Este documento es una traducción al Castellano de la nota del grupo de trabajo del W3C "RDFa 1.1 Primer", publicada el 7 de junio de 2012. La presente traducción se concluyó el 25 de enero de 2013.

La versión original en inglés es el único documento válido y se encuentra en:
http://www.w3.org/TR/2012/NOTE-rdfa-primer-20120607/
Puede ver la última versión del documento en inglés en: http://www.w3.org/TR/rdfa-primer/

Se ha tratado de respetar al máximo el contenido del documento original en inglés, adaptando la expresión al español para ayudar a una mejor comprensión del mismo. Por tanto, esta traducción puede contener errores, en ningún caso achacables a sus autores originales. Cualquier sugerencia de corrección, duda o comentario sobre la misma puede realizarse dirigiéndose a su autor: Juan Antonio Pastor Sánchez.

Manual de RDFa 1.1

Marcado enriquecido de datos estructurados para documentos web

Nota del Grupo de Trabajo del W3C de 7 de junio de 2012

Versión original (en Inglés):
http://www.w3.org/TR/2012/NOTE-rdfa-primer-20120607/
Última versión:
http://www.w3.org/TR/rdfa-primer/
Versión anterior:
http://www.w3.org/TR/2012/WD-rdfa-primer-20120508/
Versión 1.0 anterior de la nota:
http://www.w3.org/TR/2008/NOTE-xhtml-rdfa-primer-20081014/
Editores:
Ben Adida, Creative Commons, ben@adida.net
Ivan Herman, W3C, ivan@w3.org
Manu Sporny, Digital Bazaar, msporny@digitalbazaar.com
Mark Birbeck, webBackPlane.com, mark.birbeck@webBackplane.com
Traductor:
Juan Antonio Pastor Sánchez (Universidad de Murcia)

Consulte la sección de erratas en la que se pueden incluir algunas correcciones para este documento.

Resumen

Los últimos dos años han sido testigos de una evolución interesante: mientras que en un principio la Web se construyó principalmente para el consumo humano, su contenido es cada vez más utilizado por las máquinas que esperan una cierta cantidad de datos estructurados. Los sitios web han empezado a identificar el título de la página, tipo de contenido y la imagen de previsualización con el objeto de proporcionar la información adecuada para ofrecerla como fuente de noticias al usuario cuando este hace clic en el botón "Me gusta". Los motores de búsqueda han comenzado a ofrecer resultados más ricos mediante la obtención de información estructurada muy detallada de las páginas web que rastrean. A su vez, los editores web están produciendo cantidades cada vez mayores de datos estructurados dentro de su contenido web para mejorar el posicionamiento de los sitios que gestionan en los buscadores.

Una tecnología clave tras estos desarrollos es la capacidad de agregar directamente datos estructurados en páginas HTML. RDFa (Resource Description Framework in Attributes, RDF en atributos) es una técnica que permite precisamente esto: proporciona un conjunto de atributos de marcado para enriquecer la información visual en la web con datos legibles por máquinas. En este Manual, se muestra la forma de expresar datos utilizando RDFa dentro del lenguaje HTML, y en particular el modo para marcar contenidos web ya existentes para su lectura por parte de personas con el objeto de expresar anotaciones RDFa (hints) legibles por máquinas.

Este documento proporciona únicamente una introducción a RDFa 1.1. La especificación completa de RDFa, junto con algunos ejemplos, puede encontrarse en las especificaciones RDFa Core 1.1 [*RDFA-CORE*], RDFa Lite [*RDFA-LITE*], XHTML+RDFa 1.1 [*XHTML-RDFA*], y HTML5+RDFa 1.1 [*HTML-RDFA*].

Estado de este documento

Esta sección describe el estado de este documento en el momento de su publicación. Otros documentos pueden reemplazar a este documento. Una lista de las publicaciones vigentes del W3C y la última revisión de este informe técnico se puede encontrar en el índice de informes técnicos del W3C en http://www.w3.org/TR/.

Este documento fue publicado por el Grupo de Trabajo de aplicaciones web RDF como una nota de dicho grupo de trabajo. Los comentarios sobre este documento deben enviarse a public-rdfa@w3.org (subscribirse, archivos). Todos los comentarios son bienvenidos.

La publicación de una nota de un grupo de trabajo no implica la aprobación por los Miembros del W3C. Se trata de un borrador que puede ser actualizado, reemplazado o quedar obsoleto por otros documentos en cualquier momento. No es adecuado citar este documento como algo más que un trabajo en curso.

Este documento fue elaborado por un grupo que opera en el marco de la Política de Patentes del W3C del 5 de febrero 2004. El W3C mantiene una lista pública de cualquier patente divulgada, realizada en coordinación con la difusión de los resultados del grupo de trabajo; dicha página también incluye instrucciones para la divulgación de una patente. Una persona que tenga conocimiento de una patente que considere que contiene la(s) reivindicación (es) esencial(es) debe revelar la información de conformidad con la sección 6 de la Política de Patentes del W3C.

Tabla de contenido

1. Introducción

La web es un rico repositorio distribuido de información interconectada. Hasta hace poco, se organizaba pensado principalmente en su uso por parte de personas. En una página web típica, un autor HTML puede especificar un encabezamiento, a continuación un pequeño sub-encabezamiento, un bloque de texto en cursiva, algunos párrafos de texto de tamaño medio, y por último algunos enlaces de una sola palabra. Los navegadores web siguen estas instrucciones de presentación fielmente. Sin embargo, sólo la mente humana

entiende que el encabezamiento se refiere al título del artículo. El sub-encabezamiento indica el autor, el texto en cursiva la fecha de la publicación del mismo, y los enlaces de una sola palabra son categorías temáticas. Los ordenadores no entienden los matices entre la información; la brecha entre lo que entienden los programas y las personas es grande.

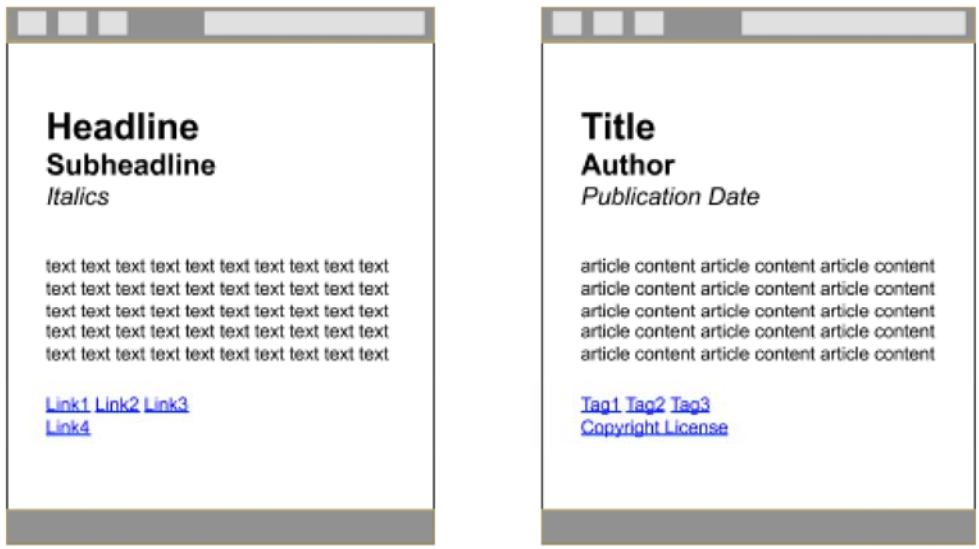


Figura 1: A la izquierda, lo que ven los navegadores. A la derecha, lo que ven las personas. ¿Es posible tender un puente entre lo que ven los navegadores y lo que vemos las personas?

¿Qué sucede si un navegador o cualquier máquina, como un rastreador web, obtiene información sobre el significado de los elementos visuales de una página web? Una cena anunciada en un blog podría copiarse en el calendario de un usuario y la información completa de contacto con el autor del blog podría añadirse a la libreta de direcciones de dicho usuario. Los usuarios de forma automática podrían recordar artículos previamente visitados según los rótulos de las categorías (como por ejemplo, etiquetas). Una foto copiada y pegada de un sitio web a un informe escolar llevaría consigo un enlace hacia el fotografo, lo que le otorgaría una acreditación apropiada de su autoría. Un enlace compartido por un usuario con sus contactos de redes sociales transportaría automáticamente datos adicionales extraídos de la página web original: una imagen en miniatura, un autor y un título específico. Cuando los datos web destinados a las personas se amplían con anotaciones destinadas a los programas de ordenador, estos se vuelven mucho más útiles, porque comienzan a comprender la estructura de los datos.

RDFa permite a los autores de HTML hacer precisamente esto. Utilizando algunos atributos HTML sencillos, los es posible marcar datos legibles para personas con indicaciones legibles por máquina para su interpretación por navegadores y otros programas. Una página web puede incluir un marcado para ítems tan simple como el título de un artículo o tan complejo como la red social completa de un usuario.

1.1 HTML frente a XHTML

Historicamente, RDFa 1.0 [*RDFA-SYNTAX*] se especificaba únicamente para XHTML. RDFa 1.1 [*RDFA-CORE*] es la nueva versión y la única usada en este documento. RDFa 1.1 se especifica para ambos lenguajes, XHTML [*XHTML-RDFA*] y HTML5 [*HTML-RDFA*]. De hecho, RDFa 1.1 también funciona para cualquier lenguaje basado en XML como SVG [*SVG11*]. Este documento utiliza HTML en todos los ejemplos; para simplificar, se ha utilizado el término "HTML" a lo largo de este documento para referirse a toda la familia de lenguajes HTML.

1.2 Validación

RDFa se basa en atributos. Mientras algunos de los atributos de HTML (por ejemplo, `href`, `src`) han sido reutilizados, otros atributos RDFa son nuevos. Esto es importante puesto que algunos de los validadores (X)HTML pueden que no validen apropiadamente el código HTML hasta que se actualicen para reconocer los nuevos atributos RDFa. En la práctica, este es un problema poco frecuente puesto que los navegadores simplemente ignoran los atributos que no reconocen. Ninguno de los atributos específicos de RDFa tienen efecto alguno en la presentación visual del contenido HTML. Los autores no han de preocuparse de que las páginas marcadas con RDFa muestren alguna diferencia para las personas con respecto a las que no lo han sido.

2. Usando RDFa

2.1 Aspectos básicos de RDFa: RDFa Lite

A continuación se procede a una introducción a RDFa utilizando un subconjunto de todas sus posibilidades denominado RDFa Lite 1.1 [*RDFA-LITE*]. El objetivo que se persigue al definir dicho subconjunto, es disponer de un conjunto de posibilidades que puedan aplicarse desde la más simple de las tareas de marcado de datos estructurados hasta otras moderadamente más avanzadas, sin sobrecargar a los autores con aspectos complejos adicionales. Muchos autores web no necesitarán utilizar más que este subconjunto mínimo.

2.1.1 Primeros pasos: añadiendo datos legibles por máquina a páginas web

Consideremos a Alice, una blogger que publica una mezcla de articulos profesionales y personales en `http://example.com/alice`. Se construirán ejemplos de marcado para ilustrar como Alice puede usar RDFa. El marcado completo de esos ejemplos está disponible en una página dedicada.

2.1.1.1 Anotaciones RDFa (hints) en sitios web de redes sociales

Alice publica un blog y le gustaría proporcionar información estructurada adicional en sus páginas, como la fecha de publicación o el título. A ella le gustaría utilizar los términos definidos en el vocabulario Dublin Core [*DC11*], un conjunto de términos ampliamente utilizados, por ejemplo, en la industria editorial o bibliotecas. Su blog ya contiene dicha información:

Ejemplo

```
<html>
<head>
  ...
</head>
<body>
  ...
  <h2>The Trouble with Bob</h2>
  <p>Date: 2011-09-10</p>
  ...
</body>
```

Sin embargo, esta información está pensada únicamente para las personas; los ordenadores necesitan algunos métodos sofisticados para extraerla. Sin embargo, utilizando RDFa, ella puede marcar su página para expresar *datos estructurados*:

Ejemplo

```
<html>
<head>
  ...
</head>
<body>
  ...
  <h2 property="http://purl.org/dc/terms/title">The Trouble with Bob
  <p>Date: <span property="http://purl.org/dc/terms/created">2011-0
  ...
</body>
```

(Nótese el marcado en color rojo: esas son las anotaciones ("hints") RDFa.

Un modo útil de visualizar los datos estructurados sería:

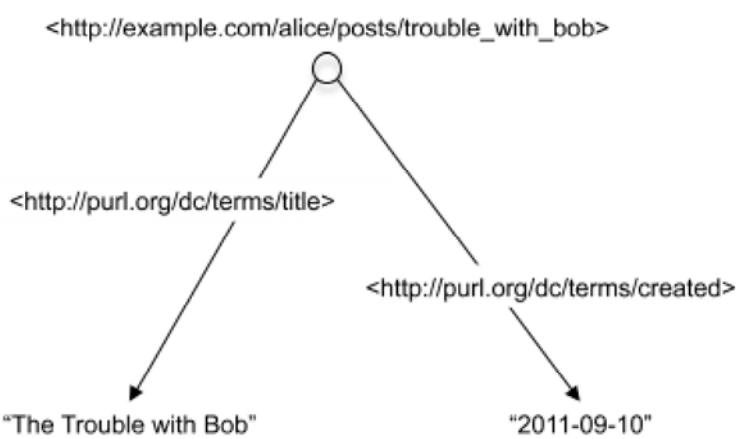


Figura 2: Una visualización de los datos estructurados para una entrada en un blog con el título "The Trouble with Bob" y su fecha de creación.

Vale la pena destacar que RDFa utiliza URLs para identificar casi todo. Por esta razón, en lugar de utilizar propiedades como `title` o `created`, se usa `http://purl.org/dc/terms/title` y `http://purl.org/dc/terms/created`. La razón tras esta decisión de diseño se basa en la portabilidad de los datos, la

consistencia y el intercambio de información. El uso de URLs elimina la posibilidad de ambigüedades en la terminología. Sin asegurar que no hay ambigüedad, el término "title" puede significar "el título de una obra", "el título de un puesto de trabajo", o "la escritura de una propiedad inmobiliaria". Cuando cada término del vocabulario es una URL, está disponible a un clic de distancia una detallada explicación del significado del mismo. Esto permite que tanto personas como máquinas puedan seguir el enlace para encontrar el significado de un término concreto de un vocabulario. Mediante el uso de una URL para identificar un tipo de título en particular, por ejemplo `http://purl.org/dc/terms/created`, tanto las personas como las máquinas pueden entender que dicha URL se refiere de forma inequívoca a la "Fecha de creación de un recurso", tal como una página web.

Mediante el uso de direcciones URL como identificadores, RDFa proporciona una forma sólida para desambiguar términos de vocabularios. De este modo se vuelve trivial la tarea de determinar cuando los términos utilizados en diferentes documentos significan lo mismo o tienen significados distintos. Si las direcciones URL son las mismas, los términos del vocabulario significan lo mismo. También se hace muy fácil crear nuevos términos y documentos de vocabularios. Si se puede publicar un documento en la Web, automáticamente se tiene la capacidad para crear un documento que contiene un nuevo vocabulario con nuevos términos.

2.1.1.2 Enlaces con Sabor

El ejemplo anterior muestra como Alice puede marcar el texto para hacerlo legible por máquina. A ella también le gustaría marcar los enlaces de forma que sean legible por máquina, para expresar el tipo de enlace que está siendo descrito. RDFa permite al editor añadir un "sabor", por ejemplo una etiqueta, a un enlace sobre el que se puede hacer clic, y que puedan entender las aplicaciones informáticas. Esto hace que el mismo marcado sea útil tanto para las personas como para las máquinas.

En el pié de página de su blog, Alice ya ha declarado que su contenido puede ser reutilizado libremente, siempre y cuando ella reciba la debida acreditación de su autoría cuando sus artículos sean citados. El código HTML incluye un enlace a una licencia Creative Commons [*CC-ABOUT*]:

Ejemplo

```
<p>All content on this site is licensed under
   <a href="http://creativecommons.org/licenses/by/3.0/">
     a Creative Commons License</a>. ©2011 Alice Birpemswick.</p>
```

Una persona entiende claramente esta frase, en particular el *significado* del enlace con respecto al documento actual: indica la licencia del documento, las condiciones bajo las que puede distribuirse el contenido de la página. Desafortunadamente, cuando Bob visita el blog de Alice su navegador únicamente ve un enlace plano que podría apuntar tanto a uno de los amigos de Alice como a su currículum vítae. Para que el navegador de Bob entienda que dicho enlace actualmente apunta a una página con los términos de la

licencia del documento, Alice necesita añadir algún *sabor*, alguna indicación al respecto del tipo de enlace del que se trata.

Alice puede añadir este sabor usando otra vez el atributo `property`. En efecto, cuando el elemento contiene el atributo `href` (o `src`), `property` se asocia automáticamente con el valor de este atributo en lugar del contenido textual del elemento `a`. El valor del atributo es el identificador URL `http://creativecommons.org/ns#license`, definido por Creative Commons:

Ejemplo

```
<p>All content on this site is licensed under
   <a property="http://creativecommons.org/ns#license" href="http:/,
     a Creative Commons License</a>. ©2011 Alice Birpemswick.</p>
```

Con este pequeño cambio, el navegador de Bob comprenderá ahora que este enlace tiene un sabor: indica la licencia del blog:

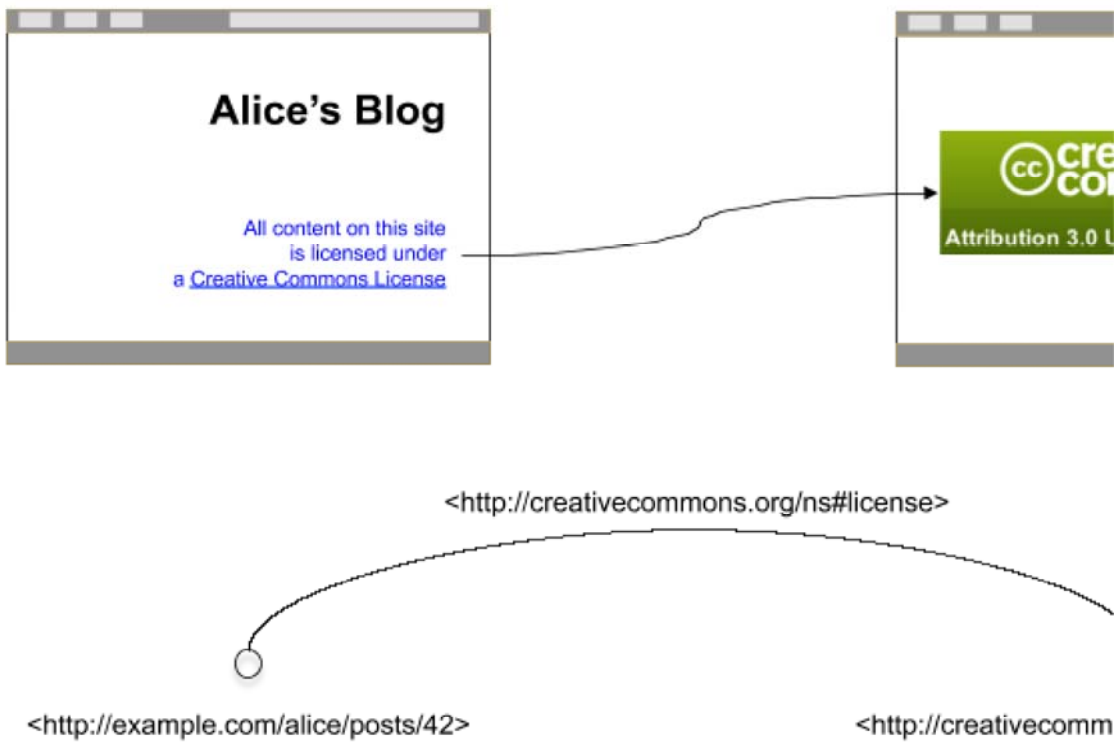


Figura 3: Un enlace con sabor: el enlace indica la licencia de la página web. Las páginas web pueden representarse como nodos, el enlace es un arco (flecha) que conecta dichos nodos, y el sabor del enlace es la etiqueta de la flecha.

Alice está muy agradecida de haber podido añadir únicamente las anotaciones ("hints") de datos estructurados mediante RDFa, sin que tuviera que repetir el contenido del texto o las referencias URL de los enlaces.

2.1.1.3 Estableciendo un vocabulario por defecto

En una serie de casos de uso sencillos, como en el ejemplo del blog de Alice, los autores HTML normalmente utilizarán únicamente un vocabulario. Sin

embargo, mientras que la tarea para generar direcciones URL completas a través de sistemas de gestión de contenidos (Content Management Systems, CMS) no reviste problema alguno, hacerlo a mano puede ser tedioso y propenso a errores por parte de las personas. Para paliar este problema RDFa introduce el atributo `vocab` que permite que el autor declare un vocabulario único para un fragmento de código HTML. Así, en lugar de:

Ejemplo

```
<html>
<head>
  ...
</head>
<body>
  ...
  <h2 property="http://purl.org/dc/terms/title">The Trouble with Bob
  <p>Date: <span property="http://purl.org/dc/terms/created">2011-09
  ...
</body>
```

Alice puede escribir:

Ejemplo

```
<html>
<head>
  ...
</head>
<body vocab="http://purl.org/dc/terms/">
  ...
  <h2 property="title">The Trouble with Bob</h2>
  <p>Date: <span property="created">2011-09-10</span></p>
  ...
</body>
```

Nótese cómo los valores de las propiedades individuales son ahora "términos", los cuales simplemente se concatenan a la URL definida mediante el atributo `vocab`. El atributo se puede colocar en cualquier elemento HTML (es decir, no sólo en el elemento `body` como en el ejemplo) y su efecto es válido para todos los elementos por debajo de ese punto.

Los vocabularios por defecto y las URIs completas pueden combinarse en cualquier momento. Es decir, Alice podría escribir:

Ejemplo

```
<html>
<head>
  ...
</head>
<body vocab="http://purl.org/dc/terms/">
  ...
  <h2 property="title">The Trouble with Bob</h2>
  <p>Date: <span property="http://purl.org/dc/terms/created">2011-09
  ...
</body>
```

Tal vez un ejemplo más interesante de lo anterior es la combinación de los encabezamientos con el fragmento de la licencia de la página web:

Ejemplo

```
<html>
<head>
  ...
</head>
<body vocab="http://purl.org/dc/terms/">
  ...
  <h2 property="title">The Trouble with Bob</h2>
  <p>Date: <span property="created">2011-09-10</span></p>
  ...
  <p>All content on this site is licensed under
   <a property="http://creativecommons.org/ns#license" href="http:/
     a Creative Commons License</a>. ©2011 Alice Birpemswick.</p>
</body>
</html>
```

Es necesario utilizar la URL completa de la licencia para evitar que se mezclen vocabularios. Como alternativa, Alice podría haber escogido utilizar de nuevo el atributo `vocab`:

Ejemplo

```
<html>
<head>
  ...
</head>
<body vocab="http://purl.org/dc/terms/">
  ...
  <h2 property="title">The Trouble with Bob</h2>
  <p>Date: <span property="created">2011-09-10</span></p>
  ...
  <p vocab="http://creativecommons.org/ns#">All content on this site
    <a property="license" href="http://creativecommons.org/licenses/
      a Creative Commons License</a>. ©2011 Alice Birpemswick.</p>
</body>
</html>
```

ya que el atributo `vocab` en el párrafo de la licencia anula la definición heredada del cuerpo del documento.

2.1.1.4 Multiples Items por página

Por supuesto, la página del blog de Alice puede contener múltiples entradas. A veces Eve, la hermana de Alice, también introduce alguna entrada en el blog. En la página principal del blog se enumeran las 10 entradas más recientes, cada una con su propio título, autor y párrafo introductorio. ¿Cómo debería marcar Alice el título de cada una de las entradas de forma individual pese a que todas aparecen en la misma página? RDFa proporciona el atributo `resource` para especificar el "contexto", es decir, la dirección exacta a la que se aplica el marcado RDFa:

Ejemplo

```
<body vocab="http://purl.org/dc/terms/">
   ...
   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      <p>Date: <span property="created">2011-09-10</span></p>
      <h3 property="creator">Alice</h3>
      ...
   </div>
   ...
   <div resource="/alice/posts/jos_barbecue">
      <h2 property="title">Jo's Barbecue</h2>
      <p>Date: <span property="created">2011-09-14</span></p>
      <h3 property="creator">Eve</h3>
      ...
   </div>
   ...
</body>
```

(Nótese que se usan URLs relativas en el ejemplo; el valor de `resource` podrían haber sido de *cualquier* tipo, relativas o absolutas). Nuevamente, puede representarse esto como un diagrama que conecta las URLs con las propiedades:

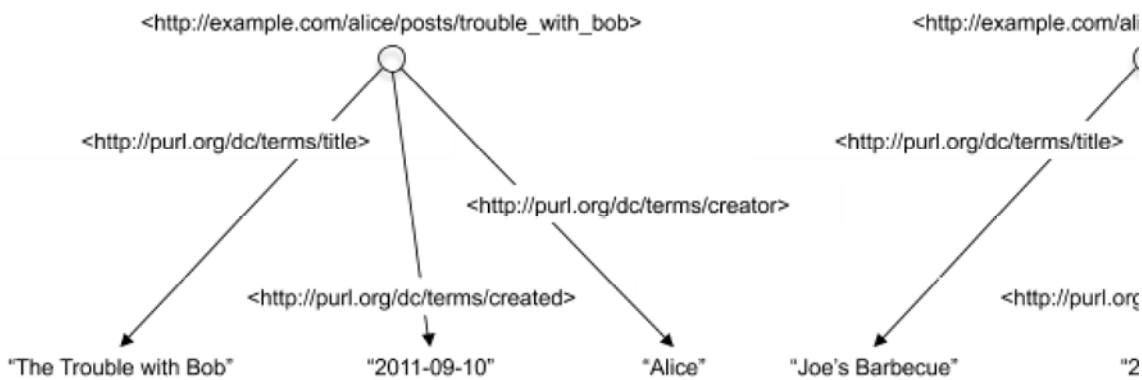


Figura 4: Múltiples Items por página: cada entrada del blog se representa mediante un nodo, con propiedades asociadas al mismo.

Alice puede utilizar la misma técnica para acreditar apropiadamente a Bob como autor de las fotografías que ella suele incluir en las entradas:

Ejemplo

```
<div resource="/alice/posts/trouble_with_bob">
    <h2 property="title">The trouble with Bob</h2>
    ...
    The trouble with Bob is that he takes much better photos than I
    ...
    <div resource="http://example.com/bob/photos/sunset.jpg">
      <img src="http://example.com/bob/photos/sunset.jpg" />
      <span property="title">Beautiful Sunset</span>
      by <span property="creator">Bob</span>.
    </div>
 </div>
```

Observese como el valor del atributo `resource` más interior, `http://example.com/bob/photos/sunset.jpg`, "sobreescribe" el valor más externo `/alice/posts/trouble_with_bob` para todo el marcado dentro del contenedor `div`. Una vez más, se incluye un diagrama que representa los datos incluidos en este fragmento de marcado:

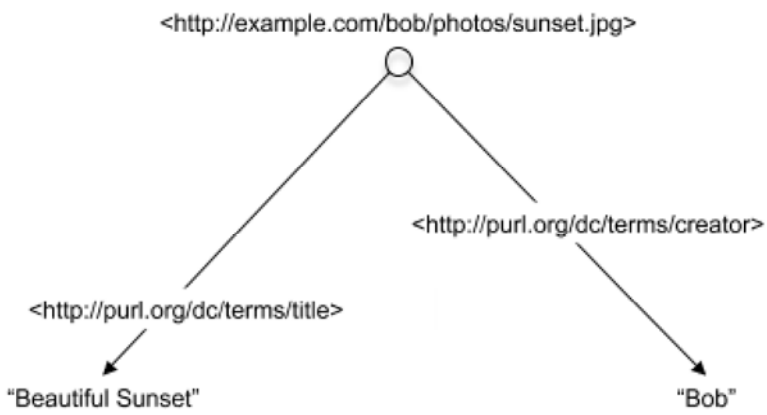


Figura 5: Describiendo una fotografía

2.1.2 Explorando a fondo: Redes sociales

2.1.2.1 Información de contacto

A Alice también le gustaría que la información sobre sí misma, como su dirección de correo electrónico, número de teléfono y otros detalles, esté fácilmente disponible para el software de gestión de contactos de sus amigos. En esta ocasión, en vez de describir las propiedades de una página web, va a describir las propiedades de una persona: ella misma.

Alice ya dispone de información de contacto visible en su blog.

Ejemplo

```
<div>
  <p>
     Alice Birpemswick,
     Email: <a href="mailto:alice@example.com">alice@example.com</a>
     Phone: <a href="tel:+1-617-555-7332">+1 617.555.7332</a>
  </p>
</div>
```

El vocabulario Dublin Core no proporciona nombres de propiedades para describir la información de contacto, pero si lo hace el vocabulario Friend-of-a-Friend [*FOAF*]. Alice decide utilizar el vocabulario FOAF. Como primer paso, se declara una "Persona" FOAF. Para ello Alice utiliza el atributo RDFa `typeof`, que está específicamente pensado para declarar un nuevo item de datos de un tipo concreto:

Ejemplo

```
<div typeof="http://xmlns.com/foaf/0.1/Person">
  ...
```

Alice se da cuenta de que ella sólo tiene la intención de utilizar el vocabulario FOAF en este punto, por lo que utiliza el atributo `vocab` para simplificar el marcado adicional (anulando los efectos de cualquier atributo `vocab` que pueda haber sido utilizado previamente en, por ejemplo, el elemento `body`.

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/" typeof="Person">
  ...
```

Entonces, Alice indica el contenido de la página que representa su nombre completo, su dirección de correo electrónico y su número de teléfono:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/" typeof="Person"><p>
  <p>
    <span property="name">Alice Birpemswick</span>,
    Email: <a property="mbox" href="mailto:alice@example.com">alice(
    Phone: <a property="phone" href="tel:+1-617-555-7332">+1 617.55
  </p>
</div>
```

Debe tenerse en cuenta que Alice no ha especificado un `resource` como hizo cuando añadió los metadatos a la entrada del blog. Pero si Alice no declara aquello sobre lo que se está hablando ¿cómo puede conocer el procesador RDFa a qué se está refiriendo? En RDFa, en ausencia de un atributo `resource`, el atributo `typeof` del `div` que engloba el fragmento de código anterior establece implícitamente el sujeto de las propiedades marcadas en ese `div`. Es decir, el nombre, la dirección de correo electrónico y el número de teléfono están asociados con un nuevo nodo de tipo "Person". Este nodo no tiene URL que lo identifique, y se denomina *nodo en blanco* como se muestra en la figura:

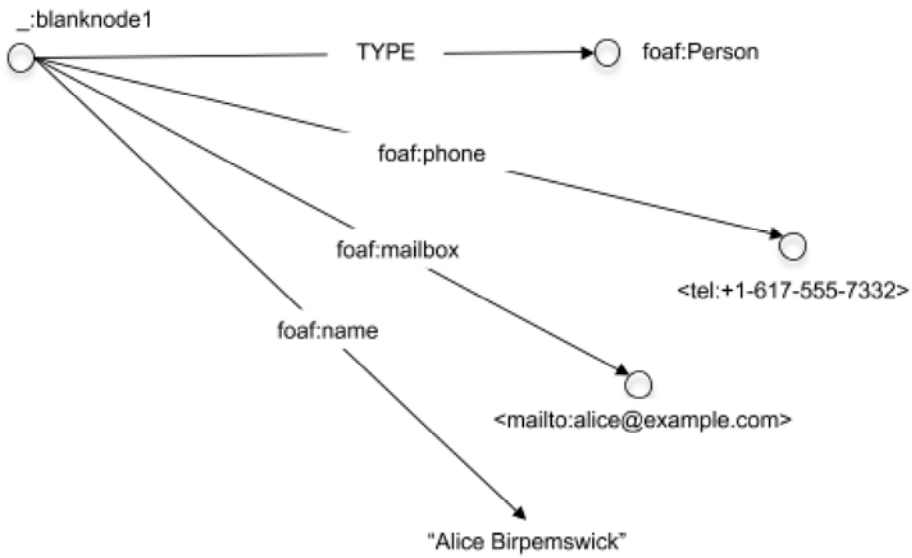


Figura 6: Un nodo en blanco: los nodos en blanco no se identifican mediante una URL. En cambio, muchos de ellos tienen un atributo RDFa `typeof` que identifica el tipo de dato que representan.
(Se ha utilizado la forma abreviada para etiquetar las flechas, con el objeto de ahorrar espacio y hacer más claro el diagrama. Las etiquetas reales son siempre las URLs completas.)

2.1.2.2 Describiendo redes sociales

Alice continua marcando su página añadiendo información sobre sus amigos, indicando al menos sus nombres y páginas web personales. Para ello, parte del siguiente código HTML:

Ejemplo

```
<div>
   <ul>
      <li>
        <a href="http://example.com/bob/">Bob</a>
      </li>
      <li>
        <a href="http://example.com/eve/">Eve</a>
      </li>
      <li>
        <a href="http://example.com/manu/">Manu</a>
      </li>
   </ul>
</div>
```

Primero, Alice indica que los amigos que está describiendo son personas, a diferencia de animales o amigos imaginarios, usando una vez más `Person` dentro de los correspondientes atributos `typeof`.

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/">
   <ul>
      <li typeof="Person">
        <a href="http://example.com/bob/">Bob</a>
      </li>
      <li typeof="Person">
        <a href="http://example.com/eve/">Eve</a>
      </li>
      <li typeof="Person">
        <a href="http://example.com/manu/">Manu</a>
      </li>
   </ul>
</div>
```

Más allá de declarar el tipo de datos que nos ocupa, cada `typeof` crea un nuevo nodo en blanco con sus propiedades individuales. Por lo tanto, Alice puede indicar la página web personal de cada amigo:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/">
   <ul>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/bob/">Bob</a
      </li>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/eve/">Eve</a
      </li>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/manu/">Manu<
      </li>
   </ul>
</div>
```

A Alice también le gustaría mejorar el marcado, expresando además el nombre de cada persona mediante RDFa. Esto puede hacerse añadiendo un elemento `span` separado con su correspondiente atributo `property`:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/">
  <ul>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/bob/"><span
      </li>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/eve/"><span
      </li>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/manu/"><spar
      </li>
   </ul>
</div>
```

Alice se alegra de que con tan poco marcado adicional haya podido expresar plenamente y a la vez una página de lectura agradable para las personas y un conjunto de datos legible por máquina.

Alice es miembro de 5 sitios diferentes de redes sociales. Está cansada de introducir información repetida sobre sus amigos en cada red social nueva, por lo que decide enumerar a sus amigos en un único lugar de su sitio web, combinándolo con sus propios datos FOAF. Con RDFa, puede indicar esto a sus amistades en su propia página web y permitir que los sitios de las redes sociales lean automáticamente esta información. Hasta el momento, Alice ha enumerado tres personas pero no ha especificado su relación con ellos; podrían ser sus amigos o sus poetas favoritos del siglo XVII. Para indicar que los conoce utiliza la propiedad FOAF `foaf:knows`:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/" typeof="Person">
   <p>
    <span property="name">Alice Birpemswick</span>,
    Email: <a property="mbox" href="mailto:alice@example.com">alice(
    Phone: <a property="phone" href="tel:+1-617-555-7332">+1 617.555
  </p>
   <ul>
      <li property="knows" typeof="Person">
        <a property="homepage" href="http://example.com/bob/"><span
      </li>
      <li property="knows" typeof="Person">
        <a property="homepage" href="http://example.com/eve/"><span
      </li>
      <li property="knows" typeof="Person">
        <a property="homepage" href="http://example.com/manu/"><spar
      </li>
   </ul>
</div>
```

Con esto, Alice describiría la siguiente red social:

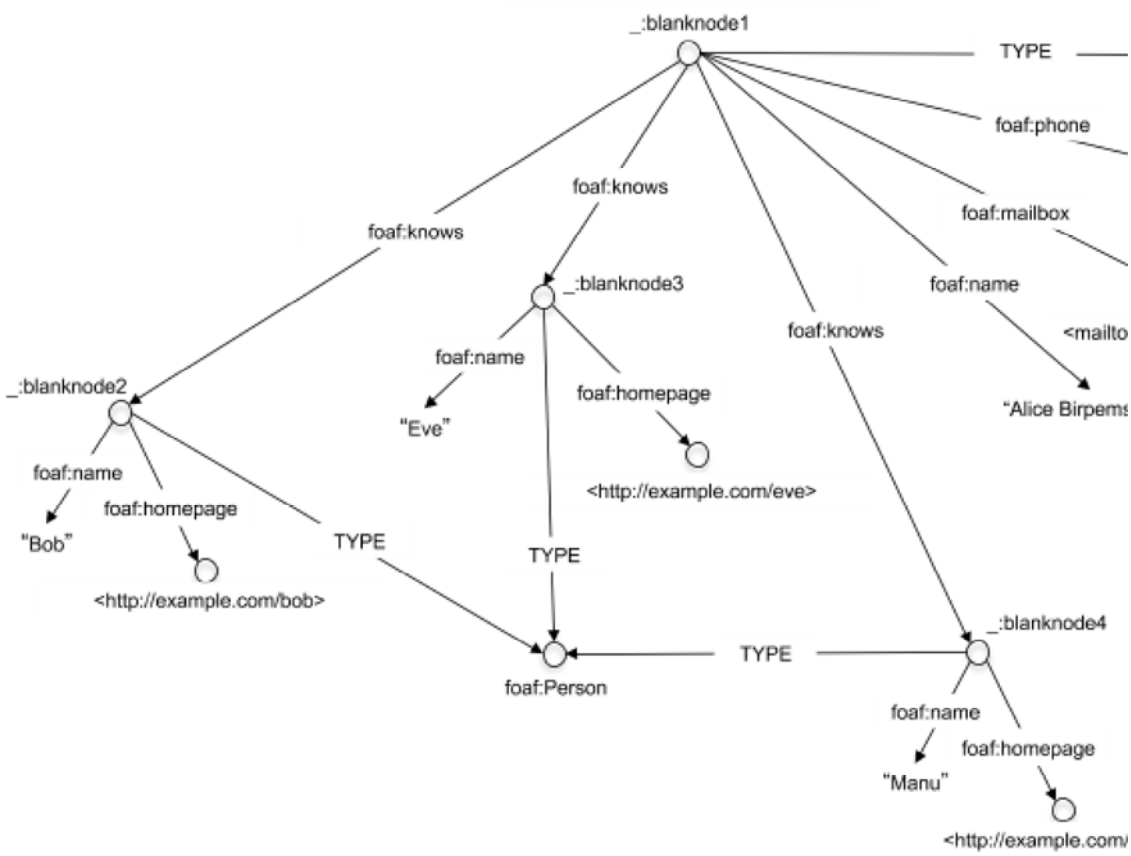


Figura 7: La red social de Alice. Obsérvese que, con RDFa, Alice podría expresar un conjunto de información bastante compleja para que otros usuarios puedan utilizarlo.

2.1.3 Referencias internas

Alice puede desear añadir sus datos personales a los items individuales del blog. Decide combinar sus datos FOAF con los items del blog, es decir:

Ejemplo

```
<div vocab="http://purl.org/dc/terms/">

   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      ...
      <h3 vocab="http://xmlns.com/foaf/0.1/" property="http://purl.
        <span property="name">Alice Birpemswick</span>,
        Email: <a rel="mbox" href="mailto:alice@example.com">alice@e
        Phone: <a rel="phone" href="tel:+1-617-555-7332">+1 617.555.
      </h3>
      ...
   </div>
   ...
</div>
```

Los datos estructurados que ella genera serían los siguientes:

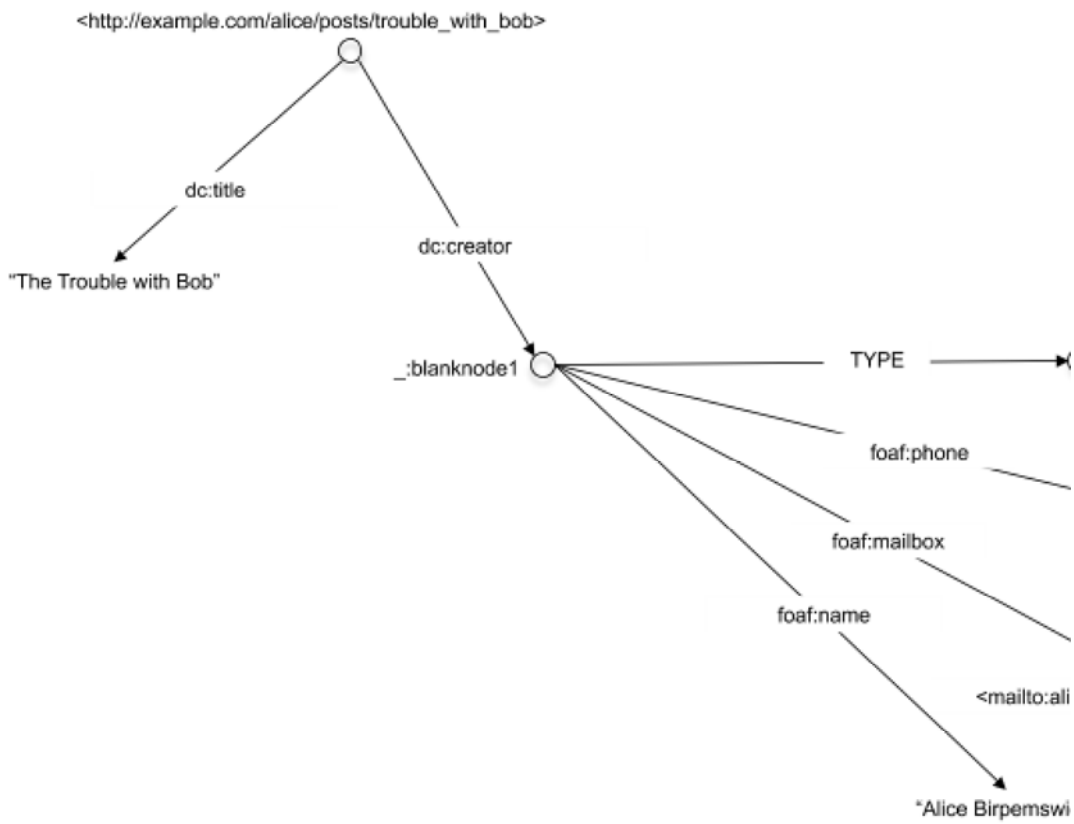


Figura 8: Item del blog de Alice con datos sobre ella misma.

Desgraciadamente, esta solución no es óptima por dos motivos. En primer lugar, Alice tuvo que usar la URI completa de la propiedad `creator`: esto es así debido a que el atributo `vocab` se utiliza para establecer los términos FOAF, es decir, haber usado únicamente el término de `creator` sin prefijo habría sido mal interpretado. Se volverá a tratar el uso de varios vocabularios en otra sección más adelante.

El otro motivo es que a Alice le gustaría diseñar su página Web para que sus datos personales no aparezcan en la página de cada artículo individual del blog, sino más bien en un lugar como una nota al pie o una barra lateral. Es decir, lo que le gustaría ver sería algo así como:

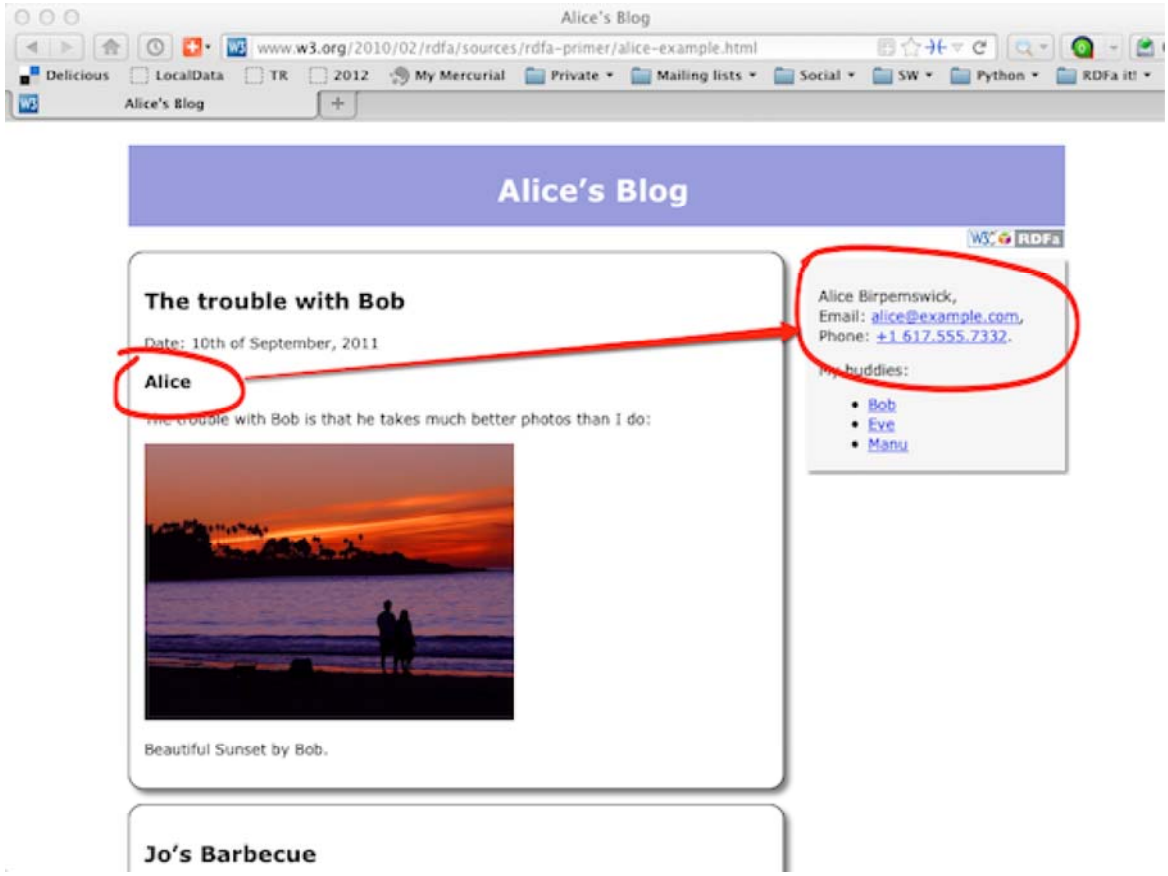


Figura 9: Estructura del sitio web de Alice: items individuales del blog a la izquierda, datos personales, enlazados desde el blog utilizando términos RDFa, en una barra lateral a la derecha.

Si los datos FOAF se incluyen en cada item del blog, Alice tendría que crear un conjunto complejo de reglas CSS para obtener el efecto visual que ella desea.

Para solucionar esto, Alice decide hacer uso de la estructura que ya se utiliza para sus datos FOAF, pero esta vez le asigna una URI independiente mediante el atributo de `resource`:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#me" typeof="Pers
  <p>
   <span property="name">Alice Birpemswick</span>,
     Email: <a property="mbox" href="mailto:alice@example.com">alic
     Phone: <a property="phone" href="tel:+1-617-555-7332">+1 617.5
  </p>
  ...
</div>
```

En realidad, se considera una buena práctica utilizar URIs reales siempre que sea posible, es decir, generalmente es preferible esta nueva alternativa utilizada por Alice. En efecto, si se utiliza una URI real, entonces es posible hacer una referencia clara a esa pieza particular de información, mientras que se vuelve más complicado hacerlo mediante el uso de nodos en blanco.

Note

El marcado `resource="#me"` es una convención FOAF: la URL que representa a la persona Alice es `http://example.com/alice#me`. No debe confundirse con la página web de Alice, `http://example.com/alice`. Por supuesto, Alice podría haber usado un URI diferente si, por ejemplo, su blog y su página personal se mantuvieran separadas, por ejemplo, se podría haber utilizado `resource="http://alice.example.com/alice/home#myself"` en lugar de `resource="#me"`.

Usando una URI explícita para sus datos FOAF, Alice puede añadir una referencia directa al elemento del blog utilizando el atributo `resource`:

Ejemplo

```
<div vocab="http://purl.org/dc/terms/">
   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      <h3 property="creator" resource="#me">Alice</h3>
      ...
   </div>
</div>
   ...
<div class="sidebar" vocab="http://xmlns.com/foaf/0.1/" resource="#r
  <p>
   <span property="name">Alice Birpemswick</span>,
     Email: <a property="mbox" href="mailto:alice@example.com">alice
     Phone: <a property="phone" href="tel:+1-617-555-7332">+1 617.55
  </p>
  ...
</div>
```

El atributo `resource` aparece, en este caso, junto con el atributo `property` *en el mismo elemento*: en esta situación `resource` indica el "destino" u "objeto" de la relación. El uso de este atributo permite a Alice "distribuir" varias partes de sus datos estructurados en su página. Lo que obtiene es una versión ligeramente modificada de la estructura anterior, donde la única diferencia es el uso de una URI explícita en vez de un nodo en blanco:

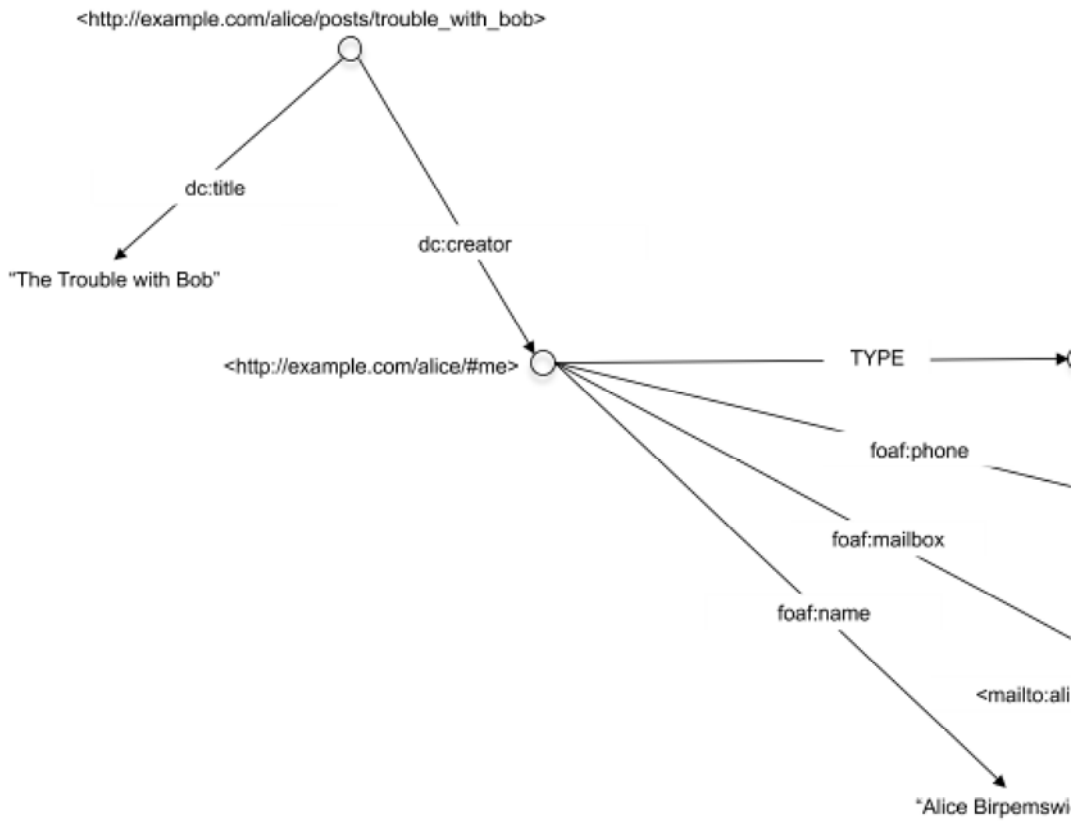


Figura 10: Item del blog de Alice con datos sobre ella misma, usando una URI explícita en sus datos FOAF.

Utilizando este enfoque, es muy fácil añadir también referencias a los *mismos* datos de diferentes blogs:

Ejemplo

```
<div vocab="http://purl.org/dc/terms/">
   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      <h3 property="creator" resource="#me">Alice</h3>
      ...
   </div>
</div>
   ...
<div vocab="http://purl.org/dc/terms/">
   <div resource="/alice/posts/my_photos">
      <h2 property="title">I will post my photos nevertheless…</h2>
      <h3 property="creator" resource="#me">Alice</h3>
      ...
   </div>
</div>
   ...
<div class="sidebar" vocab="http://xmlns.com/foaf/0.1/" resource="#r
  <p>
   <span property="name">Alice Birpemswick</span>,
     Email: <a property="mbox" href="mailto:alice@example.com">alice
     Phone: <a property="phone" href="tel:+1-617-555-7332">+1 617.55
  </p>
```

```
  ...
</div>
```

Obteniendo la siguiente estructura:

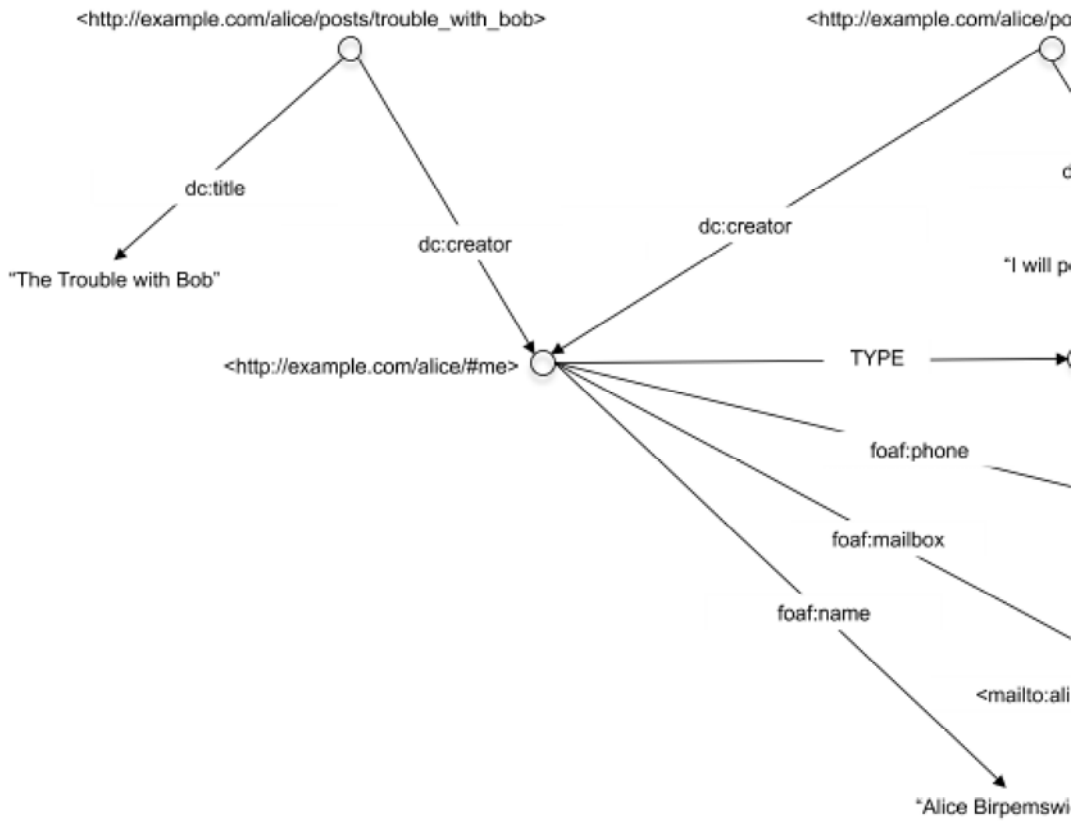


Figura 11: Algunos items del blog de Alice con datos sobre ella misma, usando una URI explícita para sus datos FOAF..

Note

Combinado con `property`, el atributo `resource` desempeña exactamente el mismo papel que `href`, utilizado anteriormente para los enlaces con "sabor" ("flavor"), con la excepción de que no proporciona al navegador un enlace sobre el que se pueda hacer clic tal y como hace `href`. El atributo `resource` también puede utilizarse en cualquier elemento HTML, al contrario de `href` cuyo uso está restringido a los elementos HTML `a` y `link`.

2.1.4 Usando múltiples vocabularios

Los ejemplos anteriores muestran que en casos más complejos han de usarse múltiples vocabularios para expresar la diversidad de aspectos de los datos estructurados. Se ha visto como Alice ha utilizado los vocabularios Dublin Core, así como FOAF y Creative Commons, pero pueden haber más. Por

ejemplo, Alice puede desear añadir elementos del vocabulario definido por los motores de búsqueda del sitio web schema.org [*SCHEMA*].

Alice puede utilizar las direcciones URLs completas de todos los términos, o puede utilizar el atributo `vocab` para abreviar los términos del vocabulario predominante. Pero, en algunos casos, los vocabularios no pueden ser separados fácilmente, lo que significa que el uso del atributo `vocab` puede ser incómodo. Como ejemplo sirva el siguiente marcado HTML que podría realizar Alice:

Ejemplo

```
<html>
 <head>
    ...
 </head>
 <body vocab="http://schema.org/">
   <div resource="/alice/posts/trouble_with_bob" typeof="BlogPostinç
      <h2 property="http://purl.org/dc/terms/title">The trouble witl
      ...
      <h3 property="http://purl.org/dc/terms/creator" resource="#me'
      <div property="articleBody">
        <p>The trouble with Bob is that he takes much better photos
      </div>
      ...
   </div>
   ...
  </body>
 </html>
```

Debe observarse que los términos de schema.org y Dublin Core Note se entrelazan para un blog específico, siendo una elección arbitraria cuando usar el atributo `vocab` para `http://purl.org/dc/terms/` o `http://schema.org/`. Se ha visto el mismo problema en una sección previa cuando se mezclaron los términos de FOAF y Dublin Core.

Para evitar este problema, RDFa ofrece la posibilidad de utilizar los términos con prefijo: un atributo especial `prefix` puede asignar prefijos para representar URLs y, utilizando dichos prefijos, pueden abreviarse los elementos del vocabulario. Se utiliza la sintaxis `prefix:reference`: simplemente se concatena la dirección URL asociada con `prefix` a `reference` para crear una URL completa. (Ha de tenerse en cuenta que ya se ha utilizado esta convención para simplificar las figuras de este documento). A continuación se muestra el código HTML del ejemplo utilizando los prefijos:

Ejemplo

```
<html>
 <head>
   ...
 </head>
 <body prefix="dc: http://purl.org/dc/terms/ schema: http://schema.c
   <div resource="/alice/posts/trouble_with_bob" typeof="schema:Bloç
      <h2 property="dc:title">The trouble with Bob</h2>
      ...
      <h3 property="dc:creator" resource="#me">Alice</h3>
      <div property="schema:articleBody">
        <p>The trouble with Bob is that he takes much better photos
```

```
      </div>
      ...
    </div>
  </body>
</html>
```

El uso de prefijos puede reducir en gran medida los posibles errores al concentrar las referencias a los vocabularios en un lugar del fichero. Al igual que `vocab`, el atributo `prefix` puede aparecer en cualquier lugar en el fichero HTML, afectando únicamente a los elementos posteriores. `prefix` y `vocab` también pueden mezclarse, por ejemplo:

Ejemplo

```
<html>
 <head>
   ...
 </head>
 <body vocab="http://purl.org/dc/terms/" prefix="schema: http://sche
   <div resource="/alice/posts/trouble_with_bob" typeof="schema:Bloc
      <h2 property="title">The trouble with Bob</h2>
      ...
      <h3 property="creator" resource="#me">Alice</h3>
      <div property="schema:articleBody">
        <p>The trouble with Bob is that he takes much better photos
      </div>
      ...
   </div>
 </body>
</html>
```

Note

Una cuestión importante puede surgir si el elemento `html` contiene un gran número de declaraciones de prefijo. La codificación de caracteres (UTF-8, UTF-16, ASCII, etc) que se utiliza para un fichero HTML5 se declara utilizando un elemento `meta` en la cabecera. En HTML5 la declaración meta debe estar dentro de los primeros 512 bytes de la página, o el procesador de HTML5 (navegador, parser, etc) tratará de detectar la codificación usando algún tipo de proceso heurístico. Por lo tanto, una etiqueta `html` muy "larga" puede dar lugar a problemas. Una forma de evitarlo es colocar la mayor parte de las declaraciones de prefijo en el elemento `body`.

2.1.4.1 Repetición de propiedades

En el ejemplo anterior, en el que los vocabularios Dublin Core y schema.org se utilizan dentro de la misma entrada del blog, se plantea otra cuestión. Se da la circunstancia de que tanto Dublín Core como schema.org tienen una propiedad llamada `creator`. Debido a que RDFa utiliza URIs para referirse a las propiedades esto no supone un problema. Sin embargo, si Alice quiere usar ambas propiedades en la misma entrada del blog (por ejemplo, porque quiere que los motores de búsqueda indexen su entrada en el blog, pero al mismo

tiempo quiere que aplicaciones Dublin Core, como catálogos, recopilen las entradas de su blog) esto es lo que debería hacer:

Ejemplo

```
<html>
 <head>
   ...
 </head>
 <body prefix="dc: http://purl.org/dc/terms/ schema: http://schema.c
   <div resource="/alice/posts/trouble_with_bob" typeof="schema:Blog
      <h2 property="dc:title">The trouble with Bob</h2>
      ...
      <h3 property="dc:creator" resource="#me"><span property="scher
      <div property="schema:articleBody">
        <p>The trouble with Bob is that he takes much better photos
      </div>
     ...
   </div>
 </body>
</html>
```

Lo anterior resulta un poco incómodo. Afortunadamente, RDFa permite que el valor del atributo `property` sea una lista de valores, es decir, también puede escribir:

Ejemplo

```
<html>
 <head>
   ...
 </head>
 <body prefix="dc: http://purl.org/dc/terms/ schema: http://schema.c
   <div resource="/alice/posts/trouble_with_bob" typeof="schema:Blog
      <h2 property="dc:title">The trouble with Bob</h2>
      ...
      <h3 property="dc:creator schema:creator" resource="#me">Alice<
      <div property="schema:articleBody">
        <p>The trouble with Bob is that he takes much better photos
      </div>
     ...
   </div>
 </body>
</html>
```

obteniéndose la siguiente estructura:

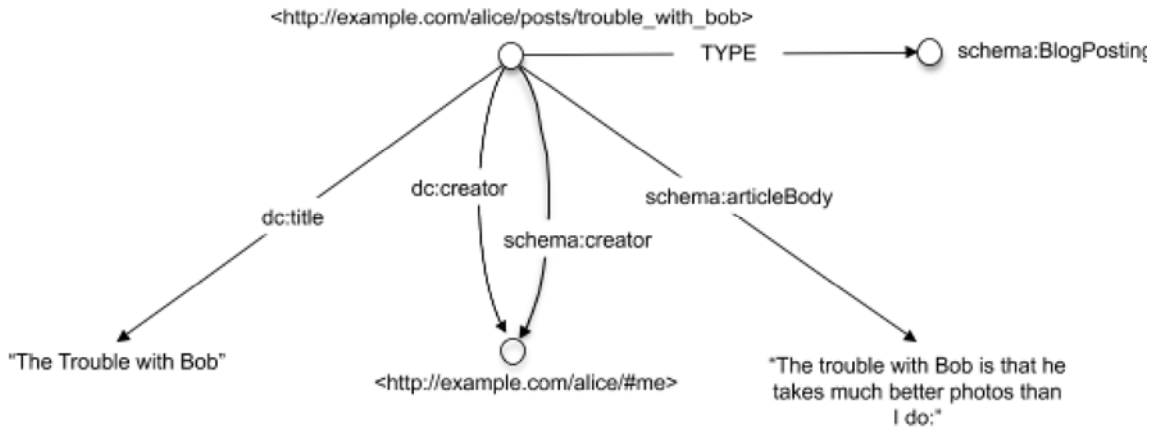


Figura 12: Item del blog de Alice utilizando dos vocabularios diferentes, incluyendo dos propiedades con el mismo contexto y objeto.

2.1.4.2 Prefijos por defecto (contexto inicial)

Ciertos vocabularios son muy utilizados por la comunidad web junto con determinados prefijos. El vocabulario Dublin Core es un buen ejemplo. Estos vocabularios tan comunes tienden a definirse una y otra vez, y, a veces los autores de páginas Web se olvidan declararlos por completo.

Para solucionar este problema, RDFa introduce el concepto de *contexto inicial* que define un conjunto de prefijos por defecto. Estos prefijos, cuya lista se mantiene y se actualiza periódicamente por el W3C, proporcionan una serie de prefijos predefinidos conocidos por el procesador de RDFa. Las declaraciones de prefijo de un documento siempre prevalecen sobre las declaraciones realizadas a través de estos valores por defecto, pero si el autor de la página web olvida declarar uno de estos vocabulario tan comunes, como Dublin Core o FOAF, el procesador de RDFa recurrirá a ellos. La lista de los prefijos por defecto están disponibles en la web para cualquiera que desee consultarla.

El siguiente ejemplo no declara el prefijo `dc` utilizando el atributo `prefix`:

Ejemplo

```
<html>
 <head>
   ...
 </head>
 <body>
   <div>
      <h2 property="dc:title">The trouble with Bob</h2>
      ...
      <h3 property="dc:creator" resource="#me">Alice</h3>
       ...
   </div>
 </body>
</html>
```

Sin embargo, un procesador RDFa sigue reconociendo los términos `dc:title` y `dc:creator` y amplía dichos valores a sus URLs correspondientes. El procesador de RDFa es capaz de hacer esto porque el prefijo `dc` forma parte de los prefijos por defecto en el contexto inicial.

Note

Los prefijos por defecto se utilizan como un mecanismo para corregir los documentos RDFa donde los autores accidentalmente olvidaron declarar prefijos comunes. Hay que tener en cuenta que aunque los autores confíen que estén disponibles para documentos RDFa 1.1, los prefijos podrían cambiar en el transcurso de 5-10 años. No obstante, la política del W3C es que una vez que un prefijo se define como parte de un perfil predeterminado, ese prefijo no será modificado o eliminado. Sin embargo, la mejor manera de asegurarse de que los prefijos que utilizan los autores siempre se asignan con el vocabulario deseado es utilizar el atributo `prefix` para declararlos.

Dado que los prefijos por defecto están destinados a ser un mecanismo de último recurso para ayudar a los autores noveles, el marcado mostrado en el ejemplo anterior no es recomendable. En el resto de este documento se utilizan las prácticas de escritura recomendadas, declarando todos los prefijos para que queden explícitas las intenciones del autor del documento.

2.2 Profundizando: RDFa Core

Como se ha visto en las secciones anteriores, RDFa Lite es bastante potente. Alice podría expresar conjuntos realmente complejos de información estructurada. Sin embargo, hay casos en los que los atributos presentados hasta ahora no cubren todas las necesidades, o dan lugar a estructuras HTML muy incómodas y propensas a errores. En esos casos RDFa ofrece otras posibilidades, a través atributos adicionales que pueden "acudir al rescate", algunos de los cuales se presentan en esta sección.

Note

RDFa Lite no define una clase distinta de procesadores RDFa. En otras palabras, se supone que los procesadores que cumplen la conformidad con RDFa deben manejar todas las funciones de RDFa, no sólo aquellas utilizadas por RDFa Lite.

2.2.1 Usando el atributo `content`

Cuando creó su blog, Alice decidió utilizar esta estructura sencilla para añadir información Dublin Core a las entradas de su blog (ver además la Figura 2):

Ejemplo

```
<html>
 <head>
   ...
 </head>
```

```
  <body>
    ...
    <h2 property="http://purl.org/dc/terms/title">The Trouble with Bo
    <p>Date: <span property="http://purl.org/dc/terms/created">2011-0
    ...
  </body>
</html>
```

Sin embargo, para hacer eso, Alice utilizó una convención . En efecto, aunque la cadena "2011-09-10" identifica inequívocamente una fecha legible por máquina, no parece algo muy natural para un lector humano. Seguramente un lector nativo de Inglés preferiría algo así como "10th of September, 2011". Por otro lado, aunque también es posible para una máquina analizar e interpretar dicha cadena como una fecha, resulta claramente más complicado hacerlo. El problema es que por defecto RDFa utiliza el contenido textual de un elemento como valor de una propiedad. Aunque esto funcione bien en la mayoría de los casos, a veces, como en el ejemplo anterior, conlleva consecuencias incómodas.

Para solventar este problema RDFa permite reutilizar el atríbuto `content` de HTML. La entrada del blog podría escribirse del siguiente modo:

Ejemplo

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
    <h2 property="http://purl.org/dc/terms/title">The Trouble with Bo
    <p>Date: <span property="http://purl.org/dc/terms/created" conten
    ...
  </body>
</html>
```

Se obtiene una estructura exactamente igual que la anterior (Figura 2). La diferencia es la presencia del atributo `content`: se ordena al procesador RDFa anular el comportamiento por defecto que utiliza el contenido textual, pasando a utilizar el valor del atributo `content`. Mediante este atributo Alice podría proporcionar una fecha más legible, al tiempo que mantiene un contenido inequívoco para las máquinas al utilizar datos estructurados.

El atributo `content` tiene otro uso importante. La aproximación "tradicional" de añadir metadatos simples a una página web ha utilizado la cabecera del documento a través de los elementos `link` y `meta`. Aunque no haya problema para usar `link` en RDFa Lite (puesto que utiliza el atributo `href` que puede ser utilizado para, por ejemplo, definir enlaces "con sabor"), el hecho es que, en un fichero que cumpla la conformidad con HTML, el elemento `meta` puede que no tenga contenido textual. Esto significa que el *único* modo de utilizar la cabecera para este tipo de declaraciones es mediante el uso del atributo `content`. Por ejemplo, utilizar el elemento `meta` es la aproximación sugerida por Facebook para el vocabulario del Protocolo de Grafos Abiertos [*OGP*]; si por ejemplo, Alice quiere hacer uso del botón "Me gusta" en sus entradas, debería añadir el siguiente código en la cabecera:

Ejemplo

```
<html>
 <head prefix="og: http://ogp.me/ns#" >
   ...
   <meta property="og:title" content="The Trouble with Bob" />
   <meta property="og:type"  content="text" />
   <meta property="og:image" content="http://example.com/alice/bob-u
   ...
 </head>
 <body>
  ...
 </body>
</html>
```

Note

En este ejemplo el prefijo para el vocabulario del Protocolo de Grafos Abiertos [OGP] se define mediante el atributo `prefix`. Por desgracia, muchos autores olvidan hacerlo. Afortunadamente el prefijo `og` es parte el contexto inicial de RDFa, en consecuencia, la información resultante será válida incluso sin la declaración del prefijo...

2.2.2 Tipos de datos

Alice ya ha colocado información en su página sobre la licencia de uso:

Ejemplo

```
<p>All content on this site is licensed under
   <a property="http://creativecommons.org/ns#license" href="http:/,
     a Creative Commons License</a>. ©2011 Alice Birpemswick.</p>
```

Pero ella quiere completar lo anterior y dejar constancia de la fecha de declaración del copyright utilizando también datos estructurados. Para ello puede usar el término `date` de Dublin Core:

Ejemplo

```
<p>All content on this site is licensed under
   <a property="http://creativecommons.org/ns#license" href="http:/,
     a Creative Commons License</a>. ©<span property="dc:date">2011<
```

No obstante, el valor utilizado para indicar la fecha puede ser ambigua para las máquinas. Por supuesto, si un programa "sabe" que `http://purl.org/dc/terms/date` se refiere a una fecha, entonces por supuesto que es posible deducir que la cadena "2011" se refiere a un año. Pero puede que los procesadores, por ejemplo, proporcionen una presentación visual de todos los datos estructurados de una página específica, y podría utilizar un componente de software para interpretarlo y mostrarlo como un año y otro para hacer lo mismo pero como un número entero. ¿Cómo podría saber un procesador cual escoger?

Alice puede decidir que resulta útil añadir información adicional a ese elemento en la forma de un tipo de dato (*datatype*). Esta información adicional puede ser transmitida al procesador RDFa mediante el atributo RDFa `datatype`:

Ejemplo

```
<p>All content on this site is licensed under
   <a property="http://creativecommons.org/ns#license" href="http:/,
     a Creative Commons License</a>. ©<span property="dc:date" datat
```

donde `xsd:gYear` se refiere a `http://www.w3.org/2001/XMLSchema#gYear`, y es uno de los tipos de datos estándar definidos por la especificación de tipos de datos del W3C [*XSD11*] que contiene tipos como el booleano, entero, fecha o coma flotante. (`xsd` es uno de los prefijos por defecto de RDFa.)

2.2.3 Alternativa para establecer el contexto: `about`

Alice ha utilizado los siguientes patrones para definir los datos estructurados para cada entrada individual:

Ejemplo

```
<div resource="/alice/posts/trouble_with_bob">
   <h2 property="title">The trouble with Bob</h2>
   <h3 property="creator" resource="#me">Alice</h3>
   ...
</div>
```

El papel del atributo `resource` en el elemento `div` es el de establecer el "contexto", es decir, el sujeto para todas las declaraciones posteriores. Además, cuando se combina con el atribto `property`, `resource` puede utilizarse para definir el objeto de la declaración (al igual que `href`).

Este patrón es perfectamente correcto, pero puede llegar a ser demasiado verboso (con mucho código) en algunos casos. En efecto, supongamos que Alice desea crear una página índice separada para todas sus entradas, y la única información que le gustaría incluir en ella, como datos estructurados, son las referencias a los títulos. Siguiendo el mismo patrón, tendría que hacer algo como:

Ejemplo

```
<ul>
  <li resource="/alice/posts/trouble_with_bob"><span property="title
  <li resource="/alice/posts/jos_barbecue"><span property="title">Jo
  ...
</li>
```

Efectivamente, lo anterior funciona, pero esta solución resulta algo complicada. Mezclar toda la información anterior en un único elemento, es decir:

Ejemplo

```
<ul resource="/alice/posts/trouble_with_bob">
  <li resource="/alice/posts/trouble_with_bob" property="title">The
  ...
</li>
```

podría no ser correcto; la combinación de `property` y `resource` podría generar una declaración distinta a la que originalmente se pretendía obtener.

RDFa introduce un atributo separado, denominado `about`, que puede usarse como una alternativa a `resource` en el establecimiento del contexto. Usando dicho atributo, Alice podría escribir:

Ejemplo

```
<ul>
  <li about="/alice/posts/trouble_with_bob" property="title">The tr
  <li about="/alice/posts/jos_barbecue" property="title">Jo's Barbe
  ...
</li>
```

La diferencia fundamental entre `about` y `resource` es que el primero solo se utiliza para establecer el contexto, ya sea combinándolo con el atributo `property` en el mismo elemento o no. Esto también significa que, para dicho uso, `about` y `resource` son intercambiables, es decir, en la entrada original, Alice podría haber elegido escribir:

Ejemplo

```
<div about="/alice/posts/trouble_with_bob">
   <h2 property="title">The trouble with Bob</h2>
   <h3 property="creator" resource="#me">Alice</h3>
   ...
</div>
```

2.2.4 Alternativa para establecer la propiedad: `rel`

Otro patrón que Alice utilizó en su código es el siguiente:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#me">
   <ul>
      <li property="knows" resource="http://example.com/bob/#me" typ
        <a property="homepage" href="http://example.com/bob/"><span
      </li>
      <li property="knows" resource="http://example.com/eve/#me" typ
        <a property="homepage" href="http://example.com/eve/"><span
      </li>
      <li property="knows" resource="http://example.com/manu/#me" ty
        <a property="homepage" href="http://example.com/manu/"><spar
      </li>
   </ul>
</div>
```

Cada "rama" en la lista define un objeto separado (nodos en blanco en este caso) y la misma propiedad (`foaf:knows`) es utilizada para vincularla con el mismo contexto. El atributo `property="knows"` ha de repetirse en cada elemento de la lista para definir la propiedad correspondiente. Si esta estructura es generada por un CMS (Content Management System, Sistema de Gestión de Contenidos) no supone ningún problema. Sin embargo, si dicha estructura se crea manualmente, es claramente propensa a errores: es posible que se olvide incluir el nombre de la propiedad o que se escriba mal.

En su lugar, Alice podría utilizar otro atributo RDFa, denominado `rel`. Usando este atributo el código HTML correspondiente quedaría de la siguiente forma:

Ejemplo

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#me">
   <ul rel="knows">
      <li resource="http://example.com/bob/#me" typeof="Person">
        <a property="homepage" href="http://example.com/bob/"><span
      </li>
      <li resource="http://example.com/eve/#me" typeof="Person">
        <a property="homepage" href="http://example.com/eve/"><span
      </li>
      <li resource="http://example.com/manu/#me" typeof="Person">
        <a property="homepage" href="http://example.com/manu/"><spar
      </li>
   </ul>
</div>
```

En contraste con el atributo `property`, `rel`*nunca* considera el contenido textual de un elemento (o el valor del atributo `content`). En su lugar, si no se ha especificado claramente el objeto de un enlace mediante, por ejemplo, el atributo `resource` o el atributo `href`, el procesador supone que debe continuar y encontrar uno o más objetos en la jerarquía y utilizarlos. Esto es lo que sucede en este caso: el atributo `knows` del elemento `ul` no incluye ningún objeto evidente; sin embargo, el procesador lo encuentra en los elementos individuales `li` y los utiliza. El uso de este tipo de patrón está muy extendido en RDFa.

Note

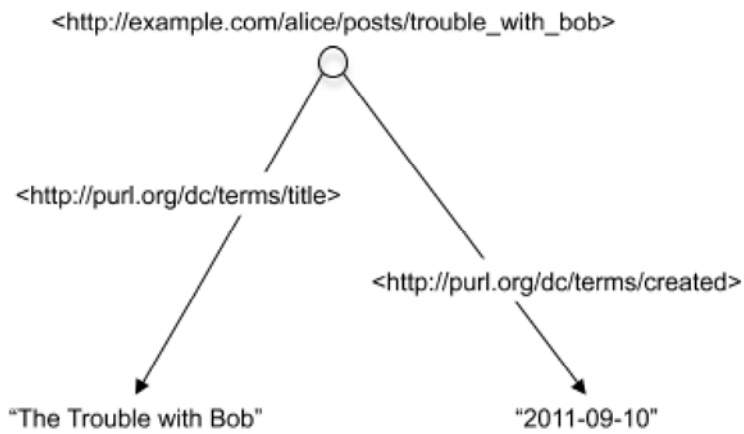
En muchas situaciones, `property` y `rel` son intercambiables cuando los datos estructurados previstos incluyan enlaces (con sabor). Sin embargo, hay sutiles diferencias que implican, por ejemplo, "encadenamientos" que deben usarse con cuidado. El lector interesado puede consultar la sección relevante de la especificación RDFa 1.1 para más detalles. En general, se aconseja mantener `property` cuando sea posible.

3. ¿Mencionaste algo sobre RDF?

RDFa se beneficia de la potencia de RDF [*RDF-PRIMER*], el estándar del W3C's para datos interoperables legibles por máquina. Aunque los lectores de este documento no esperen comprender RDF, algunos pueden estar interesados en como se interrelacionan ambas especificaciones.

RDF, (Resource Description Framework, Marco de Descripción de Recursos) es la representación abstracta de los datos utilizados para los grafos de los ejemplos de este documento. Cada flecha del grafo se representa como una tripleta sujeto-propiedad-objeto: el sujeto es el nodo en el que comienza la flecha, la propiedad es la flecha en sí misma, y el objeto es el nodo o literal al final de la flecha. Un conjunto de tales tripletas RDF también se denomina "grafo RDF" y se almacena en un "Almacen de tripletas" (Triplestore) o "Almacen de grafos" (Graph Store).

Consideremos el grafo del primer ejemplo:

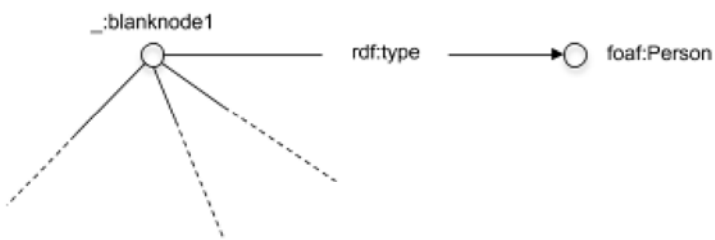


Las dos tripletas RDF de dicho grafo se escriben, utilizando la sintaxis Turtle [*TURTLE*] para RDF, del siguiente modo:

Ejemplo

```
<http://www.example.com/alice/posts/trouble_with_bob>
    <http://purl.org/dc/terms/title> "The Trouble with Bob" ;
    <http://purl.org/dc/terms/created> "2011-09-10" .
```

Las flechas que representan la propiedad **TYPE** no son diferentes de otras flechas. **TYPE** es otra propiedad esencial de RDF representada como `rdf:type`. El vocabulairo `rdf` se encuentra definido en `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. La información de contacto del ejemplo anterior podría representarse mediante el siguiente grafo:



La importancia de RDF radica en que proporciona un lenguaje universal para representar datos y relaciones. Una unidad de datos puede tener cualquier número de propiedades que son expresadas como URLs. Dichas URLs pueden ser utilizadas por cualquier editor, de igual modo que cualquier editor

web puede enlazar con cualquier página, incluso cuando no han sido creadas por ellos. Si se disponen de datos, en forma de tripletas RDF y obtenidos de diferentes localizaciones, es posible utilizar el lenguaje de consulta SPARQL [*RDF-SPARQL-QUERY*] para buscar los "amigos de Alice que hayan creado algún ítem cuyo título contega la palabra 'Bob'", con independencia de si esos ítems sean entradas de blog, vídeos, eventos de un calendario u otro tipo de datos.

RDF es un modelo de datos abstracto diseñado para maximizar la reutilización de vocabularios. RDFa es un modo de expresar datos RDF dentro de documentos HTML, de manera que sean legibles y reutilizables por máquinas.

3.1 Vocabularios personalizados

Según Alice vaya marcando sus páginas con RDFa, descubrirá nuevas necesidades para expresar datos, como sus fotos favoritas, que no son cubiertas por ningún vocabulario existente. Si fuera preciso, Alice puede crear un vocabulario personalizado ajustado a sus necesidades. Una vez creado el vocabulario, puede utilizarse para el marcado RDFa como cualquier otro vocabulario.

Las instrucciones acerca de la creación de vocabularios, utilizando lo que se denomina RDF Schema, están disponibles en la sección 5 del documento RDF Primer [*RDF-PRIMER*]. A alto nivel, la creación de un vocabulario para RDF implica:

1. Seleccionar una URL donde resida el vocabulario, por ejemplo: `http://example.com/photos/vocab#`.
2. Publicar el documento del vocabulario en dicha URL. El documento del vocabulario define las clases y propiedades que conforman el vocabulario. Por ejemplo, Alice puede definir las clases `Photo` y `Camera`, así como la propiedad `takenWith` que relaciona una fotografía con la cámara utilizada para tomarla.
3. Utilizando el vocabulario en un documento HTML ya sea con el atribto `vocab` o con el mecanismo de declaración de prefijos. Por ejemplo: `prefix="photo: http://example.com/photos/vocab#"` y `typeof="photo:Camera"`.

Merece la pena señalar que cualquiera que pueda publicar un documento en la Web también puede publicar un vocabulario y definir de esta forma nuevos campos de datos que desee expresar. RDF y RDFa permiten la ampliación de vocabularios de un modo distribuido.

4. Herramientas RDFa

Hay una amplia variedad de herramientas que se pueden utilizar para generar o procesar datos RDFa. Una buena fuente que recopila referencias de estas herramientas es la página sobre RDFa de la wiki de la Web Semántica del W3C. Pese a todo se debe tener cuidado ya que algunas de estas herramientas puedan estar relacionadas con versiones anteriores de RDFa. Otra fuente puede ser la página web de implementación de la comunidad RDFa. Ambas fuentes están en constante cambio. Por cierto, la segunda

fuente forma parte de un sitio web de una comunidad que contiene otros ejemplos de uso de RDFa, información general, así como información sobre como participar en la comunidad. En particular, los ejemplos de fragmentos RDFa se pueden probar usando el editor en tiempo real RDFa 1.1 que también puede mostrar una representación visual de los datos estructurales subyacentes.

5. Agradecimientos

En el momento de su publicación, los miembros activos del Grupo de Trabajo para la aplicación de RDF en la Web eran:

- Stéphane Corlosquet, Massachusetts General Hospital
- Ivan Herman, W3C
- Gregg Kellogg (Invited Expert)
- Niklas Lindström (Invited Expert)
- Shane McCarron, Applied Testing and Technology, Inc. (Invited Expert)
- Steven Pemberton, Centre Mathematics and Computer Science
- Manu Sporny, Digital Bazaar (Chair, Invited Expert)
- Ted Thibodeau, OpenLink Software

Gracias también a Grant Robertson y Guus Schreiber quienes, aunque no formaran parte del Grupo de Trabajo, han proporcionado comentarios útiles sobre borradores iniciales de esta nota.

A. Referencias

A.1 Referencias normativas

No hay referencias normativas.

A.2 Referencias informativas

[CC-ABOUT]
Creative Commons: About Licenses URL: http://creativecommons.org/about/licenses/

[DC11]
Dublin Core Metadata initiative. *DCMI Metadata Terms* October 2010. DCMI Recommendation. URL: http://dublincore.org/documents/dcmi-terms/

[FOAF]
Dan Brickley, Libby Miller. *FOAF Vocabulary Specification 0.98.* 9 August 2010. URL: http://xmlns.com/foaf/spec/

[HTML-RDFA]
Manu Sporny and Shane McCarron *HTML+RDFa 1.1* 29 March 2012. W3C Working Draft. URL: http://www.w3.org/TR/2012/WD-rdfa-in-html-20120329/

[OGP]

The Open Graph Protocol. December 2010. URL: http://ogp.me

[RDF-PRIMER]
Frank Manola; Eric Miller. *RDF Primer.* 10 February 2004. W3C Recommendation. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[RDF-SPARQL-QUERY]
Andy Seaborne; Eric Prud'hommeaux. *SPARQL Query Language for RDF.* 15 January 2008. W3C Recommendation. URL: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115

[RDFA-CORE]
Shane McCarron; et al. *RDFa Core 1.1: Syntax and processing rules for embedding RDF through attributes.* 07 June 2012. W3C Recommendation. URL: http://www.w3.org/TR/2012/REC-rdfa-core-20120607/

[RDFA-LITE]
Manu Sporny; *RDFa Lite 1.1.* 07 June 2012. W3C Recommendation. URL: http://www.w3.org/TR/2012/REC-rdfa-lite-20120607/
Traducción en castellano: http://skos.um.es/TR/rdfa-lite/

[RDFA-SYNTAX]
Ben Adida, et al. *RDFa in XHTML: Syntax and Processing.* 14 October 2008. W3C Recommendation. URL: http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014

[SCHEMA]
Schemas—schema.org

[SVG11]
Erik Dahlström; et al. *Scalable Vector Graphics (SVG) 1.1 (Second Edition).* 16 August 2011. W3C Recommendation. URL: http://www.w3.org/TR/2011/REC-SVG11-20110816/

[TURTLE]
David Beckett, Tim Berners-Lee. *Turtle: Terse RDF Triple Language.* January 2008. W3C Team Submission. URL: http://www.w3.org/TeamSubmission/turtle/

[XHTML-RDFA]
Shane McCarron; et. al. *XHTML+RDFa 1.1.* 07 June 2012. W3C Recommendation. URL: http://www.w3.org/TR/2012/REC-xhtml-rdfa-20120607/
Traducción en castellano: http://skos.um.es/TR/xhtml-rdfa/

[XSD11]
Henry S. Thompson et al.; *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes* 05 April 2012. W3C Recommendation. URL: http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/