Universidad de Deusto
Deustuko Unibertsitatea

mobility research lab

# Web Semántica Práctica

Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es

## Contents

Universidad de Deusto
Deustuko Unibertsitatea

mobility research lab

# 1. Web 2.0

Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es

## Web 2.0

## ¿Qué es Web 2.0?

- Una “palabreja” (buzzword) que hace referencia a:
  - Todo aquello nuevo y popular en la web
  - Web participativa tanto de humanos como de máquinas
  - Cambio en la manera en que la gente ve la web:
    - “Read/Write Web” y la “Web como una Plataforma”
- Acuñado por Tim O'Reilly y Dale Dougherty
  - Observaron que varias aplicaciones web utilizan tecnologías existentes de una manera nueva e innovadora
    - Basada en una industria más madura (economía web sana)



## Revolución Web 2.0

- Repentina renovación de energía en la web
- Nuevas aplicaciones apareciendo cada día
- Grandes empresas mostrando su talento
- Inversión en web start-ups de nuevo
- Pero:
  - No comentamos los errores del 2000
  - Temas de usabilidad/accesibilidad comprometidos
  - Aplicaciones interesantes, pero no modelo negocio

## Web 2.0 como Plataforma

- La Web está pasando de ser un sistema de envío de documentos a …
  - Una plataforma de aplicaciones
- Simplifica la distribución
- Promociona el modelo de subscripción en vez de la compra de una vez



## Web 1.0 vs. Web 2.0

| Web 1.0 | Web 2.0 |
|---|---|
| Altavista | Google |
| Hotmail | Yahoo Mail |
| Ofoto | Flickr |
| Mp3.com | iTunes |
| Geocities | Blogger |
| MapQuest | Google Maps |
| Encarta | Wikipedia |
| Slashdot | Digg |

## Requisitos para un Aplicación Web 2.0

- Datos abiertos
  - Formatos de datos abiertos
  - Habilidad para usar datos fuera de la aplicación
  - Permite al usuario crearse sus propios datos
- Arquitectura de participación
  - Provee un servicio no un producto
  - Incentiva la participación
  - Inteligencia colectiva
  - Fácil reutilizar y mezclar
  - Formar parte de una comunidad
- Buena experiencia de usuario
  - Fácil de usar y atractiva
  - Interfaz de usuario rica
  - Funciona como una aplicación tradicional



## Tecnologías Web 2.0 Claves

- Apertura de datos a través de APIs y Servicios Web
- RSS
- Ajax
- Estándares web (DOM, XHTML, CSS)

## Tecnologías de Presentación Web 2.0

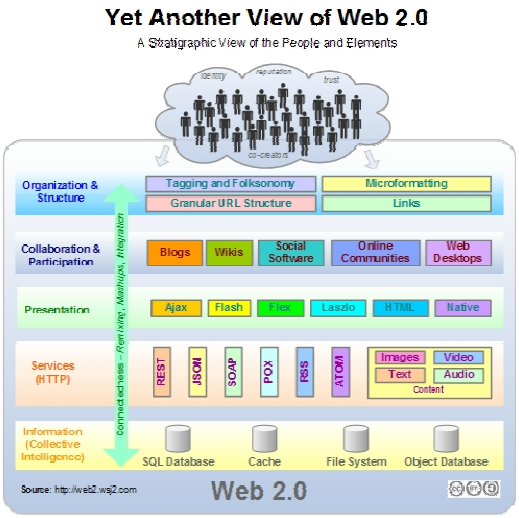
- Aplicaciones AJAX
  - http://www.ajaxian.com/
- Desktop Widgets
  - http://widgets.yahoo.com/
- Aplicaciones Flex
  - http://www.adobe.com/products/flex/
- OpenLazlo
  - http://www.openlaszlo.org/
- XUL
  - http://www.mozilla.org/projects/xul/
- Smart Clients and Avalon
  - http://msdn.microsoft.com/winfx/technologies/presentation/default.aspx



## Aplicación Web 2.0

- Un portal Web 2.0 suele presentar las siguientes características:
  - Rico mecanismo de interacción: Ajax, Lazslo
  - CSS
  - XHMTL valido o utilización de microformatos (añadir semántica en HTML)
  - Sindicación y agregación de datos basada en RSS y Atom
  - Publicación de Weblogs
  - Mashups
  - REST o XML WebServices APIs

## Arquitectura Web 2.0






## AJAX

- AJAX (Asynchronous Javascript and XML), técnica de desarrollo que genera aplicaciones web más interactivas combinando:
  - XHTML y CSS para la presentación de información
  - Document Object Model (DOM) para visualizar dinámicamente e interactuar con la información presentada
  - XML, XSLT para intercambiar y manipular datos
    - JSON y JSON-RPC pueden ser alternativas a XML/XSLT
  - `XMLHttpRequest` para recuperar datos asíncronamente
  - Javascript como nexo de unión de todas estas tecnologías


## ¿Por qué AJAX?

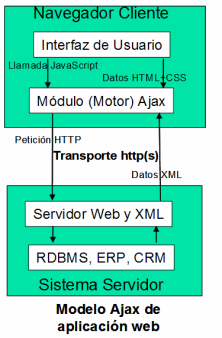
- Las aplicaciones web proliferan debido a su simplicidad, pero:
  - Ofrecen una menor interactividad y usabilidad en comparación con las aplicaciones desktop.
  - La interacción del usuario con una aplicación web se interrumpe cada vez que se necesita algo del servidor
- Varias tecnologías han sido diseñadas para resolver este problema:
  - Java Applets, FLASH
- AJAX permite lo mismo pero sin plug-ins

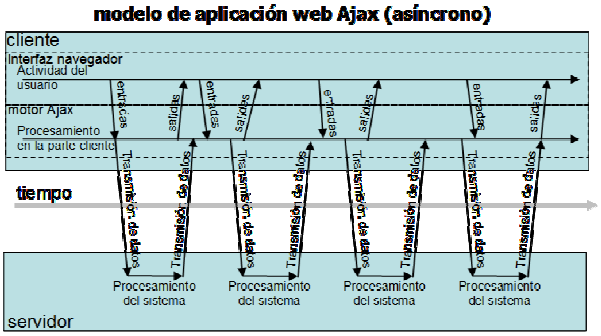


## Características AJAX

- Aplicaciones son más interactivas al estilo desktop
  - *Look and feel* similar a las aplicaciones de sobremesa sin plug-ins o características específicas de los navegadores
- Reduce tamaño de la información intercambiada
  - Muchas micro-peticiones, flujo de datos global inferior
- Libera de procesamiento a la parte servidora???
- Actualiza porciones de la página en vez de la página completa
- Necesario asegurar aplicación AJAX funciona en todo navegador

## Arquitectura AJAX






## Aplicaciones AJAX Famosas

- Empresas de referencia en la web definen soluciones AJAX:
  - Google
    - Orkut (https://www.orkut.com/Login.aspx) es una comunidad virtual que conecta online a gente a través de una red de amigos.
    - Gmail (www.gmail.com)
    - Google Suggest (http://www.google.com/webhp?complete=1&hl=en) → sugiere valores de búsqueda a medida que escribes caracteres
    - Google Maps (http://maps.google.com/)
  - Yahoo!
    - Flickr (http://www.flickr.com/) es una aplicación para gestionar y compartir fotos
    - Oddpost (http://oddpost.com/learnmore)
      - El equipo de Oddpost ha rediseñado Yahoo! Mail siguiendo la filosofía AJAX
- En definitiva, AJAX es un buena solución técnica con gran aplicabilidad, demostrada por aplicaciones reales complejas.

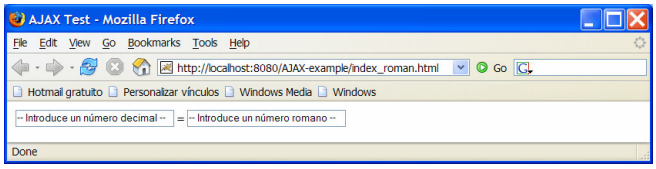
## Problemas con AJAX

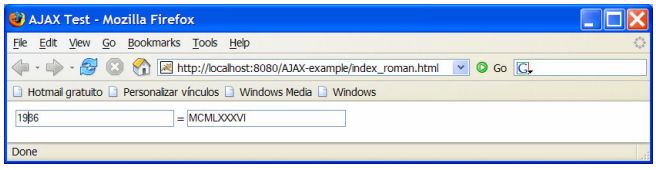
- Disponibilidad del objecto `XMLHttpRequest`
- Usabilidad
- Carga del Servidor
- Comportamiento Asíncrono



## Ejemplo AJAX

- Conversor números romanos a árabes



## Ejemplo AJAX


- En el HTML:

```
<input type="text" size="30" id="decimalNum" value
  ="-- Introduce un número decimal -- “
  onkeyup="traducirDecimalARomano();">
<input type="text" size="30" id=“romanNum"   value
  ="-- Introduce un número romano -- “
  onkeyup="traducirRomanoADecimal();">
```



## Ejemplo AJAX


- En JavaScript:

```
function traducirDecimalARomano() {
   var idField = document.getElementById("decimalNum");
   if (isPositiveInteger(idField.value)) {
      var url = "convert?numDecimal=" + escape(idField.value);
      if (window.XMLHttpRequest) {
         req = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
         req = new ActiveXObject("Microsoft.XMLHTTP");
      }
      req.open("GET", url, true);
      req.onreadystatechange = callback;
      req.send(null);
      return true;
   } else {
      alert("Texto introducido no es un número entero: " +
   idField.value);
      idField.value = "";
      return false;
   }
}
```

## Ejemplo AJAX

- En JavaScript:

```
function callback() {
   if (req.readyState == 4) {
      if (req.status == 200) {
         // update the HTML DOM
         var message =
req.responseXML.getElementsByTagName("message")[0];
         var responseElement =
document.getElementById("romanNum");
                  responseElement.value =
message.childNodes[0].nodeValue;
      }
   }
}
```



## Frameworks AJAX

- JavaScript puras:
  - http://prototype.conio.net/
  - http://script.aculo.us/
  - http://openrico.org/demos.page
- Parte Servidora:
  - http://www.getahead.ltd.uk/dwr
  - http://atlas.asp.net/Default.aspx?tabid=47

## Wikis



- Un tipo de portal que permite a los usuarios editar, añadir, borrar su contenido de manera rápida y sencilla
  - Herramienta efectiva de escritura colaborativa
  - A través del browser y utilizando una sintaxis muy simple el usuario puede escribir documentos
- MediaWiki es una buena herramienta




## Weblogs



- Bitácoras web que recogen artículos periódicos en orden cronológico inverso
- Se concentran en una temática particular:
  - Comida
  - Política
  - Tecnología
- Dan oportunidad a que la gente comente en la bitácora
- Herramientas: Blogger, WordPress

## Sindicación

- La sindicación web es una forma de sindicación donde un parte de un portal es hecho disponible para ser usado por otros
- Un portal facilita web feeds:
  - Web feed = documento XML con elementos de contenido (título, descripción) y enlaces a versiones largas del contenido
  - Varios formatos:
    - Rich Site Summary (RSS 0.91)
    - RDF Site Summary (RSS 0.9, 1.0 and 1.1)
    - Really Simple Syndication (RSS 2.0)
    - Atom



- Utilizamos agregadores para subscribirnos a web o podcast feeds
- Promociona un modelo “push” para la web



## RSS: Formato de Sindicación

```
<rss version="2.0">
  <channel>
    <title>Ejemplo de canal</title>
    <link>http://example.com/</link>
    <description>Ejemplo de fuente RSS</description>
    <language>es</language>
    <item>
      <title>1 &lt; 2</title>
      <link>http://example.com/1_less_than_2.html</link>
      <description>1 &lt; 2, 3 &lt; 4. En HTML, &lt;b&gt;
  comienza una frase en negrita
        y puedes comenzar un enlace con &lt;a href=
      </description>
      <enclosure url="http://rss.org/mp3s/news1.mp3"
  length="12216320" type="audio/mpeg" />
    </item>
  </channel>
</rss>
```

## Ingredientes Tecnológicos Web 2.0

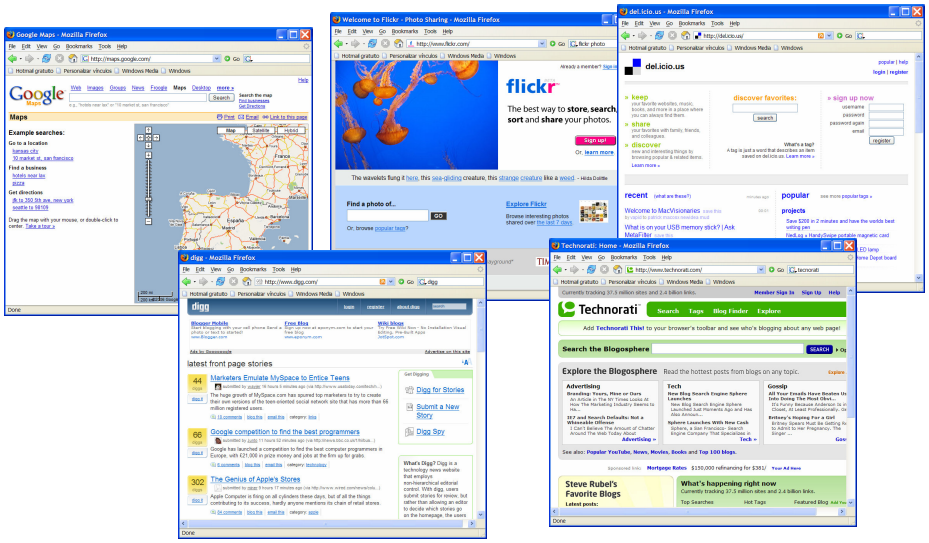
- Web 2.0 permite ensamblar nuevas aplicaciones “mezclando” funcionalidad de otras aplicaciones Web 2.0:
- Esto es posible gracias a:
  - Disponibilidad de APIs:
    - Google Maps API, permite ligar información de otras fuentes sobre un mapa
    - Otras APIs de eBay, Yahoo, Amazon
  - RSS como una interfaz: es un potente mecanismo de comunicación de cambios en portales y permite integrar datos de diversas fuentes
  - Folksonomías: o anotación comunitaria permite a un portal crear una categorización de sus contenidos de acuerdo a la opinión de sus visitantes.
  - Social networking: es la mejora de una aplicación cuando los usuarios designan su relación con los usuarios del mismo portal o aplicación



## Mash-ups

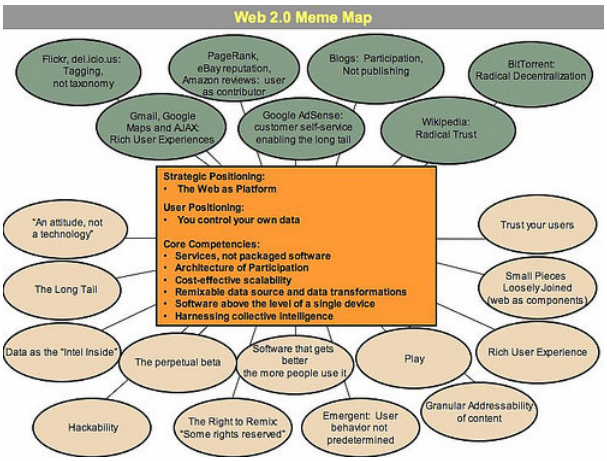
- Mash-up: una aplicación web que combina contenido de varias fuentes en una experiencia integrada
  - http://www.programmableweb.com/matrix
- Hay muchos mash-ups basados alrededor de GoogleMaps:
  - http://googlemapsmania.blogspot.com/
  - Algunos ejemplos:
    - Tagzania (http://www.tagzania.com/)
    - Maplandia.com News Center (http://www.maplandia.com/news/)
    - Real-time location of Dublin commuter trains (http://dartmaps.mackers.com/)
    - HousingMaps gets the locations of properties for sale or rent from Craigslist on the fly
    - Cheap Gas (http://www.mywikimap.com/)
    - Chicagocrime.org that taps into Google Maps to display where crimes occur in Chicago (http://www.chicagocrime.org/map/)

## Aplicaciones Web 2.0 Famosas






## Web 2.0: Resumen



## Referencias Web 2.0

- Web 2.0
  - Blog Dion Hinchcliffe
    - http://web2.wsj2.com/
  - Excelente presentación sobre Web 2.0
    - http://www.squidoo.com/introtoweb20/
  - Tim O'Reilly – "What Is Web 2.0, Design Patterns and Business Models for the Next Generation of Software"
    - http://www.oreillynet.com/lpt/a/6228



Universidad de Deusto
Deustuko Unibertsitatea

mobility research lab

# 2. SOA y Web Services

Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es

## Service Oriented Arquitecture (SOA)

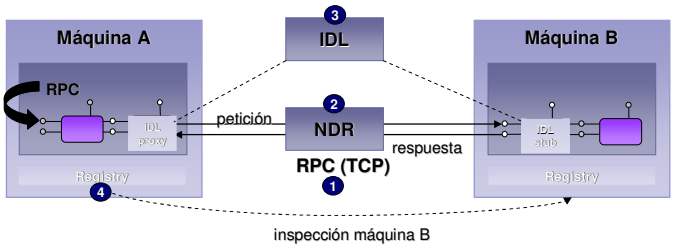
- Perspectiva de arquitectura software que utiliza servicios para dar soporte a los requerimientos de los usuarios
- Diferentes nodos hacen disponibles servicios que los participantes pueden acceder
- SOA promociona servicios desligados interoperables
  - La interoperabilidad se garantiza a través de la definición de contratos (WSDL)
  - No requiere uso de Servicios Web, aunque es lo normal
- Lenguajes de alto nivel como BPEL o la especificación WS-Coordination permiten orquestar servicios básicos en compuestos representando procesos de negocio



## Service Oriented Arquitecture (SOA)

- SOA permite proveer funcionalidad de aplicaciones y su consumo como servicios
- Los servicios pueden ser invocados, publicados y descubiertos
  - Son abstraídos de la implementación mediante una simple interfaz, basada en estándares.
- Conjunto de:
  - políticas
  - prácticas
  - frameworks
  - patrones de arquitectura

## Antes de SOA




1. Protocolo de Comunicación
2. Formato de Mensaje
3. Descripción del lenguaje
4. Mecanismo de Descubrimiento



## Antes de SOA

- Sin estándares universales no hay interoperabilidad
- Múltiples tecnologías para hacer lo mismo
  - No interoperables entre sí
  - Ligados a una plataforma

| | DCOM | CORBA | Java RMI |
|---|---|---|---|
| RPC Protocol | RPC | IIOP | IIOP or JRMP |
| Message Format | NDR | CDR | Java Ser. Format |
| Description | IDL | OMG IDL | Java |
| Discovery | Windows Registry | Naming Service | RMI Registry or JNDI |

## Motivación SOA

- Los usuarios no quieren cerrarse a una plataforma
- Es necesaria una arquitectura sin premisas e independiente de …
  - plataforma
  - lenguaje
  - objetos
  - mecanismos de llamada
- Bienvenido a SOA (Service Oriented Arquitecture)



## Visión SOA

- SOA ve el mundo de una forma *distinta*
  - Servicios autónomos
  - Fronteras explícitas, asumir heterogeneidad
  - Plataformas dispares
  - Integración basada en mensajes XML

## SOA Reference Model

- Pretende estandarizar el concepto de SOA
  - En Marzo del 2006, el grupo OASIS liberó su primer borrador
    - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm



## Conceptos Clave en SOA

| Término | Definición |
|---|---|
| *Service-Oriented Architecture* | Un paradigma para organizar y utilizar funcionalidad distribuida bajo el control de diferentes entidades. Ofrece mecanismos para ofrecer, descubrir, interactuar y usar las capacidades disponibles. |
| *Servicio* | Mecanismo mediante el cuál las necesidades de un consumidor son satisfechas con las capacidades de un productor |
| *Orquestación* | Mecanismo para la concatenación de servicios |
| *Coreografía* | Define mecanismos para la cooperación entre nodos participantes en una arquitectura SOA |
| *Stateless* | No depende en ningún estado anterior. Los servicios reciben toda la información que necesitan en la petición. |
| *Directorio* | Repositorio que describe los servicios disponibles en un dominio. |
| *Binding* | La relación entre un proveedor y un consumidor es dinámica, se establece en tiempo de ejecución. |

## Ventajas SOA

- SOA promociona la reutilización e interconexión de soluciones IT existentes en vez de empezar desde 0
  - Se ajusta perfectamente a los cambios de mercado
- SOA es una evolución de enfoques anteriores
- El uso de SOA implica la importancia de definir interfaces bien definidas e interoperables
  - Reduce los costes de integración y permite la evolución dinámica



## Importancia SOA

- Según Gartner:
  - *"By 2008, SOA will be a prevailing software engineering practice, ending the 40-year domination of monolithic software architecture (0.7 probability)"*

## Problemas de SOA

- Gestión de los metadatos de servicios
- Niveles de seguridad apropiados, ya que se usan servicios externos
  - WS-Security definido para dar respuesta a esto
- SOA y WS-* está en evolución
  - Pocos profesionales que dominan estas tecnologías



## SOA y los Servicios Web

- Los Servicios-Web son la clave de SOA
- Redefinición de las tecnologías distribuidas basada en XML
  - Comunicación vía protocolos de Internet
    - HTTP, SMTP, FTP…
  - SOAP como formato de mensaje
  - WSDL como definición de servicios
  - UDDI como localizador de Servicios-Web

## Servicios Web Básicos (WS-I)

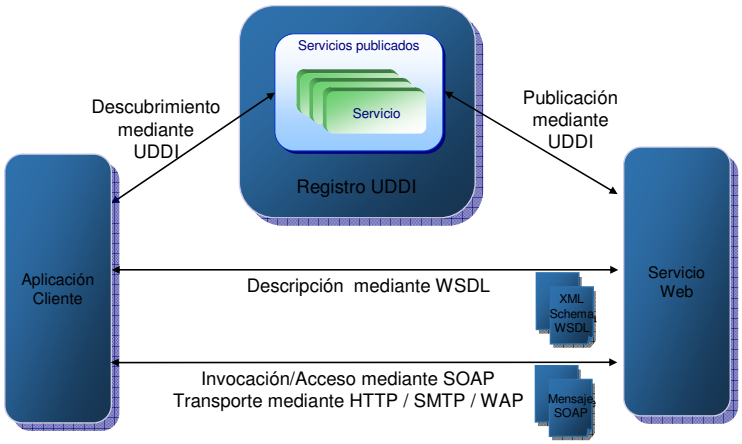
| | |
|---|---|
| **UDDI** | Registro de WS |
| **WSDL** | Descripción de WS |
| **XSD** | Sistema de tipos Portable |
| **SOAP** | Protocolo de mensajes |
| **XML 1.0 + Namespaces** | Mensajes Serializados |

- Protocolos
- Lenguajes de Descripción
- Mecanismos de Descubrimiento

**Ver ws-i.org para mas detalles**



## Servicios Web Protocolos



## Los Servicios Básicos no son Suficiente

- Los WS básicos (XSD, SOAP, WSDL, UDDI) consiguen una comunicación básica
  - Proporcionan intercambio básico de mensajes XML
  - Interconexión de sistemas heterogéneos
  - La compartición de esquemas permite mayores abstracciones
- Pero, la mayoría de las aplicaciones empresariales necesitan **MÁS**…



## Mejoras Necesarias

- Los Sevicios-Web tienen muchas necesidades comunes
  - Modelo de seguridad 'orientado a mensajes'
  - Mensajería estable y confiable
  - Soporte de Transacciones (entre WS)
  - Mecanismos de Direccionamiento y Ruteo
  - Mensajería Asíncrona
  - Metadatos para 'Políticas' de WS
  - Soporte para datos binarios

## ¿Cómo introducir esas mejoras?

- SOAP proporciona un marco de trabajo para gestionar aspectos nuevos
  - Header/Body permiten extensibilidad

```
<soap:Envelope
 xmlns:soap="...">
  <soap:Header>
   <!-Extensibilidad estándar con Cabeceras -->
  </soap:Header>
  <soap:Body>
   <!-- Trabajo -->
  </soap:Body>
</soap:Envelope>
```



## WS-*

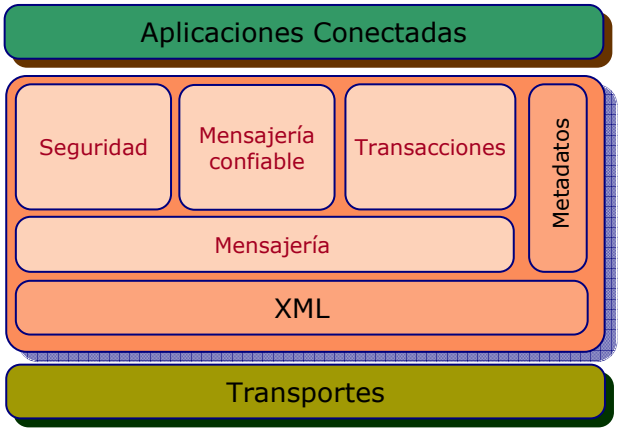
- WS-* (nuevas ESPECIFICACIONES WS) extiende SOAP con cabeceras estándar
  - Hay implementaciones de diferentes fabricantes (IBM, Sun, MS, etc.)
- Especificaciones estándar definidas en:
  - http://www.oasis-open.org

## WS-* Especificaciones

- **Messaging**
  - WS-Addressing
  - WS-Eventing
  - MTOM (Attachments)
- **Reliability**
  - WS-ReliableMessaging
- **Security**
  - WS-Security
  - WS-Trust
  - WS-SecureConversation
  - WS-Federation
- **Transactions**
  - WS-Coordination
  - WS-AtomicTransaction
  - WS-BusinessActivity
  - BPEL
- **Metadata**
  - WS-Policy
  - WS-PolicyAssertions
  - WS-PolicyAttachment
  - WS-SecurityPolicy
  - WS-Discovery
  - WS-MetadataExchange



## Arquitectura Futura WS-*



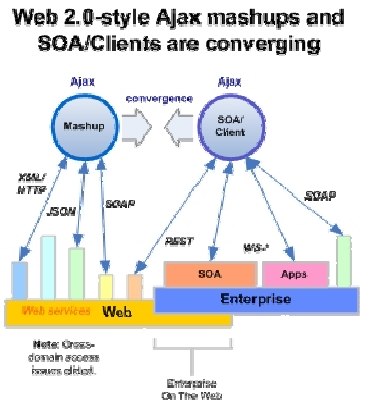
## Algunas Especificaciones Avanzadas

- WS-Addressing
  - Permite el paso de referencias a una implementación de un servicio web
  - Conjunto de propiedades
- MTOM (Message Transmission Optimization Mechanism)
  - Método para el envío eficiente de datos binarios
- WS-Security
  - Permite la autenticación entre peers
  - Confidencialidad en los mensajes
- WS-ReliableExchange
  - Garantiza el envío robusto de mensajes
- WS-Eventing
  - Permite un modelo de comunicación publish/subscribe en SOA



## Enterprise Web 2.0

- Convergencia Web 2.0 y SOA
  - Web 2.0 = Global SOA
  - Web 2.0 interfaz para SOA
- SOA:
  - Más centralizada, controlada
  - Sin interfaz
- Web 2.0 requiere de SOA



## References Web Services

- SOA
  - The Next Big Thing: Service-Oriented Architecture (SOA) Takes a New Route
    - http://java.sun.com/developer/technicalArticles/Interviews/routeone_qa.html?feed=JSC
  - OASIS SOA Reference Model
    - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm
- WS-* Specifications
  - An Introduction to the Web Services Architecture and Its Specifications
    - http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx?pull=/library/en-us/dnwebsrv/html/introwsa.asp
  - WS-BPEL Guide
    - http://smartcomps.kgbinternet.com/confluence/pages/viewpage.action?pageId=182



Universidad de Deusto
Deustuko Unibertsitatea

mobility research lab

# 3. Practical Cases: Deusto WebLab y Deusto Sentient Graffiti

Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es

## Deusto WebLab

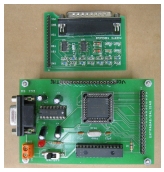
Universidad de Deusto
Deustuko Unibertsitatea

- En la Universidad de Deusto tenemos nuestro propio WebLab
  - Financiado por:
  - Gobierno Vasco
  - Universidad de Deusto
- Lo estáan utilizando alumnos en prácticas de asignaturas
  - 3º de Ingeniería Técnica Industrial especialidad en Electrónica Industrial, 2º semestre del curso 2004-2005, con PLDs
  - 5º de Ingenier´ıa en Autom´atica y Electr´onica Industrial, 1ºsemestre del curso 2005-2006, con FPGAs
  - 3º de Ingeniería Técnica Industrial especialidad en Electrónica Industrial, 2º semestre del curso 2005-2006, con PLDs



## Deusto WebLab

Universidad de Deusto
Deustuko Unibertsitatea



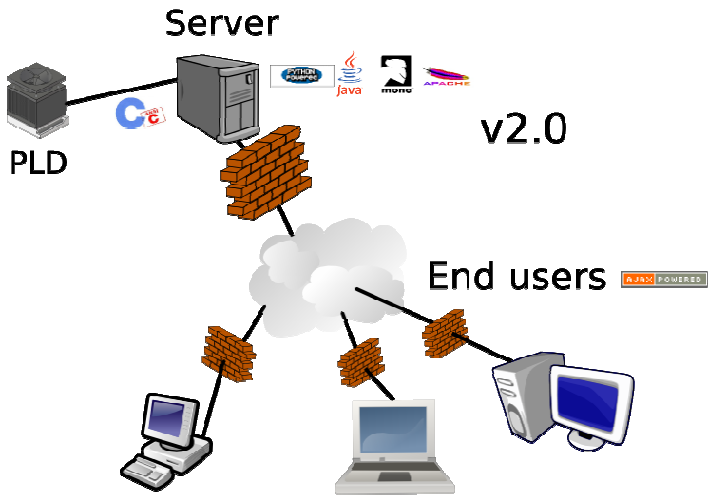
**WebLab PLD DEUSTO**

**Acceso al WebLab de lógica programable de la Universidad de Deusto:**

**UNIVERSIDAD DE DEUSTO**


Salir Continuar

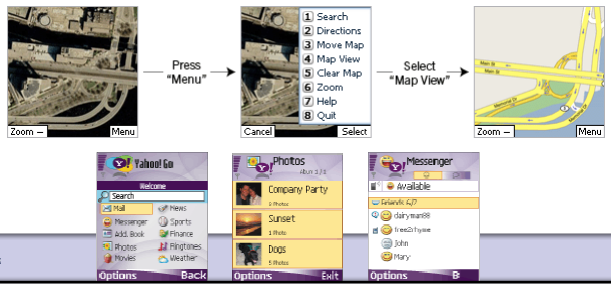
## Deusto WebLab






## Mobility 2.0 = Mobile Web 2.0

- Mobility 2.0 = Mobile Web 2.0
  - Web sites are becoming programmable
  - PROBLEM: We enjoy Web 2.0 in desktop but in mobile devices?
- Some relevant examples:
  - Google's Local for Mobiles (http://www.google.com/glm/)
  - Yahoo! Go Mobile (Contacts, Email, Photos, Messenger) http://go.connect.yahoo.com/go/mobile
  - Moblog clients (Mobile Blogger, KABLOG)



## Mobile Mash-ups

- Mobile Mash-up: a web application adapted to mobile devices combining content from several sources into an integrated experience
  - Traditional mobile phone-based data usage is downstream
  - Mobile Mash-ups can definitely push the upstream usage
- Some cool mobile mash-ups:
  - Mobile Gmaps displays Google Maps, Yahoo! Maps, Windows Live Local and Ask.com Maps and satellite imagery on Java ME devices (http://www.mgmaps.com/)
  - Shozu = basic blog XML-RPC services + photo upload (http://www.shozu.com/portal/)
  - Socialight (http://socialight.com) places virtual "sticky" notes anywhere in the real world.
    - A StickyShadow = media (text, picture) + access rights + location



## Platforms to Develop Mobile Mash-ups

- Two main models:
  - **Browsing apps**, web apps which take into account limitations unique to mobility (e.g. small device)
    - Client capable of hardly any processing
    - XHTML (ASP.NET Mobile Web Controls & JSF)
  - **Smart Client apps**: downloaded and installed in the device
    - Capable of some processing, storage and intermittent communication
    - J2ME, Compact.NET, Python for Series 60, BREW uiOne, Flash Lite
  - Other minor ones: **hybrid?**, SIM, messaging and embbeded apps
- Current problems of mobile space apps:
  - Few mobile services are profitable (broadcast ones)
  - No consensus, same application developed for several platforms

## AJAX & Mobile Devices

- AJAX is a very important facet of Web 2.0
  - Avoids start-stop cycles thanks to Ajax Engine
    - The AJAX engine emits asynchronous calls to the server
    - The user **does not wait**
  - A combination of a number of existing technologies.
  - Solves two problems:
    - Superior UI experience
    - Standarised form of data retrieval

But NOT so much presence in mobile devices !!!



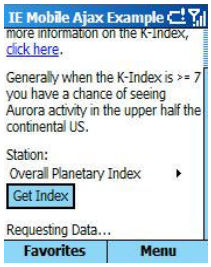
## AJAX & Mobile Devices

- Will AJAX replace J2ME, Compact.NET or XHTML as the platform to develop Mobile Applications?
  - AJAX (Asynchronous JavaScript and XML) makes **even more sense in the mobile space as it enables the creation of Web based services that are so fast they seem like local apps**
    - So far limited input and slow network connections prevented wider adoption
    - Now, simply load the AJAX app in the mobile and use XML to exchange data with the server:
      - Bandwidth constraint no problem any longer
      - Transparently update the information on the mobile

BUT WE NEED FLAT RATES AND ACCESS TO PHONE APIS!!!

## AJAX support on Mobile Devices

- All the devices that come with Opera Browser or Windows Mobile 5 support AJAX
  - High range Nokia s60 a s90
  - Nokia 770
- Small Rendering Technology paramount !!!

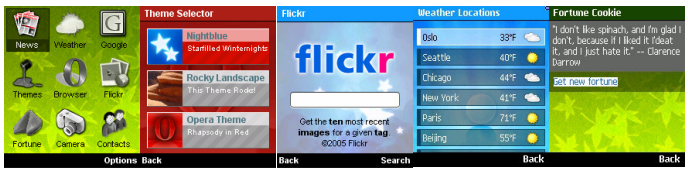






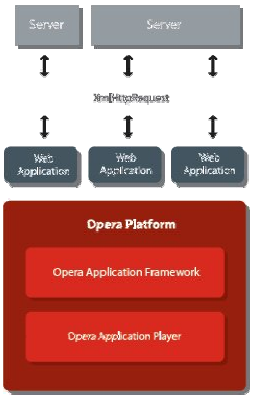
## Hybrid approach: Opera Platform

- AJAX development on mobile devices is possible with the Opera Platform, code named Freedom
  - Based on well-known Web Technologies such as HTML, CSS and JavaScript http://www.opera.com/platform (homepage)
    - http://my.opera.com/operaplatform/links/ (documentation & tools)
- Features:
  - Enables integration between:
    - Handheld devices' local applications
    - Opera Browser environment
    - Operator's online content
  - Allows operators to push their content and services on the handset
  - Hybrid between Browsing and Smart Client apps



## Opera Platform SDK

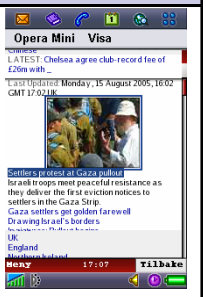
- The Opera Platform architecture consists of three parts:
  - **Application Player**, an extended version of the Opera browser, provides web applications with access to native phone functionality such as messaging, calendar, battery and signal status.
  - **Application Framework**, which supports interaction between installed web applications.
    - It also offers predefined UI elements, such as menu systems and dialog boxes to ease application development, according to Opera.
  - **Web applications** created with open standard technologies such as HTML, CSS and JavaScript.
    - Access the phone's functionality through the Opera Platform DOM interface
    - Communicate with servers using `XMLHttpRequest`






## Opera Mini: Customizing Web Rendering

- Opera Mini is a Java ME web browser for mobile devices
  - Versions for low and high memory phones
- Fetches content through a proxy that runs the layout engine of the Opera desktop browser
  - Proxy uses Small Rendering Technology to reformat webpages
  - Content compressed 70-90% and delivered in OBML




## Google Maps Mashups & Where 2.0

- There are many originals mash-ups out there based on GoogleMaps:
  - http://googlemapsmania.blogspot.com/
- Some examples:
  - Maplandia.com News Center (http://www.maplandia.com/news/)
  - Real-time location of Dublin commuter trains (http://dartmaps.mackers.com/)
  - HousingMaps gets the locations of properties for sale or rent from Craigslist on the fly
  - Cheap Gas (http://www.mywikimap.com/)
  - Chicagocrime.org that taps into Google Maps to display where crimes occur in Chicago (http://www.chicagocrime.org/map/)
- Where 2.0 is a conference that gathers people on location-based web apps



## An Interesting Mobile Mash-up Scenario

- "I was sitting in the back of a cab one Saturday evening. I was using Kmaps (http://kmaps.ulocate.com/) to pull up listings of the closest restaurants. I choose one based on user posted reviews, directed the driver using an attached Google Maps mash-up, and upon arrival, tagged the map with my precise location so my friend could meet me. My friend wanted to know what the restaurant was serving before he decided to come so I snapped a picture of the menu, uploaded a quick picture and note to my blog with my tagged location and was immediately called by a 3rd friend who had seen the blog post and wanted to come as well"
  - http://marketspaceadvisory.typepad.com/marketspace_advisor/2006/02/adventures_with.html

## Ubiquitous Web & Geofolksonomies

- Ubiquitous Web (UW) = pervasive web infrastructure in which all physical objects are resources accessible by URIs, providing information and services that enrich users' experiences in their physical context as the web does in the cyberspace
  - Apps dynamically adapt to the user's needs, device capabilities and environmental conditions.
- Making UW reality:
  - Social tagging is a very efficient way of categorizing resources on the web, e.g. del.icio.us
  - GeoFolksonomies = social tagging of geographic locations, e.g. Tagzania, mobile version?
  - AwareFolksonomies = users may associate objects with contextual attributes and metadata
    - If the contextual attributes are met the metadata is made available



## Deusto Sentient Graffiti

- We want to make Ubiquitous Web reality through an Aware Folksonomy:
  - Mixing social tags, location, profiles, preferences, Semantic Web
- **Goal:** enable the edition, discovery and navigation of virtual post-it notes placed in the Deusto campus
  - A post-it note is an XML document with some contextual attributes (profile of creator, location, time interval, attributes (tags))
  - An inference engine will in real-time match the mobile device owner's context against the available post-it notes at his location
  - Should work both indoors (RFID) and outdoors (GPS)
  - Should enable transparent handoffs between Wi-Fi and GPRS
- Hardware requirements: Wi-Fi, GPRS/UMTS, GPS, RFID

## Google Map Extension Hacks

- Web Map Service (WMS) produces a map from a URL
  - map = portrayal of geographic information as a digital **transparent** image file (.GIF o .PNG)
  - URL indicates what information is to be shown on the map:
    - portion of the earth
    - desired coordinate reference system
    - output image width and height
  - Specification managed by Open Geospatial Consortium (OGC)
    - http://portal.opengeospatial.org/files/?artifact_id=5316
- Overlay Custom Maps over Google Maps
  - http://blog.kylemulka.com/?p=287
  - http://johndeck.blogspot.com/
- Automatic Tile Cutter (retrieves .PNGs from Google Maps Tile Server)
  - http://mapki.com/index.php?title=Automatic_Tile_Cutter
- Geocoders: assigning geographic coordinates (e.g. latitude-longitude) to street addresses



## Conclusion

- Arrival of Web 2.0 dynamic asynchronous interfaces to mobile devices will make us forget WAP's bad experience
- Mobile Mash-ups can foster up-stream data usage
  - Mobile operator's can significantly increase ARPU
- Mobile Mash-ups can be really helpful and are finally reality with available mature platforms
  - Hybrid browsing/smart client platforms seem the future
- What about Mobile Context-Aware Mash-ups research opportunities?

## References


- Mobile web 2.0: AJAX for mobile devices
  - Ajit Jaokar Blog: http://opengardensblog.futuretext.com
- Russel Beattie Notebook
  - http://www.russellbeattie.com/notebook/
- Mobility 2.0
  - C. Enrique Ortiz' Mobility Weblog – http://www.cenriqueortiz.com/weblog/
- Annotate your own multimedia map
  - http://www.engadget.com/2005/03/08/how-to-make-your-own-annotated-multimedia-google-map/
- Simple Google Maps example in Python
  - http://gpswanderer.blogspot.com/2005_04_01_gpswanderer_archive.html
- Python for Series 60
  - http://www.forum.nokia.com/python
- XML APIs for creating mashups
  - http://www.programmableweb.com
- W3C MWI & Mobile Web 2.0
  - http://blog.webservices.or.kr/hollobit/presentation/ngweb2006-hollobit.pdf




Universidad de Deusto
Deustuko Unibertsitatea

mobility research lab

# 4. – Web Semántica


Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es

## Web Semántica

- Problema de la Web Actual:
  - El significado de la web no es accesible a máquinas
- **Web Semántica** → crea un medio universal de intercambio de información, aportando semántica a los documentos en la web
  - Añade significado comprensible por ordenadores a la Web
  - Usa técnicas inteligentes que explotan esa semántica
  - Liderada por Tim Berners-Lee del W3C



## Web vs. Web Semántica

- La Web permite acceder a todo tipo de información fácilmente
  - Los motores de búsqueda nos ayudan a encontrar información
    - Pero, los resultados devueltos no son siempre correctos
- Web Actual:
  - Colección de documentos ligados por hipervínculos
    - El texto de un enlace es una palabra clave que hace referencia a otros documentos
  - Útil para describir, con un énfasis en presentación visual, bloques de texto, imágenes y formularios
    - Pero, una máquina no puede extraer semántica de listado de productos en una página web

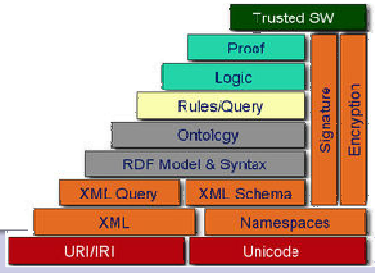
## Web vs. Web Semántica

- La Web Semántica pretende crear un medio universal para intercambiar información y relacionar conceptos
- Web Semántica:
  - Conjunto de conceptos ligados a otros conceptos
    - RDF y OWL permiten indicar cómo un concepto se relaciona con otro
      - Añaden significado al contenido, facilitando el uso autónomo de la web por ordenadores



## Semantic Web Stack

- La Web Semántica está compuesta de:
  - XML, sintaxis para documentos estructurados
  - XML Schema, restringe la estructura de documentos XML
  - RDF es un modelo de datos que hace referencia a objetos y sus relaciones
  - RDF Schema, vocabulario para definir propiedades y clases de recursos RDF
  - OWL, añade más vocabulario que RDFS, relaciones entre clases, cardinalidad, igualdad …



## Propósito Web Semántica

- Mejorar la usabilidad y utilidad de la Web y sus recursos interconectados, mediante:
  - **Anotación semántica** → documentos mejorados con metadatos semánticos leíbles por máquinas o metadatos representando hechos sobre cualquier concepto (lugar, persona, etc.)
  - **Ontologías** → vocabularios de metadatos comunes y mapas entre ellos que guían marcado de documentos para que los agentes puedan utilizar la semántica suministrada
    - Autor de la página o autor del libro
  - **Agentes** → realizan tareas para usuarios utilizando estos metadatos (shopbot)
  - **Infrastructura** → Servicios Web que suministren información a agentes (Trust Service – informa calidad información)
- Los principales facilitadores de la Web Semántica son URIs, XML, XML NameSpaces y RDF



## Resource Description Framework (RDF)

- Modelo basado en la definición de sentencias acerca de recursos en formato:
  - Sujeto-predicado-objeto ⇔ RDF Triple
    - Sujeto: recurso descrito
    - Predicado: relación entre sujeto y objeto
    - Objeto: el valor asociado al sujeto

## Resource Description Framework (RDF)

- Modelo para describir pseudo-grafos dirigidos etiquetados:
  - Dirigido → cada arco tiene una dirección
  - Etiquetado → cada arco tiene una etiqueta
  - Pseudo-grafo → puede haber más de un arco entre nodos
- Un modelo RDF es una colección no ordenada de sentencias o ternas, con:
  - Sujeto (nodo)
  - Predicado (arco)
  - Objeto (nodo)




## Resource Description Framework (RDF)

- Un grafo RDF crea una web de conceptos
  - Realiza aserciones sobre relaciones lógicas entre entidades
- Información en RDF puede ligarse con grafos en otros lugares
  - Mediante software se pueden realizar inferencias
    - Lenguajes de consulta sobre triple stores como SPARQL
- Mediante RDF hacemos que la información sea procesable por máquinas
  - Agentes software pueden guardar, intercambiar y utilizar metadatos sobre recursos en la web
- **Ontología** → jerarquía de términos a utilizar en etiquetado de recursos

## Conceptos Fundamentales RDF

- URIs → mecanismo utilizado por RDF para identificar unívocamente conceptos
- Literals → objetos con contenido real en vez de URIs
  - `this line --- (was written on) --> "20060603"[date]`
- Reification → uso de sentencias como sujeto de otras sentencias

```
this news ---(has category)---> "semantic web"
[this news ---(has category)---> "semantic web"] ---(added by)---> stefano
```



## Formatos RDF

- El mecanismo de serialización oficial de RDF es RDF/XML
  - Tipo MIME es `application/rdf+xml`
  - No es muy leíble
- Ej. Expresión en RDF de "Artículo en Wikipedia sobre Maradona"

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description
    rdf:about="http://en.wikipedia.org/wiki/Maradona">
     <dc:title>Diego Armando Maradona</dc:title>
     <dc:publisher>Wikipedia</dc:publisher>
  </rdf:Description>
</rdf:RDF>
```

## Formatos RDF

- N3 es una notación que permite definir ternas o relaciones "sujeto-predicado(verbo)-objeto", de una manera más concisa
- Ej1:

```
<http://en.wikipedia.org/wiki/Maradona>
  <http://purl.org/dc/elements/1.1/title> "Diego
  Armando Maradona" .
```

- Ej2:

```
@prefix wcj: http://example.org/wcjava/uri/ .
wcj:programmers wcj:enjoy wcj:java .
```



## RDFS

- Extensión semántica de RDF
  - Conjunto se de sentencias que definen clases y propiedades
- Sirve para decir cosas como:
  - Esta URI debería ser considerada como una clase (`rdfs:Class`) o propiedad (`rdfs:Property`)
  - Indicar si una etiqueta (`rfds:label`) o comentario (`rdfs:comment`) es leible por humanos
  - Esta URL está definida por (`rfds:DefinedBy`)
  - Esta clase es subclase de (`rdfs:subClassOf`)
  - Esta propiedad es subpropiedad de (`rfds:subPropertyOf`)
  - Esta propiedad conecta esta clase de sujetos (`rdfs:domain`) con esta clase de objetos (`rdfs:range`)

## OWL

- Acrónimo de Web Ontology Language, derivado de DAML+OIL
- Extensión de vocabulario a RDF → añade más metadatos a los nodos, clases y propiedades para razonar sobre ellas
  - OWL es a RDF lo que XSL es a XML
- Algunos ejemplos:
  - Esta propiedad es transitiva (`owl:TransitiveProperty`)
  - Esta propiedad es simétrica (`owl:SymmetricProperty`)
  - Esta propiedad es inversa a esta otra (`owl:InverseOf`)
  - Equivalente a (`owl:equivalentProperty`)
  - Lo mismo que (`owl:sameAs`)
  - Esta propiedad puede aparecer sólo una vez (`owl:cardinality`)



## Ejemplo OWL

- Supongamos el siguiente modelo RDF:

```
<http://www.betaversion.org/~stefano/> -(is author of)->
   http://www.betaversion.org/~stefano/linotype/
<http://www.apache.org/~stefano/> -(is author of)->
   http://www.apache.org/~stefano/agora/
<http://web.mit.edu/people/stefanom/> -(is author of)->
   <http://simile.mit.edu/gadget/>
```

- Aunque pertenecen al mismo autor, no están relacionadas entre ellas, con la ayuda de OWL podemos mapear estas URIs

```
<http://www.apache.org/~stefano/> -(owl:sameAs)->
   <http://www.betaversion.org/~stefano/>
<http://web.mit.edu/people/stefanom/> -(owl:sameAs)->
   <http://www.betaversion.org/~stefano/>
```

- Si mezclamos ambos modelos y ejecutamos un razonador podríamos responder a "dime todo lo que ha escrito "http://www.betaversion.org/~stefano":

```
http://www.betaversion.org/~stefano/> -(is author of)->
   http://www.apache.org/~stefano/agora/
<http://www.betaversion.org/~stefano/> -(is author of)->
   <http://simile.mit.edu/gadget/>
```

## Lógica e Inferencia

- La lógica es la disciplina que estudia los principios de razonamiento
- Los razonadores automáticos deducen conclusiones a partir del conocimiento
- Aplicado a ontologías, puede:
  - Descubrir conocimiento ontológico implícito
  - Descubrir relaciones e inconsistencias inesperadas



## Críticas a la Web Semántica

- Para funcionar requiere hacer referencia a un vocabulario gigante y centralizado
- Debido a reification, las relaciones entre conceptos pueden ser tan extensas que se incrementa la computabilidad
- Aconsejable utilizar OWL Lite (NO DL, Full) para no imponer demasiados requerimientos computacionales

## Herramientas Web Semántica

- Jena para Java y CWM para Python
- Permiten transformar entre sintaxis N3 y RDF/XML
- Realizan inferencias



## Referencias Web Semántica

- RDF:
  - http://www.javaworld.com/javaworld/jw-12-2005/jw-1205-wicked_p.html
- OWL:
  - A No-Nonsense Guide to Semantic Web Specs for XML People
    - http://www.betaversion.org/~stefano/linotype/news/57/

Universidad de Deusto
Deustuko Unibertsitatea

mobility research lab

# 5. Programming Semantic Web with JENA

Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es

## Jena: a Framework for Semantic Web

Universidad de Deusto
Deustuko Unibertsitatea

- Jena Semantic Web Framework
  - http://jena.sourceforge.net/
- Enables among other things:
  - Create and populate RDF models
  - Persist models to a database
  - Query models programmatically with RDQL y SPARQL
  - Reasoning over ontologies
- Currently in versión 2.4
  - Download from: http://jena.sourceforge.net/downloads.html

## Create a RDF Model

- The `ModelFactory` class enables the creation of models:
  - `ModelFactory.createDefaultModel()`, allows the creation of an in-memory model
    - Returns a `Model` instance over which you can create `Resources`
  - `Model.createProperty()` allow relationship creation
  - To add statements for a model use: `Resource.addProperty()` or `Model.createStatement()` and `Model.add()`
- In Jena a statement is composed of:
  - A subject in the form of a `Resource`
  - A predicate represented by a `Property` class
  - An object, either a `Literal` or `Resource`
- `Resource`, `Property` and `Literal` inherit from `RDFNode`



## Create a RDF Model representing a Family

```
// URI declarations
String familyUri = "http://family/";
String relationshipUri = "http://purl.org/vocab/relationship/";

// Create an empty Model
Model model = ModelFactory.createDefaultModel();

// Create a Resource for each family member, identified by their URI
Resource adam = model.createResource(familyUri+"adam");
Resource beth = model.createResource(familyUri+"beth");
Resource chuck = model.createResource(familyUri+"chuck");
Resource dotty = model.createResource(familyUri+"dotty");
// and so on for other family members

// Create properties for the different types of relationship to represent
Property childOf = model.createProperty(relationshipUri,"childOf");
Property parentOf = model.createProperty(relationshipUri,"parentOf");
Property siblingOf = model.createProperty(relationshipUri,"siblingOf");
Property spouseOf = model.createProperty(relationshipUri,"spouseOf");

// Add properties to adam describing relationships to other family members
adam.addProperty(siblingOf,beth);
adam.addProperty(spouseOf,dotty);
adam.addProperty(parentOf,edward);

// Can also create statements directly . . .
Statement statement = model.createStatement(adam,parentOf,fran);
// but remember to add the created statement to the model
model.add(statement);
```

## Interrogating an RDF Model



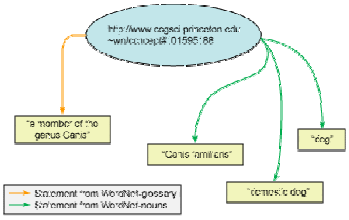
- By means of `listXXX()` method in `Model` and `Resource` interfaces
  - It returns specializations of `java.util.Iterator`
  - El método más genérico es `Model.listStatements(Resource s, Property p, RDFNode o)`
- Examples:
  - `ResIterator parents = model.listSubjectsWithProperty(parentOf);`
    - `Resource person = parents.nextResource();`
  - `NodeIterator moreParents = model.listObjectsOfProperty(childOf);`
  - `StmtIterator moreSiblings = edward.listProperties(siblingOf);`
  - `model.listStatements(adam,null,null);`



## Importing and Persisting Models



- So far we have worked with in-memory models
  - Necessary to persist and retrieve models
- Easiest solution:
  - Model.read()
  - Model.write()
- More sophisticated over RDBMS:
  - MySQL
- Example: `ImportWordnet.java`
  - Imports the following RDF models into a single Jena model:
    - WordNet-nouns
    - WordNet-glossary
    - WordNet-hyponyms

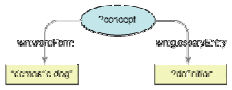


## RDF Data Query Language (RDQL)

Universidad de Deusto
Deustuko Unibertsitatea

- RDQL is a query language for RDF
  - Allows complex queries to be expressed concisely
    - A query engine performing the hard work of accessing the data model
- Example:

```
SELECT ?definition
WHERE
   (?concept, <wn:wordForm>, "domestic dog"),
   (?concept, <wn:glossaryEntry>, ?definition)
USING
   wn FOR <http://www.cogsci.princeton.edu/~wn/schema/>
```



- Another example:

```
SELECT
      ?wordform, ?definition

WHERE
      (?firstconcept, <wn:wordForm>, "panther"),
      (?secondconcept, <wn:wordForm>, "tiger"),

      (?firstconcept, <wn:hyponymOf>, ?hypernym),
      (?secondconcept, <wn:hyponymOf>, ?hypernym),

      (?hypernym, <wn:wordForm>, ?wordform),
      (?hypernym, <wn:glossaryEntry>, ?definition)

USING
      wn FOR <http://www.cogsci.princeton.edu/~wn/schema/>
```



## Using RDQL

Universidad de Deusto
Deustuko Unibertsitatea

- Need to `import com.hp.hpl.jena.rdql`
- To create a query instantiate `Query` and pass as a `String` the query
- Create `QueryEngine` and invoke `QueryEngine.exec(Query)`
- Variables can be bound to values through a `ResultBinding` object
- Example:

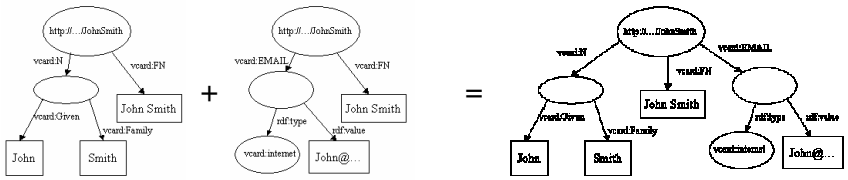
```
// Create a new query passing a String containing the RDQL to execute, containing
   variables x and y
Query query = new Query(queryString);

// Set the model to run the query against
query.setSource(model);

// A ResultBinding specifies mappings between query variables and values
ResultBindingImpl initialBinding = new ResultBindingImpl() ;

// Bind the query's first variable to a resource
Resource someResource = getSomeResource();
initialBinding.add("x", someResource);

// Bind the query's second variable to a literal value
RDFNode foo = model.createLiteral("bar");
initialBinding.add("y", foo);

// Use the query to create a query engine
QueryEngine qe = new QueryEngine(query);

// Use the query engine to execute the query
QueryResults results = qe.exec();
```

## Operations on Models

- The common set operations:
  - Union (`.union(Model)`) , intersection (`.intersection(Model)`) and difference (`.difference(Model)`)
- Example: union

```
// read the RDF/XML files
model1.read(new InputStreamReader(in1), "");
model2.read(new InputStreamReader(in2), "");
// merge the Models
Model model = model1.union(model2);
// print the Model as RDF/XML
model.write(system.out, "RDF/XML-ABBREV");
```

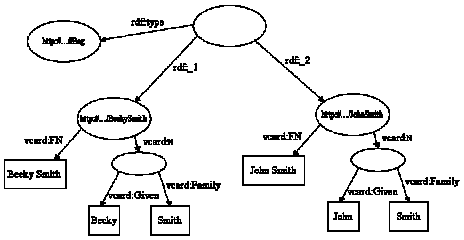
+






## Containers in RDF

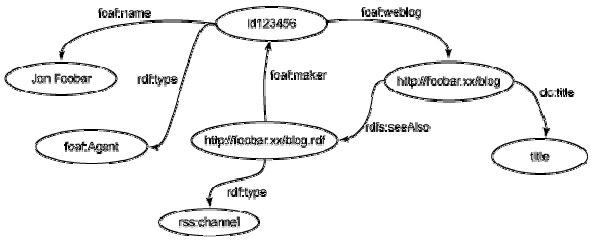
- RDF defines a special kind of resources to represent collections of things:
  - A BAG is an unordered collection
  - An ALT, unordered collection representing alternatives
  - A SEQ is an ordered collection
- A container is represented by a resource with an rdf:type property whose value should be: rdf:Bag, rdf:Alt or rdf:Seq
  - The ordinal properties are denoted by rdf:_N



## SPARQL Protocol And RDF Query Language (SPARQL)



- Builds on previously existing query languages such as rdfDB, RDQL, SeRQL
- In our experimentation with SPARQL 3 RDF graphs will be used, corresponding to blogger.rdf:
  - FOAF graphs describing contributors
  - RSS 1.0 contributor feeds
  - Bloggers graph






## Syntax SPARQL



- SPARQL query to find the URL of a contributor's blog (`david.rq`):

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?url
FROM      <bloggers.rdf>
WHERE  {
    ?contributor foaf:name "Dave Beckett" .
    ?contributor foaf:weblog ?url .
}
```

  - `PREFIX` indicates prefix for FOAF namespace
  - `SELECT` indicates what the query should return
  - `FROM` optional clause indicating the URI of the dataset to use
  - `WHERE` triple patterns expressed in Turtle syntax (graph pattern)
- Test queries from command line with `java jena.sparql`
  - `java jena.sparql --query david.rq`
  - `FROM` clause can be omitted and specified by `--data URL`

## Using SPARQL with JENA

- SPARQL is supported in JENA via ARQ module
  - It also understands RDQL queries
- Need to import package: `com.hp.hpl.jena.query`
- QueryFactory.create() returns a Query object from a file or String
- QueryExecutionFactory.create(query, model) returns a QueryExecution object
- QueryExecution supports varios methods:
  - execSelect() returns a ResultSet
  - Apart from SELECT, you can apply the following types of queries:
    - ASK, DESCRIBE, CONSTRUCT



## Executing a Simple Query with JENA

```
// Open the bloggers RDF graph from the filesystem
InputStream in = new FileInputStream(new File("bloggers.rdf"));

// Create an empty in-memory model and populate it from the graph
Model model = ModelFactory.createMemModelMaker().createModel();
model.read(in,null); // null base URI, since model URIs are absolute
in.close();

// Create a new query
String queryString =
    "PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
    "SELECT ?url " +
    "WHERE {" +
    "      ?contributor foaf:name \"Jon Foobar\" . " +
    "      ?contributor foaf:weblog ?url . " +
    "      }";

Query query = QueryFactory.create(queryString);

// Execute the query and obtain results
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();

// Output query results
ResultSetFormatter.out(System.out, results, query);

// Important - free up resources used running the query
qe.close();
```

## Refining SPARQL Queries

- DISTINCT used with SELECT

  `SELECT DISTINCT`
- Used with SELECT clause:
  - `LIMIT n` → shows upto n results
  - `OFFSET n` → ignores first n results
  - `ORDER BY var` → sorts results by normal ordering
    - `ASC(?var)` and `DESC(?var)`



## More complex queries

- RDF is often used to represent *semi-structured* data. This means that two nodes of the same type in a model may have different sets of properties.
- Optional matches

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?depiction
WHERE {
   ?person foaf:name ?name .
   OPTIONAL {
      ?person foaf:depiction ?depiction .
   } .
}
```

- Alternative matches:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name ?mbox
WHERE {
  ?person foaf:name ?name .
  {
      { ?person foaf:mbox ?mbox } UNION { ?person foaf:mbox_sha1sum ?mbox
   }
  }
}
```

## More complex queries

- Using filters

```
PREFIX rss:   <http://purl.org/rss/1.0/>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX  dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?item_title ?pub_date
WHERE {
   ?item rss:title ?item_title .
   ?item dc:date ?pub_date .
   FILTER xsd:dateTime(?pub_date) >= "2005-04-
  01T00:00:00Z"^^xsd:dateTime &&
         xsd:dateTime(?pub_date) < "2005-05-
  01T00:00:00Z"^^xsd:dateTime
}
```



## Working with Multiple Graphs

- The model after the FROM clause is the background graph
- Several graphs can be specified after the FROM NAMED <URI> clause
- Named graphs are used within a SPARQL query with the GRAPH keyword
- Example: find people found in two named FOAF graphs

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name
FROM NAMED <jon-foaf.rdf>
FROM NAMED <liz-foaf.rdf>
WHERE {
  GRAPH <jon-foaf.rdf> {
    ?x rdf:type foaf:Person .
    ?x foaf:name ?name .
  } .
  GRAPH <liz-foaf.rdf> {
    ?y rdf:type foaf:Person .
    ?y foaf:name ?name .
  } .
}
```

## Working with Multiple Graphs

- Example: determining which graph describes different people

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
  syntax-ns#>
SELECT ?name ?graph_uri
FROM NAMED <jon-foaf.rdf>
FROM NAMED <liz-foaf.rdf>
WHERE {
    GRAPH ?graph_uri {
      ?x rdf:type foaf:Person .
      ?x foaf:name ?name .
    }
}
```



## Combining Background Data and Named Graphs

- Example: getting a personalized live PlanetRDF feed

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rss:  <http://purl.org/rss/1.0/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?title ?known_name ?link
FROM <http://planetrdf.com/index.rdf>
FROM NAMED <phil-foaf.rdf>
WHERE {
   GRAPH <phil-foaf.rdf> {
     ?me foaf:name "Phil McCarthy" .
     ?me foaf:knows ?known_person .
     ?known_person foaf:name ?known_name .
   } .

   ?item dc:creator ?known_name .
   ?item rss:title ?title .
   ?item rss:link ?link .
   ?item dc:date ?date.
}
ORDER BY DESC(?date) LIMIT 10
```

## Ontologies in Jena

- They are treated a special type of RDF model, `OntModel`
  - This interface allows to manipulate programmatically an ontology:
    - Create classes, property restrictions
- Alternatively:
  - Statements meaning semantic restrictions can be added to an RDF model
  - Merge an ontology model with a data model with `Model.union()`
- Examples:

```
// Make a new model to act as an OWL ontology for WordNet
OntModel wnOntology = ModelFactory.createOntologyModel();

// Use OntModel's convenience method to describe
// WordNet's hyponymOf property as transitive
wnOntology.createTransitiveProperty(WordnetVocab.hyponymOf.getU
   RI());

// Alternatively, just add a statement to the underlying model
   to express that hyponymOf is of type TransitiveProperty
wnOntology.add(WordnetVocab.hyponymOf, RDF.type,
   OWL.TransitiveProperty);
```



## Inference in Jena

- Given an ontology and a model Jena can inference statements not explicitly expressed
- `OWLReasoner` applies OWL ontologies over a model to reason
- Example:

```
// Make a new model to act as an OWL ontology for WordNet
OntModel wnOntology = ModelFactory.createOntologyModel();
...
// Get a reference to the WordNet plants model
ModelMaker maker = ModelFactory.createModelRDBMaker(connection);
Model model = maker.openModel("wordnet-plants",true);

// Create an OWL reasoner
Reasoner owlReasoner = ReasonerRegistry.getOWLReasoner();

// Bind the reasoner to the WordNet ontology model
Reasoner wnReasoner = owlReasoner.bindSchema(wnOntology);

// Use the reasoner to create an inference model
InfModel infModel = ModelFactory.createInfModel(wnReasoner, model);

// Set the inference model as the source of the query
query.setSource(infModel);

// Execute the query as normal
QueryEngine qe = new QueryEngine(query);
QueryResults results = qe.exec(initialBinding);
```

## Jena Generic Rule Engine



- JENA Rule Engine supports rule-based inference over RDF graphs using:
  - Forward-chaining
  - Backward-chaining
  - Hybrid execution engine
- Implemented as class: `com.hp.hpl.jena.reasoner.rulesys.GenericRuleReasoner`
  - Requieres a RuleSet to define its behaviour
    - A set of com.hp.hpl.jena.reasoner.rulesys.Rule



## Jena Generic Rule Engine



- Two built-in rule engines:
  - Forward chaining RETE engine and
  - One tabled datalog engine
- Some example rules:

```
[allID: (?C rdf:type owl:Restriction), (?C
  owl:onProperty ?P), (?C owl:allValuesFrom ?D)  ->
  (?C owl:equivalentClass all(?P, ?D)) ]
[all2: (?C rdfs:subClassOf all(?P, ?D)) ->
  print('Rule for ', ?C) [all1b: (?Y rdf:type ?D)
  <- (?X ?P ?Y), (?X rdf:type ?C) ] ]
[max1: (?A rdf:type max(?P, 1)), (?A ?P ?B), (?A ?P
  ?C) -> (?B owl:sameAs ?C) ]
```

- Documentation: http://jena.sourceforge.net/inference/index.html#rules

## Semantic Web & Pervasive Spaces Research

Universidad de Deusto
Deustuko Unibertsitatea

- Semantic Space: An Infrastructure for Smart Spaces
  - They use Semantic Web to add the following features to a Space:
    - Explicit representation → adds semantics to raw data
    - Context Querying → enables answering to context queries
    - Context Reasoning → allows an app to reason about the space situation
  - IEEE Pervasive Computing, July-September 2004
    - Xiaohang Wang, Jin Song Dong, ChungYau Chin, and Sanka Ravipriya Hettiarachchi at National University of Singapore and Daqing Zhang at Institute for Infocomm Research,Singapore
      - http://www.comp.nus.edu.sg/~dongjs/papers/pervasive04.pdf

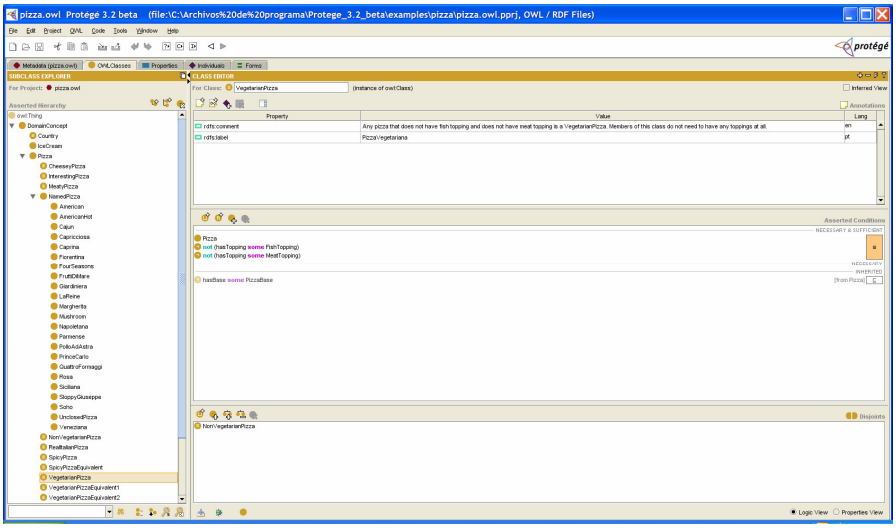


## Generating Ontologies with Protégé

Universidad de Deusto
Deustuko Unibertsitatea

- Protége is a free, open source ontology editor and knowledge base framework.
  - Implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats.
    - An ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the meaning of terms used in the vocabulary.
  - Supports two ways of modeling ontologies:
    - Protégé-Frames and Protégé-OWL
  - Downloadable from: http://protege.stanford.edu/
- W3C Ontology Definition:
  - *"An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms"*
- The Protégé OWL Tutorial:
  - http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

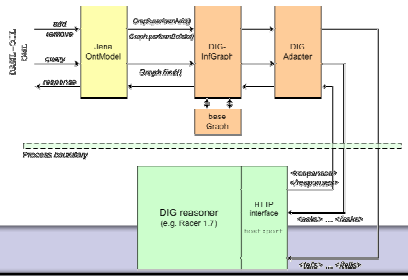
## Pizza Ontology in Protégé





## Third Party Reasoners

- You can attach external reasoners to Jena
  - Facilitated through the DIG (Description Login Implementation Group) interface
- Racer, FaCT and Pellet
  - http://www.racer-systems.com/
  - http://www.cs.man.ac.uk/~horrocks/FaCT/
  - http://www.mindswap.org/2003/pellet
- More info: http://jena.sourceforge.net/how-to/dig-reasoner.html



## Semantic Web Rule Language (SWRL)

- Aims to be the standard rule language of the Semantic Web
  - Previous attempts RuleML, Metalog, ISO Prolog
- Provides the ability to write horn-like rules expressed in terms of OWL concepts
- Rules can be used to infer new knowledge from existing OWL knowledge bases
- Example:

```
hasBrother(?x1, ?x2) ^ hasAge(?x1, ?age1) ^
  hasAge(?x2, ?age2) ^ swrlb:subtract(10, ?age2,
  ?age1) => hasDecadeOlderBrother(?x1, ?x2)
```

- Specificacition: http://www.w3.org/Submission/SWRL/
- Implementation: http://machine-knows.etri.re.kr/wiki/BossamRuleEngine_2fEnglishPage



## References Semantic Web

- RDF:
  - http://www.javaworld.com/javaworld/jw-12-2005/jw-1205-wicked_p.html
- OWL:
  - A No-Nonsense Guide to Semantic Web Specs for XML People
    - http://www.betaversion.org/~stefano/linotype/news/57/
- JENA
  - Introduction to Jena
    - http://www-128.ibm.com/developerworks/java/library/j-jena/
  - Search RDF data with SPARQL
    - http://www-128.ibm.com/developerworks/library/j-sparql/


Universidad de Deusto
Deustuko Unibertsitatea
MORE LAB
mobility research lab
Web Semántica Práctica
Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspesonales.deusto.es/dipina
http://www.morelab.deusto.es
http://www.ctme.deusto.es