multicast
internet
microfluidic
barcode
hypertext
http
playlist
timeline
mems
healthcare
membership
wlan
hosting
transgene
transgenic
netlist
networked
voxel
chimeric
unicast
voxels
mpls
hypervisor
voicemail
hosted
providers
nanotube
bitline
macroblock
mimo
networking
gesture
robotic
xbd
voip
spam
failover
virtualized
loyalty
pfet
avatar
webpage
vlan
antisense
metadata
ethernet
rfid
bitmap
ldpc
gui
cached
hyperlink
ofdm
cdma
notifications
instantiating
instantiated
plasmid
subpixel
heterocyclylalkyl
wtru
venue
cytokine
cqi
urls
nanowires
nanostructures
nanostructure
spreadsheet
timestamps
unencrypted
mrna
gnss
nanoscale
chipset
asic
nfet
sql
harq
middleware
archiving
placeholder
subtitle
sirna
pstn
lut
rssi
mtj
upload
uwb
xae
nonce
hiv
applet
gprs
imd
mram
websites
html
macroblocks
namespace
ciphertext
telematics
markup
wirelessly
oled
micromirror
wdm
malware
datagram
microelectromechanical
graphene
ontology
website
browser
uploading
url
timestamp
humanized
widget
touchscreen
timeslots
nanowire
credentials
laptop
bluetooth
authoring
parsed
caching
fpga
nanoparticle
virtualization
biosensor
xml
workflow
tangibly
legacy
femtocell
nanotubes
damascene
bitlines
mbms
xbc

10
웹 2.0 서비스의 구현기술

10 Technical Implementation of Web 2.0 Services

차례

10.1 콘텐츠 유통을 위한 RSS 기술

10.1.1 데이터 유통채널의 변화와 RSS의 등장

10.1.2 RSS의 발전 과정과 작동 원리

10.1.3 RSS의 장점과 활용 분야

10.2 콘텐츠 출판을 위한 블로그와 팟캐스팅

10.2.1 블로그의 개념과 작동 원리

10.2.2 아이튠즈(iTunes)와 팟캐스팅 (Podcasting)

10.3 시맨틱웹으로 발전하는 태깅과 마이크로포맷

10.3.1 문서에 의미를 줄 수 있는 소셜 태깅

10.3.2 마이크로포맷

10.4 웹 인터페이스의 진화, RIA와 Ajax

10.4.1 RIA의 등장 배경과 발전 방향

10.4.2 Ajax의 개념 및 특징

10.4.3 Ajax의 구현 사례

10.5 오픈 API와 매쉬업 서비스

10.5.1 오픈 API와 매쉬업 서비스의 개념

10.5.2 매쉬업 서비스의 종류

10.5.3 매쉬업 사이트 구축 사례

데이터 유통채널의 변화와 RSS의 등장 ①

- 콘텐츠 신디케이션(Contents Syndication)
  - 콘텐츠 생산자(Contents Provider)가 제공하는 콘텐츠를 다른 수요자에게 연결해주는 콘텐츠 배급 및 유통을 위한 중개사업
  - 웹 신디케이터(Web Syndicator)
    - 웹 콘텐츠를 수집하여 다른 여러 웹사이트 운영자에게 제공
  - 예) 영화유통의 배급업체, 뉴스 유통을 위한 배급
- 콘텐츠 유통 채널에도 변화
  - 온라인 웹 환경에서 단순히 웹사이트 게시로는 부족
    - 새로운 콘텐츠에 대한 정보를 사용자에게 별도로 제공해야
  - 이메일이나 메시징 방식
    - 사용자의 메일 주소를 등록한 후 정기적으로 정보 제공
    - 필요한 콘텐츠 정보만 골라내는 것도 귀찮은 작업

데이터 유통채널의 변화와 RSS의 등장 ②

- 웹피드(Web Feed) 개념의 등장
  - 콘텐츠의 정보를 사용자에게 전달하기 위한 웹문서 파일 형식
    - RSS가 가장 널리 사용되는 형식
    - 제공하고 있는 콘텐츠의 목록 및 개요 등 관련 정보를 포함
  - 콘텐츠 구독기
    - 웹피드의 주소를 등록한 후 최신 정보를 제공
    - 등록된 피드 파일에서부터 주기적으로 갱신 정보를 찾아와서 보여줌

(a) 이메일을 이용한 배포 방식

(b) 웹피드에 의한 배포 방식

데이터 유통채널의 변화와 RSS의 등장 ③

- RSS
  - 'RDF Site Summary' 혹은 'Really Simple Syndication'
    - 웹사이트에서 콘텐츠의 갱신 정보를 게시해주는 웹피드 표준형식
    - 표준 XML 파일 형식으로 콘텐츠의 요약 정보와 메타데이터를 포함
  - 기본 개념

    1) RSS 버튼을 통해 RSS 파일 주소를 자신의 리더에 등록

    2) RSS 리더는 등록된 주소의 RSS 피드에 새로 갱신된 정보를 요청

    3) RSS 리더가 등록된 피드 주소로부터 갱신된 내용만을 취합

    4) 갱신된 콘텐츠의 내용을 보려면 연결되어 있는 링크로 직접 방문

RSS의 발전 과정과 작동 원리 ①

- RSS의 발전 과정
  - 1990년 중반 마이크로소프트의 CDF(Channel Definition Format)
    - 미디어 데이터의 정보를 인터넷 채널을 통해 배포
  - 애플 MCF(Meta Contents Framework): 메타데이터 표현
  - 넷스케이프 RDF(Resource Description Framework) 개발
    - 일반적인 메타데이터 표현언어
    - 시맨틱웹(Semantic Web) 발전에도 기초
    - RDF 사용 예

```
<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/Steve_Jobs">
        <dc:title>Steve Jobs</dc:title>
        <dc:publisher>Lim Media</dc:publisher>
    </rdf:Description>
</rdf:RDF>
```

RSS의 발전 과정과 작동 원리 ②

- RSS 피드 파일 형식
  - RSS 0.9 (RDF Site Summary 혹은 Rich Site Summary)
    - 1999년 넷스케이프 사에서 XML 기반의 RDF로 표현
  - RSS 1.0
    - 2000년 RSS-DEV 워킹그룹, RDF에 기반 메타데이터 표현을 명확히
  - RSS 2.0 (Really Simple Syndication)
    - UserLand에서 0.9 버전을 지속적으로 발전시켜, 2002년 RSS 최종판
  - Atom 1.0
    - 웹컨소시엄에서는 웹피드의 표준화를 위해 2003년 Atom 개발
  - 현재 RSS 1.0, RSS 2.0, Atom 1.0 혼재되어 사용

RSS의 발전 과정과 작동 원리 ③

- RSS 2.0 피드 파일의 구성
  - 채널(<channel>)과 아이템(<item>) 정보로 구성
  - <channel>
    - 사이트에 대한 헤더 정보
    - <title>,<link>,<description>
    - 그 외에 16가지 부가적인 요소
  - <item>
    - 각각의 글에 대한 요약 정보
    - <title>, <description>
    - <link> 등 7가지 부가적인 요소

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
<channel>
  <title>W3Schools Home Page</title>
  <link>http://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>http://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>
</rss>
```

채널 정보

아이템 정보

아이템 정보

RSS의 발전 과정과 작동 원리 ④

- RSS 버튼
  - RSS 피드를 제공한다는 사실과 피드의 주소를 표시
  - 콘텐츠의 구분에 따라 여러 개의 RSS 피드 제공이 가능

RSS의 발전 과정과 작동 원리 ⑤

- RSS 리더
  - RSS 피드를 읽어주는 프로그램
    - 웹페이지를 볼 때 웹브라우저 이용, RSS 피드 내용을 볼 때 리더 이용
  - RSS 리더의 사용
    - 콘텐츠 공급 사이트에서 제공하는 RSS 피드 주소 등록 후
    - 주기적으로 방문하여 RSS 피드를 파싱하여 그 결과를 화면에 출력
  - 대표적으로 Google Reader, My Yahoo, Bloglines, 한RSS 등
    - 최근 웹브라우저에서도 RSS 피드의 리더 기능을 지원

(a) 한RSS를 이용하는 경우

(b) 웹브라우저를 이용하는 경우

RSS의 장점과 활용 분야

- RSS의 장점
  - 선택적인 콘텐츠 구독이 가능
  - 빠르고 편리하게 콘텐츠 정보 접근: 동시에 여러 채널 구독 가능
  - 과거의 구독 기록도 보관 가능
  - 콘텐츠의 재사용성이 증가
- 활용 사례
  - 뉴스 서비스 등 주기적으로 변경되는 정보를 제공하는 사이트
    - 음악이나 멀티미디어 콘텐츠를 배포하는 팟캐스팅
    - 콘텐츠 상품거래 정보, 날씨나 환율정보 제공, 이벤트 정보 제공 등
  - 광고 삽입을 통해 기업 홍보 및 마케팅 수단으로도 활용
  - 일정 공유라든지 쿠폰 발행 등의 애플리케이션에서도 활용
  - 교육 콘텐츠 분야: 교재 제공, 공지사항, 과제제출 등에 활용

블로그의 개념과 작동 원리 ①

- 블로그(Blog) 개념
  - 1997.11 존 바거(John Barger), web과 log(기록)의 합성어
    - 블로거(Blogger)가 알리고 싶은 생각이나 주장을 웹페이지에 일기처럼 적어 놓고, 다른 사람도 볼 수 있게 나열해 놓은 글들의 모음
    - 게시판과는 달리 한 사람이나 소수의 사람만이 글을 게시
  - 콘텐츠를 남에게 알리는 과정: 콘텐츠 출판(Contents Publishing)
  - 인터넷을 통해 기존 미디어에 못지않은 힘을 발휘: '1인 미디어'
- 블로그의 진화
  - 댓글과 트랙백 기능으로 다양한 의사소통, RSS로 블로그를 구독
  - 모바일 블로그 혹은 마이크로 블로그: 트위터가 대표적
  - 멀티미디어 자료를 포함하도록 진화
    - 포토블로그(Photoblog), mp3 블로그, 비디오블로그(Videoblog 혹은 Vlog), 팟캐스팅 등

블로그의 개념과 작동 원리 ②

- 블로그의 작동 과정
  1) 엔트리(Entry) 작성
     - 블로거가 자신의 글로 가장 기본 단위인 엔트리를 작성
     - 각 엔트리는 하나의 HTML 파일, 이 주소를 고유링크 혹은 퍼머링크
  2) 댓글(Reply) 달기
     - 글 밑에 간단하게 몇 줄로 댓글 혹은 덧글
  3) 트랙백(Trackback)으로 연결하기
     - 원격(Remote) 댓글
     - 글에 대한 의견을 자신의 블로그에 작성
     - 트랙백 핑(Ping)을 보내어 서로의 블로그가 링크로 연결
  4) RSS 피드로 구독하기
     - 블로그 콘텐츠의 효율적 배포

블로그의 개념과 작동 원리 ③

- 트랙백의 원리
  - 작동원리
    - A가 자신의 블로그 글을 작성할 때 트랙백 주소를 공개
    - B가 댓글을 자신의 블로그에 작성 (A의 트랙백 URL을 이용)
    - 그 글에 대해 댓글을 달았다고 트랙백 핑을 보내어 A에게 통지
    - A와 B는 서로 연결되는 링크가 생성
  - 트랙백 핑의 매개변수
    - title(제목), excerpt(초록), url(URL 주소), blog_name(블로그 이름)
  - 일방적 푸시 형태로 원본이 삭제되어도 트랙백은 남는다는 단점

블로그의 개념과 작동 원리 ④

- 블로그의 종류
  - 가입형 블로그
    - 포털사이트나 블로그 전문 웹사이트의 블로그 서비스를 이용
    - 서비스 업체에서 블로그 저장 공간이나 편집 도구 등 모두 제공
    - 커뮤니티 형성이나 콘텐츠 배포를 공유하여 방문자 수 쉽게 증가
    - 사용자가 자유롭게 꾸미고 사용하는 데에는 제한
    - 전문 블로그 서비스로는 티스토리, 이글루스, 블로거닷컴 등
  - 설치형 블로그
    - 자신의 서버에 블로그 편집 소프트웨어를 설치하여 운영
    - 블로그 레이아웃과 다양한 기능을 사용자가 자유롭게 구성
    - 웹 지식 필요, 자신이 서버 관리, 방문자 증가도 스스로 해결해야
    - 설치형 블로그 프로그램으로는 텍스트큐브, 워드프레스, 텍스타일, 무버블타입 등

아이튠즈와 팟캐스팅 ①

- 팟캐스팅
  - 콘텐츠 공급자가 오디오 방송 등의 프로그램을 미디어 파일로 제작하여 웹사이트에 포스팅하는 방법
    - RSS를 이용해서 정기적/자동으로 구독함으로써 방송 전달
  - Podcasting = 아이팟(iPod)+브로드캐스팅(Broadcasting) 합성어
  - 아이튠즈(iTunes) : 대표적인 팟캐스팅 소프트웨어
  - PC나 모바일 기기 등 모든 디지털 미디어 플레이어에 적용 가능
  - 국내 팟캐스팅의 대표적인 사례: '나꼼수'

아이튠즈(iTunes) 실행화면

팟캐스팅 로고

아이튠즈와 팟캐스팅 ②

- 팟캐스팅의 구성요소
  - 에피소드(Episode): 팟캐스터(Podcaster)가 제작한 파일
  - 팟캐쳐(Podcatcher)/수집기(Aggregator): 에피소드의 웹피드를 처리해주는 RSS 리더
- 팟캐스팅의 작동과정
  - 팟캐스터가 작성한 에피소드를 서버에 업로드하고 RSS파일 갱신
  - 청취자는 원하는 팟캐스트를 구독 신청
  - RSS 주소가 등록되어 갱신되는 파일들을 자동으로 다운로드
    - 디지털 미디어 플레이어가 컴퓨터와 연결될 때 자동으로 동기화

아이튠즈와 팟캐스팅 ③

- 팟캐스팅으로 구독
  - 팟캐스트(Podcast) 파일 안에 오디오 파일의 URL 주소를 기술
  - 팟캐스트 파일 사례

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd" version="2.0">
  <channel>
    <title>All About Everything</title>
    <link>http://www.example.com/podcasts/everything/index.html</link>
    <language>en-us</language>
    <copyright>&#x2117; &amp; &#xA9; 2005 John Doe &amp; Family</copyright>
    <itunes:subtitle>A show about everything</itunes:subtitle>
    <itunes:author>John Doe</itunes:author>
    <itunes:summary>All About Everything is a show about everything.</itunes:summary>
    <description>All About Everything is a show about everything.</description>
    <itunes:owner>
      <itunes:name>John Doe</itunes:name>
      <itunes:email>john.doe@example.com</itunes:email>
    </itunes:owner>
    <itunes:image href="http://example.com/podcasts/everything/All.jpg" />
    <itunes:category text="Technology">
      <itunes:category text="Gadgets"/>
    </itunes:category>
    <itunes:category text="TV &amp; Film"/>
    <item>
      <title>Shake Shake Shake Your Spices</title>
      <itunes:author>John Doe</itunes:author>
      <itunes:subtitle>A short primer on table spices</itunes:subtitle>
      <itunes:summary>This week we talk about salt and pepper shakers, and overall aesthetics. Come and join the party!</itunes:summary>
      <enclosure url="http://example.com/podcasts/everything/AllEpisode3.m4a" length="8727310" type="audio/x-m4a" />
      <guid>http://example.com/podcasts/archive/aae20050615.m4a</guid>
      <pubDate>Wed, 15 Jun 2005 19:00:00 GMT</pubDate>
      <itunes:duration>7:04</itunes:duration>
      <itunes:keywords>salt, pepper, shaker, exciting</itunes:keywords>
    </item>
</channel>
</rss>
```

아이튠즈와 팟캐스팅 ④

- 팟캐스팅의 특징
  - 언제 어디서나 소비자가 원하는 디지털 콘텐츠를 감상
    - 스트리밍 대신 다운로드, 휴대용 디지털기기를 이용
  - 저비용으로도 제작이 가능하여 다양한 팟캐스트 제공
    - 기업뿐 아니라 개인 및 소규모 그룹 등 누구나 제작하고 배포
  - 보드캐스팅(Vodcasting) = Video + Podcasting
- 팟캐스팅의 활용분야
  - 음악 감상 이외에도 광고나 홍보, 교육 및 뉴스 제공 분야
  - 마케팅 수단으로 적극 활용
    - 맞춤형 광고 제작 등 개인화 서비스가 가능
  - 교육 목적으로도 많이 활용
    - 열린 강좌(Open Course)의 팟캐스트를 학생들에게 제공한 사례

문서에 의미를 줄 수 있는 소셜 태깅 ①

(1) 태그와 소셜 태깅

- 태그(Tag)
  - 웹문서/자료에 연관된 정보나 키워드 정보를 '꼬리표'처럼 부착
    - 정보의 분류, 검색 등 부가적인 기능이 가능하게 해주는 메타데이터
    - 문서 내에서 단어, 문장, 이미지 등 어떤 구성요소에도 부착 가능
    - 컴퓨터와 사람이 같이 이해하고 협력할 수 있는 시맨틱웹의 발전과정
  - 태깅(Tagging): 태그를 다는 행위
  - 소셜 태깅 (Social Tagging): 일반 사용자들도 태그를 달고 공유
    - 예

문서에 의미를 줄 수 있는 소셜 태깅 ②

- 플리커(www.flickr.com)
  - 소셜태깅으로 가장 유명한 이미지 공유사이트
  - 태그를 통한 이미지 분류와 검색을 제공한다
  - 이미지에 대한 간략한 설명과 함께 태그를 직접 입력
  - 위치정보의 태그도 인터페이스를 통해 편리하게 부착 및 검색

(a) 태그 작성 화면

(b) 위치정보 태그의 활용

폭소노미 ①

- 소셜 태깅
  - 여러 사용자가 집단지성을 통해 미리 정해지지 않은 분류체계인 폭소노미(Folksonomy)나 태그 구름(Tag Cloud)의 실현이 가능
- 폭소노미
  - 대중/집단지성에 의한 분류 시스템
    - 대중을 의미하는 'folk'와 분류법을 의미하는 'taxonomy'의 합성
  - 택소노미(Taxonomy)
    - 계층구조의 카테고리 분류방식
    - 하나의 리소스에 하나의 카테고리만 적용
    - 적당한 카테고리가 존재하지 않을 때에는 정확한 분류 불가능
  - 폭소노미
    - 기존에 정확한 분류가 없거나 새로운 분야에 속하는 자료도 분류
    - 하나의 자료를 다수의 카테고리로 분류 가능

폭소노미 ②

- 폭소노미의 장점
  - 집단지성에 의한 분류로 원하는 자료를 찾기 쉽다: 경험을 바탕
  - 한 번 본 자료를 다시 찾기 쉽다: 유사한 키워드로 검색 가능
  - 많은 양의 자료를 관리하는 데 효율적이다: 계속 변화하는 자료
- 태그 구름(Tag Cloud)
  - 태그들의 관계를 시각적으로 표현
    - 태그를 2차원 공간 상에 연관도나 중요도에 따라 배치하고 글자의 크기나 색상 등에 변화

마이크로포맷 ①

(1) 마이크로포맷의 목적

- 시맨틱웹에 대한 연구
  - 문서를 구조화하고 특정한 의미를 부여하거나 추가적인 정보를 교환하기에는 HTML의 기능이 부족하다는 지적
- 마이크로포맷(Microformat)
  - HTML 문서 내에서 간단한 의미를 표현하여 포함시키는 방법
  - 주소록, 사회적 관계, 일정공유, 위치정보의 표현 등 다양한 용도
- 테크노라티(Technorati)사에서 처음 제안
  - class 속성으로 정보간 관계나 추가적인 의미를 표현하여 전달하면, 이를 약속한 대로 해석하여 활용
  - rel 속성이나 rev 속성을 이용하면 더 다양한 의미 표현이 가능
- 장점: 간단하게 의미를 표현하여 전달할 수 있다
  - 기존의 사용하던 다른 포맷을 형태로 변경하여 그대로 활용 가능

마이크로포맷 ②

(2) 마이크로포맷 활용사례: hCard, geo

- 마이크로포맷 사용않는 경우

```
<div>
  <div>임순범</div>
  <div>숙명여자대학교</div>
  <div>02-710-9424</div>
  <a href="http://www.sm.ac.kr/">http://www.sm.ac.kr/</a>
</div>
```

  - 요소만으로는 이름, 소속기관, 연락처 등을 구분 불가능

- hCard 마이크로포맷 활용

```
<div class="vcard">
  <div class="fn">임순범</div>
  <div class="org">숙명여자대학교</div>
  <div class="tel">02-710-9424</div>
  <a class="url" href="http://www.sm.ac.kr/">http://www.sm.ac.kr/</a>
</div>
```

  - 요소의 class 속성에 이름은 'fn', 소속은 'org', 전화번호는 'tel'

마이크로포맷 ③

- geo 마이크로포맷 활용 예
  - 위도와 경도 정보 표현

```
강남구청역의 위치는 <span class="geo">
    북위 <span class="latitude">37.517</span>,
    동경 <span class="longitude">127.041</span>
  </span>입니다.
```

소셜미디어에서의 활용: XFN ①

- XFN(XHTML Friends Network)
  - 하이퍼링크를 이용해 인간관계를 표현하자는 것이 목적
  - 사람 간의 개인적 관계를 하이퍼링크의 rel 속성으로 표현

표 10.1
XFN 관계 프로파일의 일부(출처: 위키피디아)

관계 유형	관계
교우 관계(하나만 선택)	contact : 접촉할 수단을 아는 사람 acquaintance : 서로 인사나 짧은 대화가 있어왔던 사람 friend : 친구. 알고 있는 동료나 동향인
물리적 관계	met : 실제로 만난 적이 있는 사람
지리적 관계(하나만 선택)	co-resident : 공통 거주자. 같은 (길)거리에 있는 사람 neighbor : 근방에 사는 이웃
직업상 관계	co-worker : 동업자 혹은 직장 동료 colleague : 같은 학문/활동 분야에 몸 담고 있는 사람
가족(하나를 선택)	child : 친자 혹은 양자, 또는 보호자 관계인 사람 parent : child의 역관계. 부모 sibling : 공통된 부모를 가진 사람. 형제, 자매, 남매 spouse : 결혼한 사람. 배우자 kin : 상대적으로 확장된 가족의 일원으로 간주되는 사람. 친척

소셜미디어에서의 활용: XFN ②

- 블로그 서비스에서 인간관계를 표현
  - XFN 관계 형식은 순서에 상관이 없으며, 블로그 모임이나 링크 페이지에 인간관계를 덧붙임
- 인간관계는 간단한 저작 인터페이스를 통하여 손쉽게 작성

```
<a href="http://www.jerryis.com/tt" rel="friend met colleague">제리</a>
<a href="http://www.pixie.co.kr" rel="acquaintance met">픽사</a>
<a href="http://www.jennifer.pe.kr" rel="child">임주은</a>
```

  - 편집기 화면

소셜미디어에서의 활용: XFN ③

- 마이크로포맷의 활용
  - HTML 문서 내에서 특정부분의 의미를 추가하여 정보를 교환
    - 이메일로 명함 교환 hCard, 일정 공유 hCalendar, 이력서 정보 교환 hResume, 리뷰정보 표현 hReview, 위치정보 표현 geo,
    - 소셜 미디어 환경에서 관계 정보 표현 XFN, XOXO, xFolk 등
  - 트위터에서 대화 표현 사례

```
<body class=" user-style-shseo1215 logged-in ">
  <div id="doc">
  <div id="top-stuff">
  <div id="banners" style="clear:both"><noscript>
  <div class="banner-outer">
  <div class="banner">
...
<div class="tweet-row">
<div class="tweet-text js-tweet-text">전곡항의 sunset (@ 전곡 마리나항) [pic]:
  <a data-expanded-url="http://4sq.com/p15GqY"
      class="twitter-timeline-link" href="http://t.co/w3tqG0j2" rel="nofollow"
      class="twitter-timeline-link">4sq.com/p15GqY</a>
</div>
</div>
```

RIA의 등장 배경과 발전 방향 ①

- RIA(Rich Internet Application) 기술
  - 웹브라우저 기반 인터페이스의 단점을 개선
    - 웹브라우저 애플리케이션은 인터넷 트래픽으로 응답속도가 늦고 HTML 언어의 제약으로 자유로운 인터페이스 구현 곤란
  - 2000.10 미국 포레스트 리서치에서 'X-인터넷'이란 개념 발표
  - 2002년 매크로미디어 플래시 제품백서에 RIA 용어 처음 등장
  - 이후 어도비 플래시 기반의 플렉스(Flex), 마이크로소프트의 실버라이트(Silverlight), 선마이크로시스템즈의 자바 FX 등

그림 10.15

RIA 기술의 발전 과정
(자료참조: KRG)

RIA의 등장 배경과 발전 방향 ②

- Ajax 기술의 등장
  - RIA 기술의 단점
    - 표준 기술이 아니며 자신들의 고유한 플랫폼을 기반으로 실행
  - HTML5에서 풍부한 사용자 인터페이스 기능을 제공
    - HTML 언어의 한계점에서 시작하였지만 이전에 웹에서 불가능했던 많은 작업들이 HTML5에서 가능
  - Ajax와 HTML5는 웹표준을 근거

Ajax의 개념 및 특징 ①

- Ajax(Asynchronous Javascript And XML)
  - 웹 애플리케이션에서 자바스크립트와 XML을 이용하고 비동기 통신방식에 기반하여 사용자 인터페이스를 구현하는 기술
  - 2005.2 제임스 가렛(Jesse James Garrett)발표한 에세이에서 유래
    - "Ajax: A New Approach to Web Application", Adaptive Path
  - Ajax는 7가지 기술들의 조합

    'Asynchronous+JavaScript+CSS+DOM+XML+XSLT+XMLHttpRequest'
    - 표준언어 XHTML과 CSS를 이용한 표현 기술
    - DOM을 이용한 동적인 화면 구성과 상호작용
    - XML과 XSLT를 이용한 데이터 교환 및 데이터 처리
    - XHR(XMLHttpRequest)를 이용한 비동기 데이터 통신
    - 이해하기 쉬운 자바스크립트 언어를 이용하여 모든 것을 통합 제어

Ajax의 개념 및 특징 ②

- Ajax 애플리케이션 모델
  - 웹서버 측 데이터 처리 로직과 클라이언트 측의 사용자 인터페이스 로직을 분리
    - 처리 결과에서 필요한 데이터만 XML 데이터로 전송
    - 클라이언트측에 HTML, CSS, 자바스크립트를 처리하는 Ajax 엔진필요

(a) 기존 웹 애플리케이션 모델

(b) Ajax 웹 애플리케이션 모델

Ajax의 개념 및 특징 ③

- ‘Asynchronous’ 개념의 비동기 통신
  - 이전의 웹 애플리케이션
    - 서버에서 처리 결과를 브라우저에 돌려보낼 때까지 대기
    - 예, 기존의 검색 엔진, 상거래 사이트에서 예금 거래
  - Ajax 애플리케이션 모델
    - 서버에서 데이터 처리하는 동안 브라우저는 사용자의 다음 작업 처리
    - 대표적인 사례: 구글지도

(a) 기존의 웹 애플리케이션 모델 (동기식)

(b) Ajax 웹 애플리케이션 모델 (비동기식)

Ajax의 구현 사례 ①

(1) 검색 제시어 기능 (검색어 추천 기능)

- 낱말을 다 입력하기 전에 추천단어 목록을 입력창 아래쪽에 표시
  - ‘Aj’입력 => ‘Aj’로 시작하는 단어, ‘a’입력 => ‘Aja’로 시작하는 단어
- 단어 목록만 XML 형태 텍스트 데이터로 전송: 전송량 매우 감소
- 제시어 창만 갱신함으로써 인터페이스의 반응속도를 증진

(a) ‘Aj’ 입력

(b) ‘Aja’ 입력

Ajax의 구현 사례 ①

(2) 지도 이미지의 비동기 처리

- 구글 지도
  - 지도의 이미지를 전부 다운로드 받기 전에 다음 작업을 실행
  - 이미지가 모두 도착하지 않았을 경우
    - 이동시: 확대하려는 부분의 지도 이미지가 없으면 회색으로
    - 확대시: 새로 받기 전의 이미지를 이용하여 단순히 확대
  - 지도 이미지를 다 받으면 정상적으로 표시

(a) 확대하기 전의 지도

(b) 일단 회색 처리

(c) 일단 이미지 확대 처리

오픈 API와 매쉬업 서비스의 개념 ①

- 오픈 API: ‘Open Application Programming Interface’
  - 기존의 API 개념을 웹사이트에 확장한 것
    - 웹사이트의 일부 기능을 남들이 사용하도록 API 형태로 공개한 것
  - 공개한 API를 이용하여 손쉽게 원하는 웹서비스 기능을 구현
    - 검색 기능이나 지도 기능 등을 자신의 사이트에서 손쉽게 이용
  - 이점
    - 서비스 활용도가 높아지면서 서비스가 더욱 활성화되어 트래픽 증가
    - 유지보수가 쉬워질 뿐 아니라 기업간 제휴나 서비스 공유가 용이
- 매쉬업(Mashup)
  - 인터넷에서 서로 다른 서비스를 조합하여 새로운 웹서비스 생성
  - 공개된 API를 이용하므로 개발비용 매우 적고 가볍게 프로그래밍
  - 개방과 공유를 기반으로하는 웹2.0 환경의 핵심 기술

오픈 API와 매쉬업 서비스의 개념 ②

- 최초의 매쉬업 서비스
  - 폴 레이드매처(Paul Rademacher)의 하우징맵(HousingMaps)
    - 구글의 지도 + 크레이그리스트(Craigslist) 부동산 정보를 연동
  - 이를 계기로 구글을 비롯한 여러 웹사이트에서 API를 공개

(a) Craigslist의 부동산 정보

(b) 구글 지도

(c) 하우징맵

매쉬업 서비스의 종류

- 매우 다양한 오픈 API가 제공
  - 구글 지도뿐 아니라 구글 검색, 유튜브의 비디오, 플리커의 이미지 제공, 아마존의 상품검색 및 구매 기능 등
- API를 이용하여 다양한 매쉬업 서비스 구현
  - 지도, 검색, 이미지나 비디오 제공, 쇼핑 등의 분야

(a) 많이 이용된 상위 10개 오픈 API (b) 많이 구현된 상위 10개 매쉬업 유형

그림 10.21

많이 이용된 오픈 API와 매쉬업 유형(출처: ProgrammableWeb.com)

구글 API

- 다양한 오픈 API를 제공
  - 구글 지도 API, 구글 검색 API, 구글 캘린더 API, 유튜브 API
  - 구글 애드센스 API, 구글 차트 API, 구글 주소록 API, 웹로그 분석
- 사례: 여행안내 사이트 ‘iGuide.travel’
  - 구글 애드센스 API, 구글 지도 API, ‘bookingwiz.com’ 검색 API

소셜 미디어 API

- 소셜미디어 사이트에서도 다양한 매쉬업 서비스
  - 페이스북의 경우 2007년 오픈 API 공개 후, 6개월 만에 수만개의 애플리케이션이 개발됨
  - 데이터를 활용한 다양한 애플리케이션 서비스

(a) 페이스북에서 친구관계 (소셜그래프)

(b) 페이스북 교신 횟수

(c) 링크드인에서 친구관계

국내에서 제공하는 API

- 대형 포털사이트를 중심으로 오픈 API를 제공
  - 네이버: 지도, 검색, 카페, 미투데이 등 기능에 대한 오픈 API 제공
  - 다음: 지도 API, 콘텐츠 데이터 제공 API, 인증이 필요한 API 등
  - 매년 공동으로 매쉬업 경진대회 개최

(a) 네이버 오픈 API 홈페이지

(b) 다음 오픈 API 홈페이지

매쉬업 사이트 구축 사례 ①

- 간단한 예제: 네이버 지도와 컨트롤을 내 사이트에 포함
  - API 키 발급: 네이버의 경우 API 이용등록 페이지에서 지도 API 키 발급 (http://dev.naver.com/openapi/register)
  - 지도를 화면에 출력하고, 줌 컨트롤을 표시하기 위한 코드

(a) 기본 지도에 줌 컨트롤 추가

(b) 범례, 마커, 교통상황 등 각종 정보 추가

매쉬업 사이트 구축 사례 ②

① 네이버에서 제공하는 스크립트와 지도 API키값을 헤드에 포함

② 지도를 출력할 컨테이너 설정

③ 위치객체 생성

④ 지도객체 생성

⑤ 줌 컨트롤 객체 추가

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>OpenAPI Map Test - Zoom Control </title>
<script type="text/javascript" src="http://openapi.map.naver.com/openapi/naverMap.naver?ver=2.0
    &key=f32441ebcd3cc9de474f8081df1e54e3"> // ① 오픈 API 호출 - API 키 </script>
</head>
<body>
  <div id="testMap" style="border:1px solid #000; width:700px; height:400px; margin:20px;
    display:block;"></div> <!-- ② 지도를 출력할 컨테이너를 설정 -->
  <script type="text/javascript">
    var oPoint = new nhn.api.map.LatLng(37.5010226, 127.0396037);  // ③ 위치 생성
    nhn.api.map.setDefaultPoint('LatLng');
    oMap = new nhn.api.map.Map('testMap',  // ④ "testMap" 태그에 지도 객체 생성 및 속성 인자 전달
      point : oPoint,                      // 지도 중심점의 좌표
      zoom : 10,                           // 초기 줌 레벨은 10으로 둔다.
      enableWheelZoom : true,              // 마우스 휠 동작으로 지도를 확대/축소할지 여부
      enableDragPan : true,                // 마우스로 끌어서 지도를 이동할지 여부
      enableDblClickZoom : false,          // 더블클릭으로 지도를 확대할지 여부
      mapMode : 0,                         // 지도 모드(0: 일반지도, 1: 겹침지도, 2: 위성지도)
      minMaxLevel : [ 1, 14 ],             // 지도의 최소/최대 축척 레벨
      size : new nhn.api.map.Size(500, 400) // 지도의 크기
    );

    var mapZoom = new nhn.api.map.ZoomControl(); // ⑤ 줌 컨트롤 선언
    mapZoom.setPosition(left:80, bottom:40);                // 줌 컨트롤 위치 지정
    oMap.addControl(mapZoom);                    // 줌 컨트롤 추가.
  </script>
</body>
</html>
```