группы *в рамках* самой SEM для удобств сохранения и обновления. Например, будет нетрудно удалить любой триплет, который был добавлен в определенном контексте, удаляя все триплеты с этим конкретным StID.

Большое внимание в работах по семантическому Web'у было уделено вопросу интероперабельности систем хранения данных (то есть SEMам), и это хорошо; но гораздо меньше внимания было уделено тому, что фактически происходит в самой SEM, и это не очень хорошо, так как необходимость представления в RDF квартлетов и пентлетов остается висеть в воздухе. Очевидно, что утверждения можно смоделировать так же, как и для конкретизации:

```

rdf:Statement rdfs:subClassOf :Pent .
_:s1 rdf:type :Pent .
_:s1 rdf:subject :x .
_:s1 rdf:predicate :y .
_:s1 rdf:object :z .
_:s1 :context :p .
_:s1 :seq "0" .

```

Но ясно, что предлагаемый формат пентлетов по существу более эффективен и не подвержен опасности конкретизации.

```

:x :y :z :p "0" .

```

Этот язык нуждается также в наличии флага контекста по умолчанию, указывающего на корневой контекст документа. Корневой контекст документа это пространство, в которое производится грамматический разбор утверждений (не цитируемых); память концептуальной информации, в которой все утверждения считаются истинными. Любая цитируемая информация документа (например, использующая контекстный синтаксис нотации³) будет принадлежать другому (возможно анонимному) контексту, а не корневому контексту документа.

TimBL⁵ по-видимому, судя по исходному тексту CWM, пользуется для обозначения корневого контекста документа строкой "...#_formula", что (если это так) выглядит несколько капризной лошадкой. Что если кто-то создаст свойство с тем же самым URI? Поддержка интероперабельности на данном уровне семантического Web'a это важная задача, которой разработчикам семантического Web'a необходимо уделять должное внимание.

⁵ Tim Berners-Lee

Эволюция

Очень важным понятием семантического Web'a является понятие эволюции: переход от одной системы к другой. Двумя ключевыми моментами здесь являются *частичное понимание* и *трансформируемость*. Мы сейчас увидим, как эти качества проявляются естественным образом при изменении масштаба системы.

Частичное понимание: от большого масштаба к среднему

Концепция частичного понимания очень важна для семантического Web'a. Она нередко встречается в ранних документах, появившихся примерно в то же время, когда стали создаваться теории семантического WEB'a.

Примером частичного понимания при переходе от крупномасштабной системы к системе среднего масштаба может быть компания, где пытаются разобраться в двух накладных, одной из компании А и другой из компании В. Всем хорошо известно, что в обеих компаниях в накладных используются подобные поля, поэтому компания, пытающаяся разобраться в накладных, может легко скомпилировать главный список отправлений, просто вычеркнув данные из языков этих накладных. Как компании А, так и компании В не обязательно знать об этом процессе.

И TimBL включил этот пример в свои заключительные заметки по XML 2000:

[...] что закончится в будущем вопросами конвертации; например, [...] в семантическом Web'e у нас будет взаимосвязь между двумя языками, так что если вы получите накладную на языке, который вы не понимаете, а у вас есть... программное бизнес-обеспечение, которое может оплачивать накладные,... проследивая ссылки по семантическому Webу, ваша машина будет способна сделать автоматическую конвертацию накладной из одного языка в другой и затем ее обработать.(TimBerners-Lee).

Трансформируемость: от меньшего масштаба к среднему

Примером малых систем семантического Web'a, объединенных вместе в систему семантического Web'a среднего масштаба, могут служить две группы, которые опубликовали свои форматы адресной книги и теперь хотят создать расширенный и улучшенный формат адресной книги путем слияния этих двух текущих форматов. Каждый, кто использует один из этих старых форматов, мог бы, по-видимому, конвертировать его в новый формат, и отсюда видна значимость интероперабельности. Это как раз то, что обычно и происходит, когда переходят от малых систем семантического Web'a к системе семантического Web'a среднего масштаба, хотя при этом приходится сталкиваться с рядом неудобств и проблем несовместимости. Семантический Web снижает остроту этого и автоматизирует 99% необходимой работы (он может конвертировать поле А в поле В, но он не может внести за вас любые новые данные... конечно, новые поля можно на какое-то время оставить пустыми).

Содействие эволюционности

Как мы документируем эволюцию языков? Это очень важный и срочный вопрос, который TimBL сформулировал достаточно акkuratно.

Там, например, где схема библиотеки Конгресса говорит об "авторе", а Британская библиотека говорит о "создателе", один бит в RDF смог бы информировать о том, что для любой персоны x и любого ресурса y , если x (LoC) автор y , то x это (BL) создатель y . Это пример правила, решающего проблемы эволюционированности. Где процессор может найти это правило? [...]

[Дорожная карта](#) семантического Web'a, Tim Berners-Lee

Один из возможных ответов: в сторонней базе данных. Очень часто бывает не практично хранить в записях LoC или BL (как в примере TimBL'a) информацию о том, что два поля означают одно и тоже. Следовательно, эта информация должна будет записываться третьей, имеющей хорошую репутацию, стороной.

Одна такая "третья сторона", которой была поставлена задача исследовать этот вопрос, - [SWAG](#) (the Semantic Web Agreement Group), арбитражная группа семантического Web'a. Ее организаторами были Seth Russell, Sean B. Palmer, Aaron Swartz, and William Loughborough, Работа этой группы направлена на обеспечение интероперабельности в семантическом Web'e. Они создали то, что можно было бы назвать первым сторонним словарем семантического Web'a, [WebNS SWAG Dictionary](#).

В полет: трудно, но надо

Хотя в настоящее время семантический Web, как целое, все еще в зародышевом состоянии, народ начинает обращать на него внимание; издается информация, использующая RDF, делая ее, таким образом, пригодной для семантического Web'a.

Однако делается еще далеко не все для полноценного объединения информации... другими словами, "семантическая" часть "семантического Web'a" движется вперед достаточно уверенно, но где же "Web"? Никто эффективно не пользуется терминами, введенными другими; при использовании других терминов всего лишь бесцельно пытаются как-то повысить качество кода, генерируя только шум. Если вы собираетесь воспользоваться чьими-то данными, попытайтесь заранее выяснить, будет ли от этого какая либо польза. Например, лишь от того, что вы в своем RDF воспользовались термином "dc:title" вместо привычного ":title", будет ли это означать, что сразу же приложение DublinCore сможет понимать ваш код? Конечно, нет. на самом деле это означает, что если свойство "dc:title" в вашем случае введено для возможности применения так, чтобы информацию можно было использовать для других целей в ближайшем будущем, то вероятно вам это *может* пригодиться, так как "dc:title" термин, используемый в большинстве случаев.

Другая сторона этого вопроса может быть связана с проблемой, подобной той, с которой столкнулись на ранних этапах Всемирной паутины: зачем обременять себя публикацией Web-сайта, если нет никакого другого сайта, к которому нужно присоединиться или который хотел бы присоединиться к нам? Зачем обременять себя публикацией Web-сайта, если почти ни у кого нет браузеров? И зачем писать браузер, когда так мало Web-сайтов? Кто-то должен сильно потрудиться, чтобы все это появилось, а это медленный процесс.

Что можно сделать в этой ситуации? Может быть, имеет смысл рассортироваться. Еще один хорошо известный принцип, который вполне применим к приложениям

семантического Web'a, заключается в том, что не нужно изобретать велосипед; а именно, если кто-то уже сочинил схему, содержащую осмысленный, хорошо понимаемый и используемый набор терминов, который вы тоже хотите использовать в своем приложении, то не нужно пытаться заново повторить уже проделанную работу. Иногда это может привести к некоторому виду "войны схем", но выживание самой удачной обеспечит то, что важная часть лучшей схемы будет использоваться чаще всего. По-видимому, именно это имеет в виду TimBL, когда говорит, что термины будут просто "всплывать" из семантического облака. Что когда народ продолжает использовать термин "zip" без ссылки в записи, что его термин "zip" эквивалентен вашему термину "zip", который эквивалентен термину "zip" кого-то третьего, то мы будем просто использовать одно и тоже URI; здесь мы выходим на широкий путь повышения качества интероперабельности.

Это работает?

Какие имеются приложения семантического Web'a?

Я обращался к этому вопросу в предыдущей статье: [The Semantic Web: Taking Form](#) (Семантический Web: становление), но стоит повторить еще раз: семантический Web *уже* работает и народ его использует.

Приложения семантического Web'a

К несчастью, семантический Web во многом отличается от World Wide Web, включая и невозможность непосредственного направления человека к Web-сайту, чтобы он сам увидел, что это такое и как оно работает. Впрочем, ряд небольших приложений семантического Web'a написан и работает. Одним из лучших является эксперимент Дана Конноли (Dan Connolly) для диаграмм дуг и вершин:

Одной из целей деятельности в области семантического Web'a по компонентам усовершенствованной разработки является демонстрация того, как RDF и технологии семантического Web'a могут быть использованы в проекте W3C для повышения эффективности, надежности и т.д. На ранних этапах построения RDF-модели процесса W3C для визуализация разрабатываемой модели я использовал инструмент, который, в конечном счете, оказался очень успешным, так что я начал применять его, где только мог.

[Диаграммы кружков и стрел при использовании правил стилей, Дан Конноли.](#)

Конечно, этот проект семантического Web'a имеет скорее демонстрационный характер, но он действительно иллюстрирует осуществимость легкого построения приложений с помощью инструментального пакета семантического Web'a. Другим хорошим работающим примером является [RDFWeb](#), написанный Даном Брикли и др. (Dan Brickley et al.). RDFWeb это управляемая гиперсреда пространства блогов; сайт, на котором вся информация хранится в форме RDF, и которая затем используется для получения текста на XHTML. Планируется реализовать на этом сайте и более развитые принципы семантического Web'a.

Чем вы можете помочь?

Принять участие в построении семантического Web'a можно многими различными способами. Вот некоторые из них:

- Опубликуйте какие-либо глобально полезные данные на RDF;
- напишите программу логического вывода на любом выбранном вами языке;
- пропагандируйте обучение и распространение информации;
- примите участие в разработке схемы RDF и/или DAML;
- внесите свой вклад в представление состояния в RDF, несколько пренебрегая областью исследования;
- попробуйте применить свой собственный опыт разработки семантического Web'a, ознакомьте нас с новыми точками зрения.
- При построении нового приложения воспользуйтесь не своей привычной системой, а попробуйте создать новый проект на семантическом Web'e.

Есть также и много других путей оказания помощи: относительно деталей обратитесь в наше сообщество.

Что теперь? Читать

Что касается документа 2001-09, то объем материала, относящегося к обучению семантическому Web'у и его углубленному освоению, можно, безусловно, описать как "жалкий" (отсюда и это введение, для начала). Вот небольшой список нескольких *хороших* учебников и доступных на настоящий момент материалов, не упорядоченных по какому-то соображению:

- <http://www.w3.org/2000/10/swap/Primer> (Getting Into RDF & Semantic Web Using N3)
- <http://www.w3.org/DesignIssues/Semantic> (Semantic Web Roadmap)
- <http://purl.org/swag/whatIsSW> (What Is The Semantic Web?)
- <http://uwimp.com/eo.htm> (Semantic Web Primer)
- <http://logicerror.com/semanticWeb-long> (Semantic Web Introduction - Long)
- <http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html> (SciAm: The Semantic Web)
- <http://www.xml.com/pub/a/2001/03/07/buildingsw.html> (Building The Semantic Web)
- <http://infomesh.net/2001/06/swform/> (The Semantic Web, Taking Form)
- <http://www.w3.org/2001/sw/Activity> (SW Activity Statement)
- <http://www.w3.org/2000/01/sw/>

Дополнительную информацию, все последние новости и т.д. можно найти в совершенно прекрасной работе Дейва Беккета (Dave Beckett) [Resource Description Framework \(RDF\) Resource Guide](#)

Многие разработчики семантического Web'a и RDF общаются в чате RDF IG IRC на сайте irc.openprojects.net, #rdfig

Благодарности

Большое спасибо [Аарону Сварцу](#), Дану Брикли и Данни Айерс за их отзывы об этой статье и указания на целый ряд деталей, которые я упустил.
Sean B Palmer, 2001-09