







**NAVER** OpenAPI  
Beginner's Guide Book

## 문서 개요

이 책은 네이버 OpenAPI를 내용과 종류를 간략하게 소개하고, 네이버 OpenAPI를 이용하여 애플리케이션을 개발하는 방법을 설명해주는 초보자용 가이드 북입니다. 독자는 이 책에 소개된 개발 사례와 예제 코드를 이해, 실습함으로써 네이버 OpenAPI를 충분히 습득할 수 있습니다.

## 1장. OpenAPI 소개

네이버 OpenAPI의 정의, 종류, 이용 방법 등을 간략하게 소개해줌으로써 독자의 흥미를 유발시킵니다. 그리고 네이버 OpenAPI를 적용한 사례를 그림과 함께 소개합니다.

## 2장. OpenAPI로 개발하기

네이버 OpenAPI를 이용하여 개발한 애플리케이션 예제를 소개합니다. 각 예제에서는 예제 코드를 제공하며, 예제 코드의 프로그래밍 절차를 상세히 설명하고 있습니다. 예제 코드가 제대로 실행되지 않을 경우에 대비하여 Troubleshooting 방법에 대한 설명도 제공합니다.

## 부록. RSS 이해하기

RSS 정의, RSS 문서의 구조, RSS 리더에 대하여 설명합니다.

## 독자

이 책은 네이버 OpenAPI 초보 사용자를 독자로 합니다. 독자는 기본적으로 HTML, 자바스크립트, PHP, (Python) 코드를 작성 및 이해할 수 있어야 합니다.

## 표기 규칙

### 참고 표기

참고 ○ 독자가 참고해야 할 내용을 기술했습니다.

### 절차 표기

이 문서에서 순차적인 절차를 표현할 때는 다음과 같은 Step 서식을 사용합니다.

### ▽ STEP 1\_ 네이버 로그인하기

### 네이버 OpenAPI 클래스/메소드 표기

이 문서에서 네이버가 제공하는 OpenAPI 클래스명과 메소드명은 기울림꼴로 표기하였습니다. 즉, `NMap.addControl()` 메소드는 `NMap.addControl()` 와 같이 표기합니다.

### 소스 코드 설명 시 표기

문장에서 소스 코드의 일부를 가리킬 때 '`preg_replace_callback()`'와 같이 `Bitstream Vera Sans Mono` 폰트로 표기하였습니다.

### 소스 코드 표기

이 문서에서 소스 코드는 회색 바탕에 검정색 글씨로 표기합니다. 코드에 대한 설명은 코드 바로 아래에 절차 번호 서식으로 기술합니다. 코드 안에 파란색 점선과 숫자는 코드 설명의 절차 번호와 코드의 위치를 매핑시키기 위해 사용합니다.

```
define('NAVERKEY', 'YOURNAVERKEY');  
function makeItem($title, $mapx, $mapy)  
{  
    줄략  
}
```

① NAVERKEY를 정의합니다.

② makeItem()함수를 작성합니다.

## CONTENTS

---

<b>1. OpenAPI 개요</b>	<b>네이버 OpenAPI</b>	08
	<b>네이버 OpenAPI의 종류</b>	09
	<b>네이버 OpenAPI 이용안내</b>	11
	<b>적용 사례</b>	12
	NEXON 검색 (HTTP://SEARCH.NEXON.COM)	12
	천리안 검색 (HTTP://SEARCH.CHOL.COM/)	12
	ENCYBER 지도 (HTTP://ENCYBER.COM/MAP/)	13
	오픈베이 (HTTP://WWW.OPENBAY.CO.KR/MAP.ASPX)	13
 <b>2. OpenAPI로 개발하기</b>	 <b>시작하기</b>	16
	네이버 API 이용 등록	16
	 <b>백과사전 검색 예제</b>	18
	문제 상황	18
	준비하기	18
	구현하기	18
	예제 코드 실행 결과	22
	전체 소스코드	22
	 <b>최적 지점 찾기 예제</b>	24
	문제 상황	24
	준비하기	24
	구현하기	24
	예제 코드 실행 결과	30
	전체 소스코드	31
	 <b>약도 만들기 예제</b>	34
	문제 상황	34
	준비하기	34
	구현하기	34
	예제 코드 실행 결과	44
	전체 소스코드	45
	 <b>지도/블로그 검색 예제</b>	50
	문제 상황	50
	준비하기	50
	구현하기	50
	예제 코드 실행 결과	58
	전체 소스코드	61
	 <b>Troubleshooting</b>	68

---

<b>부록</b>	<b>RSS 이해하기</b>	<b>70</b>
	RSS란?	70
	RSS 리더 활용하기	71
	RSS 문서의 구조	71

---

<b>찾아보기</b>	<b>그림 목록</b>	
	그림 1.1 NEXON 검색 화면	14
	그림 1.2 천리안 검색 화면	14
	그림 1.3 ENCYBER 지도 화면	15
	그림 1.4 오픈베이 판매지 위치 화면	15
	그림 2.1 네이버 API 이용등록 페이지	18
	그림 2.2 포트 설정	19
	그림 2.3 백과사전 검색예제 실행 결과 화면	24
	그림 2.4 최적 지점 찾기 예제 실행결과 화면	32
	그림 2.5 약도 만들기 예제 실행 결과 화면	46
	그림 2.6 지도 출력 화면 (1)	54
	그림 2.7 지도 출력 화면 (2)	58
	그림 2.8 지도/블로그 검색 예제 실행 결과 화면	61
	그림 2.9 RSS 링크	72
	그림 2.10 RSS 리더	73

# 01



# OpenAPI 개요

>



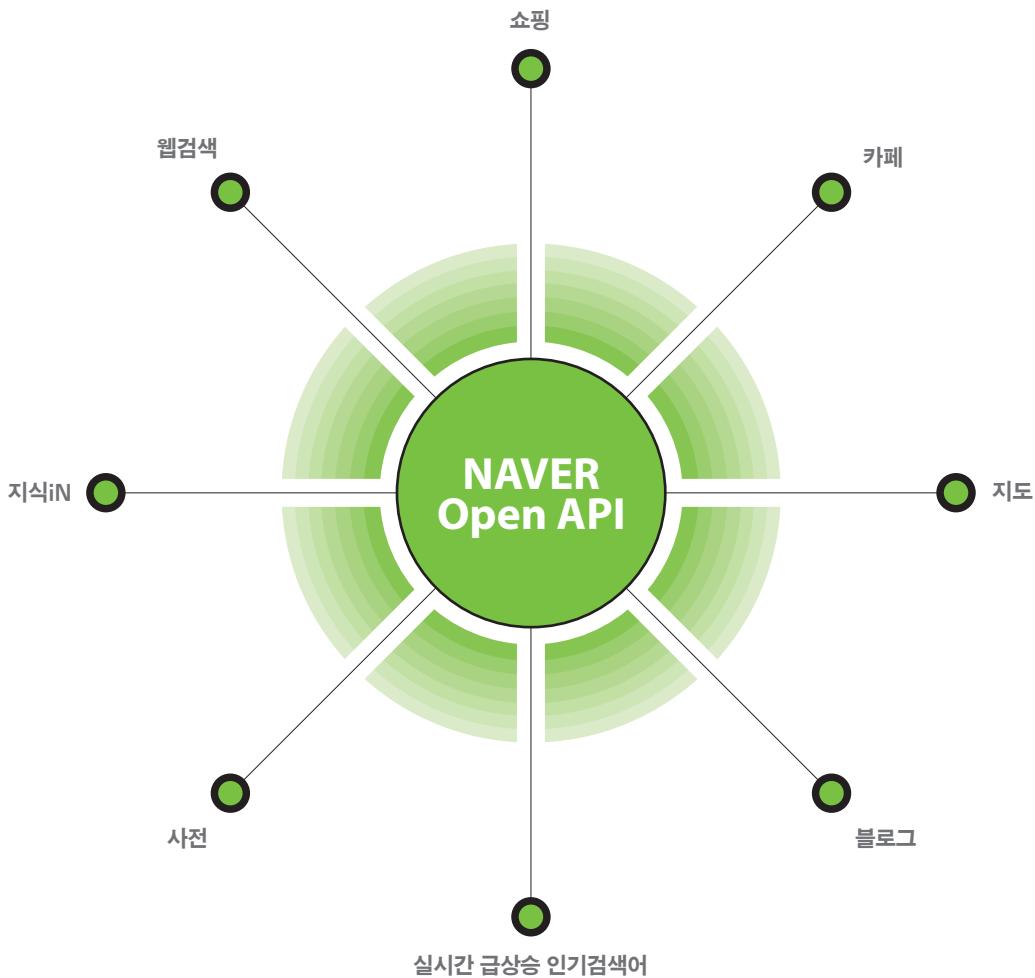
- 네이버 OpenAPI 란
- 네이버 OpenAPI의 종류
- 네이버 OpenAPI 이용안내
- 적용 사례

## 네이버 OpenAPI란

네이버 OpenAPI는 네이버의 다양한 서비스와 네이버에 쌓인 지식 정보를 사용자의 웹 페이지에서 활용할 수 있도록 공개한 개발자 인터페이스입니다.

여러분은 네이버 OpenAPI를 이용하여 네이버의 기술과 서비스를 활용할 수 있습니다. 네이버 OpenAPI는 네이버의 다양한 서비스를 웹 페이지에서 활용할 수 있도록 지원 할 뿐만 아니라, 더 나아가 여러 가지 서비스를 혼합한 창의적인 애플리케이션을 스스로 제작할 수 있도록 지원합니다. 예를 들어, 지도API를 이용하면 약도를 웹 페이지에 출력할 수 있을 뿐만 아니라, 검색API와 결합해서 블로그 검색 결과를 지도 위에 표시하도록 서비스를 혼합할 수 있습니다. 이러한 것을 서비스간 융합(mashup)이라고 하며, 이러한 융합을 통해 새로운 가치를 창출해 낼 수 있습니다. 더불어 네이버는 OpenAPI 활용 방법을 상세하게 안내하기 때문에, 여러분은 적은 투자와 작은 노력으로도 여느 포털보다 나은 인터넷 서비스를 구축할 수 있습니다.

기존에는 없던 서비스를 창출해 내는 일, 남들은 생각해 본 적이 없는 창의적인 애플리케이션을 제작하는 일, 서비스에 대한 고정관념을 뛰어 넘는 일, 내가 원하던 바로 그 맞춤형 서비스를 구현하는 일은 이제는 더 이상 어려운 일이 아닙니다. 네이버 OpenAPI가 여러분을 돋고 있습니다.



# 네이버 OpenAPI의 종류

## 검색관련기능 API

검색관련기능 API는 검색 순위 정보나 성인 검색어 판별 서비스와 같이 검색 결과를 해석 및 가공한 데이터를 제공합니다.

### 01 / 추천 검색어

네이버 검색어의 패턴을 분석해 이용자들이 입력한 검색어 중 연관도가 높은 검색어를 추천해드립니다.

### 02 / 실시간 급상승 검색어

이용자들이 지금 이 순간 가장 많이 입력하신 검색어를 보여드립니다. 최신 동향을 파악하기를 원하신다면 실시간 급상승 검색어를 꼭 활용해보세요.

### 03 / 오타변환

한/영 키를 잘못 설정하고 검색하셨을 경우, 입력하신 검색어를 자동으로 변환/추천해주는 기능입니다.

### 04 / 바로 가기

네이버에서 선정한 필수 웹사이트의 바로가기를 여러분의 홈페이지에서도 응용하실 수 있습니다.

### 05 / 성인 검색어 판별

검색어의 성인관련 여부를 판단해드립니다.

깨끗하고 안전한 웹 검색결과를 만들어보세요.

## 서비스API

서비스 API에는 지식 스폰서 API, 지도 API, 데스크톱 위젯 API가 포함됩니다.

### 06 / 데스크탑 API

데스크탑 위젯 API를 이용하여 개발자나 프로그래머가 직접 위젯을 제작, 네이버에 등록하여 배포할 수 있습니다.

### 07 / 지식 스폰서

각 분야의 전문가분들이 지식iN에 올라오는 질문에 대한 답변 활동을 하는데, 네이버에서 그 답변 활동의 편의성을 돋기 위해 API를 제공해드리고 있습니다.



## OpenAPI의 개요

네이버 OpenAPI

네이버 OpenAPI의 종류

네이버 OpenAPI 이용안내

적용 사례

## 08 / 지도 API

웹 사이트에 지도를 표시할 수 있으며, 지도상의 원하는 위치에 정보창을 통해 관련 정보를 표시할 수 있습니다

### 검색결과 API

검색결과 API는 네이버 검색 서비스를 모듈화하여 검색 서비스별 API로 제공합니다.

## 09 / 블로그 검색

블로그 검색결과를 이용하시면 여러분의 웹 사이트에서 쉽게 네이버의 블로그 검색결과를 이용하실 수 있습니다.

## 10 / 지식iN 검색

다양하고 풍부한 지식iN 검색결과를 여러분의 웹사이트에 이용하실 수 있습니다.

## 11 / 카페 / 카페글 검색

사용자가 입력한 키워드에 딱 맞는 카페와 카페 안에 숨겨진 글들을 찾아 드립니다.

## 12 / 웹문서 검색

네이버의 검색엔진이 찾아낸 수많은 웹문서 검색결과를 API로 제공해드립니다.

## 13 / 이미지 검색

네이버의 풍부한 이미지 검색결과를 공유해드립니다.

## 14 / 쇼핑 검색

네이버 쇼핑에 등록된 상품의 데이터 검색

## 15 / 책 검색

제목뿐만 아니라 저자, 출판사, 카테고리별 검색 등 다양한 옵션이 제공되는 책 검색결과를 이용하실 수 있습니다.

The screenshot shows a search results page for 'Season 2' on the Naver Blog platform. The results include several posts from different blogs, each with a thumbnail image, title, and a brief excerpt. The interface includes navigation buttons for 'First', 'Previous', 'Next', and 'Last' pages, as well as a search bar at the top. The overall layout is clean and organized, typical of a search engine's results page.

**16 / 전문자료 검색**

학술 논문, 레포트, 전문 기관의 보고서 등 신뢰도 높은 자료만을 공유해드립니다.

The screenshot shows a search results page for '네이버' on the Naver Academic Search platform. It displays several academic papers and reports, each with a thumbnail, title, and a brief abstract. The interface includes navigation buttons for page 15 of 15 pages.

**17 / 내PC 검색**

내 컴퓨터에 있는 자료를 검색하는 기능을 사용하실 수 있습니다.

**18 / 사전 검색**

네이버의 국어, 영어, 일본어 사전을 검색

**19 / 지역 검색**

네이버 지역서비스에 등록된 각 지역별 업체 및 상호 검색결과를 응용하실 수 있습니다

# 네이버 OpenAPI 이용안내

네이버 Open API는 개인, 기업체, 단체 및 기관 등 누구나 무료로 활용할 수 있습니다. 단, 상업적 용도로 사용하는 것은 제한하고 있으며, 일일 사용량을 초과할 경우에는 OpenAPI 제휴 체결을 통해서 추가적인 사용을 할 수 있도록 기회를 제공하고 있습니다. 자세한 사항은 네이버 OpenAPI 사이트에서 FAQ와 이용약관을 확인하시기 바랍니다.

- 네이버 OpenAPI: <http://openapi.naver.com>

## OpenAPI의 개요

네이버 OpenAPI

네이버 OpenAPI의 종류

네이버 OpenAPI 이용안내

## 적용 사례

# 적용 사례

### Nexon 검색 (<http://search.nexon.com>)

(주)넥슨에서는 NEXON 검색 결과에 네이버 지식iN 검색 결과와 네이버 블로그의 검색 결과를 포함해서 제공하고 있습니다. 이는 네이버 검색결과API를 NEXON 검색 사이트에서 이용한 사례입니다.

The screenshot shows the Nexon search results page. At the top, there's a navigation bar with links for '네이버 OpenAPI', '지식iN', '채널FUN', '길드', '넥슨머니', and social sharing options. Below the navigation is a search bar and a main content area. The content area displays search results from different sources:

- 네이버 OpenAPI**: A section titled '제작자' (Creator) with a link to '기사(News)' and a snippet about 'API 모드(LAUNCHER API) FTP MODE'.
- 지식iN**: A section titled '검색결과' (Search Results) with a snippet about 'API 모드(LAUNCHER API) FTP MODE'.
- 채널FUN**: A section titled '제작자' (Creator) with a snippet about 'API 모드(LAUNCHER API) FTP MODE'.
- 길드**: A section titled '제작자' (Creator) with a snippet about 'API 모드(LAUNCHER API) FTP MODE'.
- 넥슨머니**: A section titled '제작자' (Creator) with a snippet about 'API 모드(LAUNCHER API) FTP MODE'.

On the right side, there are two sidebar sections: '연간 검색어 순위' (Annual Search Ranking) and '영문 본 기사' (Original English Article), each listing several items.

그림1.1 NEXON 검색 화면

### 천리안 검색 (<http://search.chol.com/>)

인터넷 포털 천리안에서는 검색 결과에 네이버 지식iN의 검색 결과를 포함해서 제공하고 있습니다. 이는 네이버 검색결과API를 천리안 사이트에서 활용한 사례입니다.

The screenshot shows the Chol search results page. At the top, there's a navigation bar with links for '통합검색', 'open api', '검색', '사이트', '뉴스', '지식iN', '게시판', '블로그/블로그', '감자', '미미자', '쇼핑', and '지도'. Below the navigation is a search bar and a main content area. The content area displays search results from different sources:

- 지식iN (69건)**: A section titled 'php에서 openApi 이용 xml 가져오기 해' with a snippet about 'php에서 openApi 이용 xml 가져오기 해'.
- 사이트 (1건)**: A section titled '천리안 검색' with a snippet about '천리안 검색'.

On the right side, there are two sidebar sections: '지식iN 더 보기' (More Knowledge) and '사이트 등록 안내' (Site Registration Guide).

그림1.2 천리안 검색 화면

### Encyber 지도 (<http://encyber.com/map/>)

Encyber에서는 네이버 지도 API를 이용해 지도 위에 백과사전 항목을 표시해주는 서비스를 제공하고 있습니다. 지도상의 마커를 클릭하면 그 지점에 해당하는 백과사전 내용이 정보창으로 출력됩니다. 정보창의 링크를 누르면 상세 정보 페이지로 이동합니다.

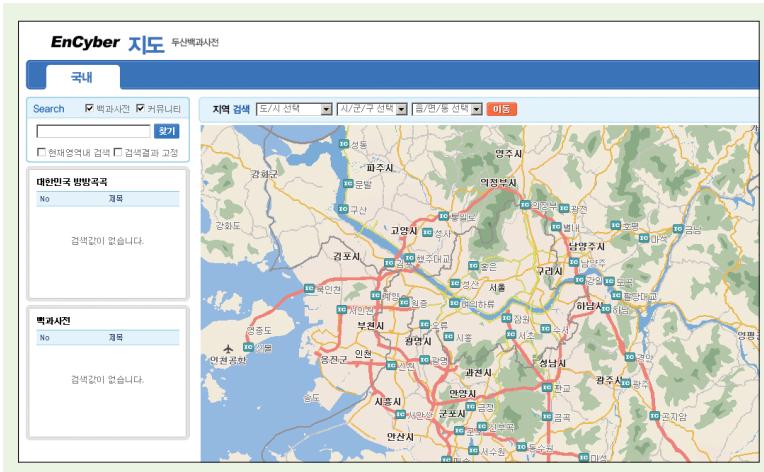


그림1.3 Encyber 지도 화면

### 오픈베이 (<http://www.openbay.co.kr/map.aspx>)

직거래 서비스를 제공하는 오픈베이에서는 네이버 지도를 이용해서 판매자를 찾을 수 있는 서비스를 제공합니다. 판매자의 위치를 지도 위에 마커로 표시해주며, 마커를 클릭하면 물품에 대한 정보를 정보창을 통해 출력합니다. 링크를 클릭하면 물품 판매 페이지로 이동할 수 있습니다.

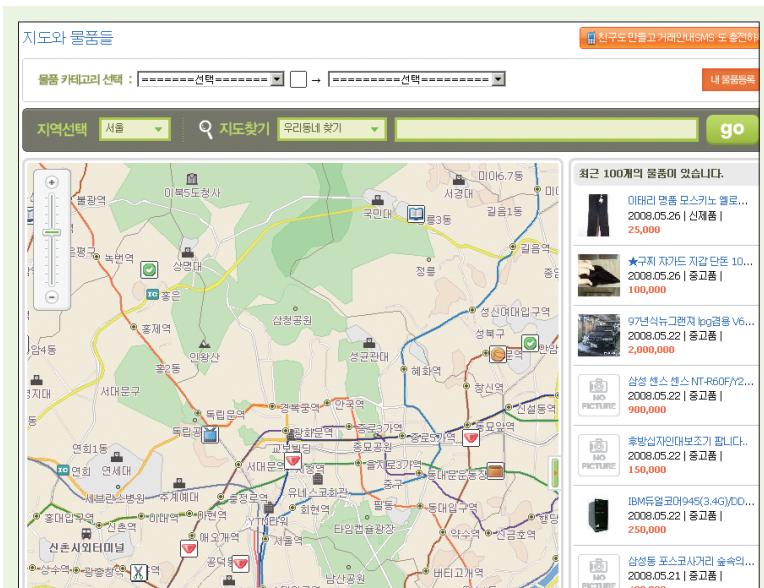


그림1.3 Encyber 지도 화면

# 02



# OpenAPI로 개발하기

>

시작하기

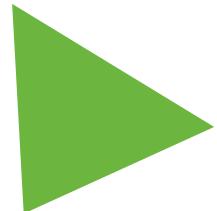
백과사전 검색 예제

최적 지점 찾기 예제

약도 만들기 예제

지도 / 블로그 검색 예제

Troubleshooting



# 시작하기

## 네이버 API 이용 등록

네이버에서 제공하는 OpenAPI를 이용하려면 먼저 네이버 API 키를 발급받아야 하며, 이를 위하여 네이버 API 이용 등록을 해야 합니다. 네이버 API 키에는 검색을 위한 검색API 키와 지도API를 사용하기 위한 지도 API 키가 있습니다. 지도API를 제외한 모든 API는 검색 API 키를 사용하여 이용할 수 있습니다. 다음은 네이버 API 키 발급 절차입니다.

### A. 검색 API 키 발급받기

검색 API 키는 네이버 API 이용등록 페이지(<http://openapi.naver.com>)에서 발급받을 수 있습니다. 네이버 API 이용등록 페이지에서 이메일, 연락처, 사용 예상 API와 사용 용도 등을 기입하고, 이용약관에 동의하면 바로 검색 API 키를 발급받을 수 있습니다. 검색 API 키는 네이버 ID당 한 개씩 발급됩니다.



그림 2.1 네이버 API 이용등록 페이지

### B. 지도 API 키 발급받기

지도 API 키도 검색 API 키와 마찬가지로 API 이용등록 페이지(<http://openapi.naver.com>)에서 발급받을 수 있습니다. 지도 API 키를 발급받으려면 지도 API를 이용하려는 서버의 도메인과 디렉터리 정보를 입력해야 합니다. 이렇게 발급받은 키는 해당 도메인과 디렉터리에서만 사용할 수 있습니다. 정상적인 절차로 지도 API 키를 발급받고도 주소가 맞지 않으면 지도가 출력되지 않습니다.

이 경우에는 지도 API 키 등록 시의 도메인/디렉터리 정보와 맞는지 확인해야 합니다. 지도 API 키는 한 개의 네이버 ID를 가지고 여러 개 발급받을 수 있습니다. 여러 도메인에서 지도 API를 사용하여 개발하는 경우에는 필요한 만큼 지도 API를 발급받아 사용하면 편리합니다.

예를 들어, <http://www.naver.com/map/> 디렉터리로 지도 API 키를 발급받은 경우, 지도 API가 동작하는 페이지와 정상적으로 동작하지 않는 페이지의 예는 다음과 같습니다.

### 1. 정상적으로 동작하는 페이지

A. <http://www.naver.com/map/> 하위 디렉터리의 모든 페이지

<http://www.naver.com/map/map.html>

<http://www.naver.com/map/seoul/kangnam/yeoksam/map.html>

### 2. 정상적으로 동작하지 않는 페이지

A. <http://www.naver.com/map/> 하위 디렉터리가 아닌 모든 페이지

<http://www.naver.com/map2/map2.html>

<http://www.naver.com/openapi/map.html>

**참고** ● 기본 포트(80)가 아닌 다른 포트를 사용할 경우에는 키를 발급받을 때 다음과 같이 포트 정보도 같이 입력하여 디렉터리를 지정해야 합니다. 포트가 다른 경우에는 지도 API가 정상적으로 동작하지 않습니다.

디렉토리 지정 (아래 설명문을 잘 읽고, 지도 키를 발급받으시기 바랍니다)

그림 2.2 포트 설정

---

## 백과사전 검색 예제

### 문제 상황

웹 페이지에 글을 쓰다가 부연 설명이나 용어 풀이가 들어갔으면 좋겠다는 생각을 할 때가 있습니다. 예를 들어 다음과 같은 문장을 적었다고 합시다.

“저는 주로 Python Language를 사용하고는 있지만, Haskell Language도 좋아합니다.”

위 문장을 적으면서 생각해보니 읽는 사람에 따라 Python이나 Haskell을 모를 수도 있다는 생각이 듭니다. 괄호를 치고 부연 설명을 달고 싶은데, 일일이 적기가 귀찮습니다. 부연 설명을 넣고 싶은 단어에 자동으로 설명이 들어가면 얼마나 좋을까 생각합니다. 이런 경우 부연 설명이 필요한 단어에 대괄호를 넣어 주면, 네이버 백과사전에 정의된 설명이 자동으로 문장 내에 삽입되도록 할 수 있습니다.

“저는 주로 [Python Language]를 사용하고는 있지만, [Haskell Language]도 좋아합니다.”

“저는 주로 Python Language(Python is a high-level programming language first released by Guido van Rossum in 1991)를 사용하고는 있지만, Haskell Language(Haskell is a standardized purely functional programming language with non-strict semantics, named after the logician Haskell)도 좋아합니다.”

이런 일을 하는 애플리케이션을 네이버OpenAPI를 이용해서 구현할 수 있습니다.

### 준비하기

백과사전 검색 예제를 따라해 보기 전에 준비해야 할 것이 있습니다.

#### A. 필수사항

- 네이버 OpenAPI 키가 필요합니다.
- PHP5 이상 지원하는 웹 서버가 필요합니다.

#### B. 권장사항

- 본 예제는 HTML과 PHP로 작성했습니다. 따라서 HTML과 PHP로 작성된 코드를 조금은 이해할 수 있어야 합니다.
- 예제에서는 정규식을 사용하고 있습니다. 정규식에 대해 알아두면 본 예제를 이해하는데 도움이 됩니다. 정규식은 <http://php.pgeon.com/manual/kr/ref.pcre.php> 페이지를 참조하시기 바랍니다.
- 백과사전 검색 API에 대한 사용자 문서를 읽어두면 예제를 이해하는데 도움이 됩니다. 백과사전 검색 API에 대한 사용자 문서는 네이버 OpenAPI 사이트(<http://openapi.naver.com/>)에서 찾아볼 수 있습니다.

### 구현하기

네이버 백과사전 검색 결과를 자동으로 문장에 삽입하는 애플리케이션을 만들려면 다음과 같은 기능을 구현해야 합니다.



## STEP 1\_ 걸모습 만들기

문장을 입력할 글상자와 버튼을 만듭니다.

### ● dic.php 파일

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
    <title>사전예제</title>
  </head>
  <body>
    <form name="dic" method="get" action=<?=$_SERVER['PHP_SELF']?>>>
<textarea name="query" cols="75" rows="20">
<?php
echo $query
?>
</textarea>
<br>
<input type="submit" value="annotate">
</form>
</body>
</html>
```

위와 같이 작성하면 가로 75자, 세로 20줄의 글상자와 **annotate** 버튼이 만들어집니다. 글상자에 문장을 입력하고 **annotate** 버튼을 누르면 dic.php(소스 파일)에 입력한 문장을 전송합니다.



## STEP 2\_ 대괄호로 둘러싼 단어 찾기

dic.php로 전송된 문장을 처리하는 기능을 구현합니다.

### ● dic.php 파일

```
<?php
$annotated = "";
$query = "";

if (isset($_GET["query"])) {
    $query = trim($_GET["query"]);
    if (strlen($query) > 0) {
        $annotated = preg_replace_callback("/\[(\[^\\]]*)\]/", "dict", $query);
        $annotated = str_replace ("\n", "<br>", $annotated);
    }
}
?>
```

- ① 글상자에서 넘겨 받은 문장은 `$_GET["query"]` 전역변수에 저장되어 있습니다. 실제로 문장이 저장되어 있는지 `isset()` 함수로 확인하고, 문장이 저장되어 있는 경우 문장을 얻어옵니다.

- ② 얄어온 문장에서 불필요한 공백을 trim() 함수를 사용하여 제거합니다.
- ③ 얄어온 문장의 크기가 0이 아닐 경우, preg\_replace\_callback() 함수를 사용해서 대괄호로 둘러싸인 단어를 처리합니다. 즉, preg\_replace\_callback() 함수는 \$query 문자열에서 대괄호로 둘러싸인 단어를 찾아낸 후 dict() 함수를 호출하여 단어에 부연 설명을 붙인 다음, 그 결과를 대괄호로 둘러싸인 단어와 치환합니다. \$query 문자열에서 대괄호로 둘러싸인 모든 단어에 대해 치환합니다.
- ④ 치환된 문자열을 \$annotated 변수에 저장합니다.



### STEP 3\_ 단어에 부연설명 붙여주기

네이버 백과사전에서 단어를 검색하고, 단어에 부연 설명을 붙여주는 dict() 함수를 작성 합니다.

#### ● dic.php 파일

```
<?php
define('NAVERKEY', 'YOURNAVERKEY'); ①

function dict($matches) {
    $title = $matches[1];
    $encodedquery = urlencode($title);
    $url = "http://Openapi!.naver.com/search?query=$encodedquery&target=encyc&key=".NAVERKEY; ④

    $item = simplexml_load_file($url)->channel->item[0];
    if ($item) {
        $description = preg_replace("/<[^>]*>/", "", $item->description); ⑤
        $description = split("\.", $description, 100); ⑥

        foreach ($description as $desc) {
            if (strlen($desc) > 2) {
                $result = trim($desc);
                break;
            }
        }
    }

    return "$title($result)";
} ⑦
else {
    return $title;
}
?>
```

- ① 네이버 OpenAPI 키를 정의합니다. define문의 두 번째 파라미터로 여러분이 발급받은 OpenAPI 키를 입력합니다.
- ② dict() 함수를 선언합니다.
- ③ dict() 함수의 입력 파라미터인 \$matches에는 검색해야 할 단어가 배열 형식으로 저장되어 있습니다. \$matches[0]에는 정규식과 매칭된 결과 전체가 들어갑니다. 즉, \$matches[0]에는 “[Python Language]”와 같이 대괄호까지 포함된 문자열이 들어갑니다. \$matches[1]부터는 정규식에서 소괄호로 둘러싸인 부분이 들어갑니다. 즉, “Python Language”와 같이 대괄호가 제거된 문자열만 저장됩니다. 따라서 \$title에 \$matches[1]를 저장합니다.
- ④ 3번 절차에서 추출한 단어를 검색하기 위해서 백과사전 검색 API를 호출합니다. 이때, \$url은 네이버 OpenAPI 요청

URL 형식이어야 합니다. query 파라미터에 \$title을 입력하고, target 파라미터에는 encyc을 입력합니다.

- ⑤ `simplexml_load_file()` 함수를 사용해서 XML 형식의 검색 결과를 파싱합니다. `simplexml_load_file()` 함수는 백과사전 검색 결과로 얻어온 XML 문서를 객체로 만들어 줍니다. XML 문서의 엘리먼트는 객체의 멤버가 됩니다. 검색 결과로 얻어진 항목들은 객체의 `channel->item` 멤버에 저장되어 있습니다. 우리는 첫 번째 항목만 필요하므로 `item[0]`을 읽어와서 \$item 변수에 저장합니다.

**참고** ● 네이버 OpenAPI의 검색 결과로 반환되는 XML 문서는 모두 RSS(Really Simple Syndication) 형식으로 되어 있습니다. RSS 문서에 대한 자세한 설명은 이 문서의 “부록 A. RSS 이해하기”를 참조하시기 바랍니다.

- ⑥ \$item 변수의 값이 0이 아닌 경우, 단어에 부연 설명을 붙입니다. 부연 설명은 `$item->description`에서 얻어올 수 있습니다. 우선 `preg_replace()` 함수를 사용해서 `$item->description`에 들어있는 html 태그를 제거합니다. 그런 다음, `split()` 함수를 사용해서 마침표를 기준으로 문자열을 자릅니다. 이렇게 해서 여러 문장으로 이루어진 배열을 얻을 수 있습니다.
- ⑦ 문장의 배열을 순회하면서 처음으로 2글자가 넘는 문장을 읽어옵니다. 문장이 아닌 것을 잘못 얻어오는 경우를 최소화 하기 위하여 문장의 길이를 2글자로 제한했습니다. 2글자를 넘는 문장을 한 개 얻으면, 문장의 앞뒤 공백을 제거한 후에 `foreach` 루프를 빠져 나갑니다.
- ⑧ 결과값인 `$title($result)`를 반환합니다. 만약 검색 결과가 없을 경우(즉, \$item 변수의 값이 0인 경우)에는 \$title만 반환합니다.



## STEP 4\_ 출력하기

최종 결과 문장을 출력하는 단계입니다.

```
<div id="annotated">
<?php
echo $annotated;
?>
</div>
```

Step2에서 언급했던 `preg_replace_callback()` 함수가 `$annotated`에 넣어준 결과를 화면에 출력합니다. 위와 같이 하면 `$annotated` 변수에 들어있는 내용이 페이지 하단에 출력됩니다.

## 예제 코드 실행 결과

예제 코드를 실행하고 글상자에 문장을 입력한 후 **annotate** 버튼을 누르면 다음과 같은 결과를 볼 수 있습니다. 결과에 삽입된 주석은 아래와 다를 수 있습니다.



그림 2.3 백과사전 검색 예제 실행 결과 화면

## 전체 소스코드

### ● dic.php 파일

```
<?php
define('NAVERKEY', 'YOURNAVERKEY');

// 백과사전 항목을 검색하고 주석을 만들어서 반환하는 함수
function dict($matches) {
    $title = $matches[1];
    $encodedquery = urlencode($title);
    $url =
        "http://openapi.naver.com/search?query=$encodedquery&display=1&target=encyc&key=".
        NAVERKEY;

    $item = simplexml_load_file($url)->channel->item[0];
    if ($item) {
        // 백과사전 항목의 표제와 내용에서 html 태그를 제거하고 문장 단위로 자름
        $description = preg_replace("/<[^>]*>/", "", $item->description);
        $description = split("\.", $description);

        // 2글자가 넘는 가장 먼저 나오는 문장을 골라서 공백을 제거하고 반환
        foreach ($description as $desc) {
            if (strlen($desc) > 2) {
                $result = trim($desc);
                break;
            }
        }
    }
}
```

```
        return "$title($result)";
    }
    else {
        return $title;
    }
}

$annotated = "";
$query = "";

// 사용자가 입력한 글에서 대괄호로 둘러싸인 문자를 찾아냄
// 찾아낸 단어들에 dict() 함수를 호출해서 주석을 추가
// 최종결과는 $annotated에 저장
if isset ($_GET["query"])
{
    $query = trim($_GET["query"]);
    if (strlen($query) > 0) {
        $annotated = preg_replace_callback("/\[(^\]]*)\]/", "dict", $query);
        $annotated = str_replace("\n", "<br>", $annotated);
    }
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
        <title>사전예제</title>
    </head>
    <body>
        <form name="dic" method="get" action="<?=$_SERVER['PHP_SELF']?>">
<textarea name="query" cols="75" rows="20">
<?php
echo $query
?>
</textarea>
        <br>
        <input type="submit" value="annotate">
    </form>
    <div>
        <?php
        echo $annotated;
        ?>
    </div>
    </body>
</html>
```

## 최적 지점 찾기 예제

### 문제 상황

저희 가족은 조만간 새 집으로 이사하려고 합니다. 그래서 어디로 이사하면 좋을지 물색하고 있는 중입니다. 가족 중 세 명은 직장에 다니고 한 명은 학교에 다니고 있습니다. 안타깝게도 각 회사와 학교는 여기저기 흩어져 있어서 이사할 집을 결정하기가 쉽지 않습니다. 가족 구성원 모두의 직장과 학교에 가까운 곳으로 이사 갈 수 있으면 좋겠지만 그건 좀 어려워 보입니다.

그래서 몇 개 후보지를 정해놓고 그 중에서 직장과 학교까지의 거리의 총합이 가장 짧은 곳으로 정하려고 합니다. 네이버 지도API를 이용하면 가장 가까운 이사 후보지를 찾아 주는 애플리케이션을 만들 수 있습니다.

### 준비하기

최적 지점 찾기 예제를 따라해 보기 전에 준비해야 할 것이 있습니다.

#### A. 필수사항

- 네이버 지도 API 키가 필요합니다.
- 자바스크립트 실행을 위해서 FireFox 1.5 이상 또는 Microsoft Internet Explorer 6 이상을 사용해야 합니다.

#### B. 권장사항

- 본 예제는 HTML과 자바스크립트로 작성되어 있습니다. 따라서 독자는 HTML과 자바스크립트로 작성된 코드를 이해할 수 있어야 합니다. 네이버 지도 API에 대한 사용자 문서를 읽어주시면 예제를 이해하는데 도움이 됩니다. 지도 API 사용자 문서는 네이버 OpenAPI 사이트(<http://openapi.naver.com/>)에서 찾아볼 수 있습니다.

### 구현하기

우선 지도 화면에 가족들의 직장과 학교를 마커로 표시합니다. 그런 다음 몇몇 이사 후보지를 선택합니다. 그리고 버튼을 누르면 가장 가까운 후보지를 찾아서 그 후보지와 각 직장/학교를 선으로 연결해 보여주는 애플리케이션을 만들어 봅시다.



### STEP 1\_ 걸모습 만들기

네이버 지도와 마커 선택창, 최적점을 찾아주는 버튼, 마커와 선을 지우는 버튼이 출력되도록 전체 화면을 구현합니다.

#### ● map.html 파일

```
<!DOCTYPE HTML SYSTEM>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta http-equiv="Cache-Control" content="No-Cache">
  <meta http-equiv="Pragma" content="No-Cache">
  <title>최적지점 찾기6 예제</title>
</head>
<body style="margin:0px 0px 0px 0px">
  <div id='mapContainer' style='width:640px; height:480px'></div>
```

```

<div>
  <br>
  <form name="markerForm" action="">
    <input type="radio" name="markerGroup" value="office">
    office<br>
    <input type="radio" name="markerGroup" value="home">
    home<br>
  </form>
  <input type="button" value="find" onclick="find();"> ③
  <input type="button" value="clear" onclick="clearMarkers();"> ④
</div>
<script type="text/javascript" src="http://maps.naver.com/js/naverMap.naver?key=YOURMAPKEY"></script> ⑤
<script type="text/javascript" src="map.js"></script> ⑥
</body>
</html>

```

- ① 지도를 출력할 컨테이너를 설정합니다. 위 예제에서는 `div` 엘리먼트를 사용하여 설정하였습니다.
- ② 이사 후보지를 나타낼 마커와 직장/학교를 나타내는 마커를 구분하기 위하여 마커의 종류를 선택하는 라디오 버튼 2개를 만듭니다. 위 예제에서는 후보지 마커 버튼의 이름을 '`home`'으로, 직장/학교 마커 버튼의 이름을 '`office`'라고 불렀습니다.
- ③ 가장 가까운 후보지를 찾아내는 `find` 버튼을 생성합니다. `find` 버튼을 클릭하면 `find()` 함수를 호출합니다.
- ④ 지도에 표시한 마커를 모두 지우는 `clear` 버튼을 생성합니다. `clear` 버튼을 클릭하면 `clearMarkers()` 함수를 호출합니다.
- ⑤ 네이버 지도 API 자바스크립트를 불러옵니다. 이때 `key` 파라미터로 여러분이 발급받은 지도 API 키를 입력해야 합니다. 위 예제에서 `YOURMAPKEY`라고 표현된 부분은 여러분이 발급받은 지도API 키를 입력합니다.
- ⑥ `map.js` 자바스크립트를 불러옵니다. `map.js`는 다음 단계에서 우리가 작성할 자바스크립트 파일입니다.



## STEP 2\_ 지도 생성 및 초기화하기

지도 객체를 생성하고 초기화하도록 구현합니다.

### ● map.js 파일

```

var map0bj = new NMap(document.getElementById('mapContainer'), 640, 480); ①
var zoom = new NZoomControl();
zoom.setAlign("right");
zoom.setValign("bottom");
map0bj.addControl(zoom);
map0bj.setBound(0, 999999, 999999, 0); ③
NEvent.addListener(map0bj, "click", clickMap); ④

```

- ① `NMap` 클래스로 지도 객체를 생성합니다. 지도 객체를 생성할 때는 지도를 출력할 컨테이너 정보를 첫 번째 파라미터로 전달해야 합니다. 두 번째, 세 번째 파라미터로 지도의 크기를 지정할 수 있습니다. 크기를 지정하지 않을 경우 자동으로 컨테이너의 너비와 높이를 설정합니다.

**참고** ● `NMap` 클래스와 같이 이름이 'N'으로 시작하는 클래스는 네이버 지도 API에 정의되어 있는 클래스입니다.

- ② *NMap.setBound()* 메소드를 사용하여 컨테이너에 표시할 지도 영역을 설정합니다. *NMap.setBound()* 메소드의 입력 파라미터로 *left*, *top*, *right*, *bottom* 값을 넣습니다. 컨테이너 안에 지도 전체를 그려야 하므로 *left*와 *bottom*에는 0을 넣고, *right*와 *top*에는 999999를 넣습니다.

**참고** ● 지도의 좌표는 남쪽으로 갈수록 작은 값을 가지므로 *top*이 아닌 *bottom*의 값이 0이라는 점에 주의합니다.

- ③ 사용자가 지도 화면을 축소 또는 확대할 수 있도록 줌 컨트롤 객체를 생성합니다. *NZoomControl* 클래스로 줌 컨트롤 객체를 생성한 후에, *NZoomControl.setAlignment()*와 *NZoomControl.setVertical()* 메소드를 사용하여 줌 컨트롤 객체의 표시 위치를 설정합니다. 그런 다음 *NMap.addControl()* 메소드로 줌 컨트롤 객체를 지도에 추가합니다. 이렇게 하면 축척 슬라이드 바가 지도 화면에 표시됩니다.
- ④ *NEvent.addListener()* 메소드를 사용하여 *click*이벤트를 등록하고, 이벤트가 발생할 때 실행할 코드를 등록합니다. 즉 지도 객체를 클릭했을 때 *clickMap()* 함수가 호출되도록 등록합니다. *clickMap()* 호출 시에는 매개변수로 클릭 위치를 전달합니다. *clickMap()* 함수 작성은 Step4 단계에서 설명합니다.



### STEP 3\_ 마커 붙이기

마커의 정보를 저장하기 위한 *markers* 객체를 선언하고, *MARKER\_KEY* 상수와 *LINE\_KEY* 상수를 정의합니다.

#### ● map.js 파일

```
var markers = {
  office: {
    prefix: 'http://sstatic.naver.com/search/local/icon3/icos_L',
    offset: 64,
    points: []
  },
  home: {
    prefix: 'http://static.naver.com/local/map_img/set/icos_free_',
    offset: 96,
    points: []
  }
};

var MARKER_KEY = 'marker', LINE_KEY = 'line';
```

- ① *markers* 객체를 선언합니다. *markers*는 마커들의 정보를 담기 위한 객체입니다. *markers.office*는 직장/학교 마커들의 정보를 담고 있고, *markers.home*은 이사 후보지 마커들에 대한 것입니다. *prefix*와 *offset*는 마커 아이콘을 만들기 위해 필요한 값을 저장하고, *points*에는 마커의 위치 정보를 저장합니다.
- ② *MARKER\_KEY*와 *LINE\_KEY*를 정의합니다. *MARKER\_KEY*는 마커를 지도에 붙일 때 사용하는 상수이고, *LINE\_KEY*는 선을 지도에 그릴 때 사용하는 상수입니다. *MARKER\_KEY*와 *LINE\_KEY*는 *NMap.addOverlay()* 메소드와 *NMap.clearOverlays()* 메소드의 매개변수로 사용됩니다. 0이 아닌 모든 값을 가질 수 있습니다.



## STEP 4\_ 이벤트 처리 함수 작성하기

사용자가 지도를 클릭했을 때 그 이벤트를 처리할 `clickMap()`함수를 작성합니다.

### ● map.js 파일

```
function clickMap(pos) {
    var buttons = document.markerForm.markerGroup;
    for (var i = 0; i < buttons.length; i++) if (buttons[i].checked) {
        marker = markers[buttons[i].value];
        var cnt = marker.points.length;
        if (cnt >= 10) {
            alert('이 예제에서 마커는 10개까지만 허용됩니다.');
            continue;
        }
        var ch = String.fromCharCode(marker.offset + cnt + 1);
        var iconUrl = marker.prefix + ch + '.gif';
        var newMarker = new NMark(pos, new NIIcon(iconUrl, new NSize(15, 14)));
        mapObj.addOverlay(newMarker, MARKER_KEY);
        marker.points.push(newMarker.getPoint());
    }
}
```

- ① 체크된 라디오 버튼에 대해서만 마커 생성 작업을 수행할 수 있도록 조건문을 작성합니다. 위 예제에서 `buttons`는 라디오 버튼의 배열이고, `checked` 속성을 확인해서 사용자가 체크한 라디오 버튼을 찾아냅니다.
- ② 마커 개수를 제한하기 위한 코드를 추가합니다. 위 예제에서는 `office`, `home` 각각 10개까지만 마커를 생성할 수 있도록 했습니다. 만약 사용자가 10개 이상 마커를 추가할 경우에는 경고 메시지를 보내고 마커를 생성하지 않습니다.
- ③ `NMark` 클래스로 마커 객체를 생성합니다. `NMark`의 첫 번째 파라미터로 마커 위치를 지정합니다. 마커의 위치 정보는 `clickMap()`의 매개변수로 받은 `pos`와 같습니다. `NMark`의 두 번째 파라미터로 마커 아이콘을 지정합니다. 마커 아이콘은 `NIIcon`의 인스턴스를 만들어서 정의하는데, `NIIcon`의 입력 파라미터로 아이콘 그림 파일의 URL과 아이콘 크기를 입력합니다.
  - a. 마커 아이콘 그림 파일의 위치는 `markers` 객체의 `prefix`와 `offset`을 조합해서 만들었습니다.
  - b. 마커 아이콘의 크기는 `NSize` 클래스로 생성했습니다.
- ④ `NMap.addOverlay()` 메소드를 사용하여 마커를 지도에 추가합니다. `NMap.addOverlay()` 메소드의 매개변수로 마커 객체와 `MARKER_KEY`를 지정합니다. 모든 마커에 대해 같은 `MARKER_KEY`를 사용하면 향후 같은 `MARKER_KEY`를 가진 마커를 한꺼번에 지울 수 있습니다.
- ⑤ 1~4) 과정을 통해 생성한 마커의 위치 정보를 `markers` 객체의 `points` 값으로 저장합니다. 이 정보는 최적 지점을 찾을 때 필요합니다.



## STEP 5\_ 최적지점 찾기

여기까지 해서 지도 위에 이사 후보지와 직장/학교의 위치를 표시하였습니다. 이제 이 위치 정보를 이용해서 최적 지점을 찾는 `find()`함수를 구현합니다.

### ● map.js 파일

```

function find() {
    var near = null;
    var nearDist = Number.MAX_VALUE;
    var offices = markers['office'].points;
    var homes = markers['home'].points; ❶

    for (var i = 0; i < homes.length; i++) {
        var home = homes[i];
        var dist = 0;
        for (var j in offices) {
            dist += offices[j].distance(home); ❷
        }
        if (dist < nearDist) {
            near = home;
            nearDist = dist; ❸
        }
    } ❹

    if (!near) {
        return;
    } ❺

    var minx = near.getX(), maxx = near.getX();
    var miny = near.getY(), maxy = near.getY(); ❻

    for (var i = 0; i < offices.length; i++) {
        office = offices[i];
        minx = Math.min(office.getX(), minx); ❼
        maxx = Math.max(office.getX(), maxx);
        miny = Math.min(office.getY(), miny);
        maxy = Math.max(office.getY(), maxy); ❼
    } ❼

    mapObj.setBound(minx, maxy, maxx, miny); ❻

    mapObj.clearOverlays(LINE_KEY); ❽

    for (var i = 0; i < offices.length; i++) {
        line = new NPolyline(); ❾
        line.addPoints(near, offices[i]);
        mapObj.addOverlay(line, LINE_KEY); ❿
    } ❽
}

```

❶ 최적 지점을 저장할 `near` 객체를 선언합니다.

- ② 이사 후보지에서 직장/학교까지 거리의 총합을 저장할 `nearDist` 변수를 선언합니다. 위 예제 코드에서 `Number.MAX_VALUE`는 `Number` 클래스가 가질 수 있는 최대값을 의미합니다.
- ③ 직장 또는 학교의 위치 정보를 저장할 `offices` 변수를 선언합니다.
- ④ 이사 후보지의 위치 정보를 저장할 `homes` 변수를 선언합니다.
- ⑤ 이사 후보지에서 각 직장/학교까지의 거리를 구해서 합산합니다. 모든 이사 후보지에 대해서 이 작업을 반복한 후, 결과 값이 가장 작은 이사 후보지를 선정하여 `near` 객체에 저장합니다.
- ⑥ 최적 지점을 찾아내고 나면 최적 지점과 모든 직장/학교가 한 화면에 들어오도록 지도의 중심점과 축척을 조절해야 합니다. 최적 지점의 좌표와 모든 직장/학교의 좌표들을 비교해서 x좌표의 최소, 최대값과 y좌표의 최소, 최대값을 구합니다. 변수 `minx, maxx, miny, maxy`에 우선 최적 지점의 좌표를 넣고 이어서 직장/학교들의 좌표를 비교하는 방법으로 구할 수 있습니다. 최적 지점의 좌표 값은 `NPoint` 객체의 `NPoint.getX()` 메소드와 `NPoint.getY()` 메소드를 이용해서 구할 수 있습니다.
- ⑦ `NMap.setBound()` 메소드를 사용하여 한 화면에 최적 지점과 직장/학교가 모두 나올 수 있도록 지도 영역을 설정합니다.
- ⑧ 최적 지점과 직장/학교 마커를 연결하는 선을 그립니다. 지도 객체의 `NMap.clearOverlays()` 메소드를 호출하여 이전에 그렸던 선을 지웁니다. 이 때 `LINE_KEY` 값을 매개변수로 입력합니다. 이렇게 하면 같은 `LINE_KEY` 값을 갖는 선은 한꺼번에 지워집니다.
- ⑨ `NPolyline` 클래스로 `line` 객체를 생성하고, `NMap.addOverlay()` 메소드를 이용하여 지도 위에 선을 그립니다. `NMap.addOverlay()` 메소드 호출 시에는 모든 선이 같은 키 값을 가질 수 있도록 같은 `LINE_KEY` 값을 줍니다.



## STEP 6\_ 마커 삭제

여기까지 해서 최적 지점을 찾아보았습니다. 마지막으로 `clear`버튼을 눌렀을 때 마커를 삭제하는 함수를 구현합니다.

### ● map.js 파일

```
function clearMarkers() {
    for (var attr in markers) {
        markers[attr].points = [];
    }

    mapObj.clearOverlays(MARKER_KEY);
    mapObj.clearOverlays(LINE_KEY);
}
```



- ① `clear` 버튼을 누르면 모든 마커를 삭제하는 `clearMarkers()` 함수를 정의합니다.
- ② `markers` 객체에 저장된 마커의 위치 정보를 지웁니다.
- ③ `NMap.clearOverlays()` 메소드를 사용해서 지도 위에 표시된 마커와 선을 모두 지웁니다.

## 예제 코드 실행 결과

위 예제 코드를 실행하면 다음과 같이 최적 지점이 지도 위에 표시됩니다.

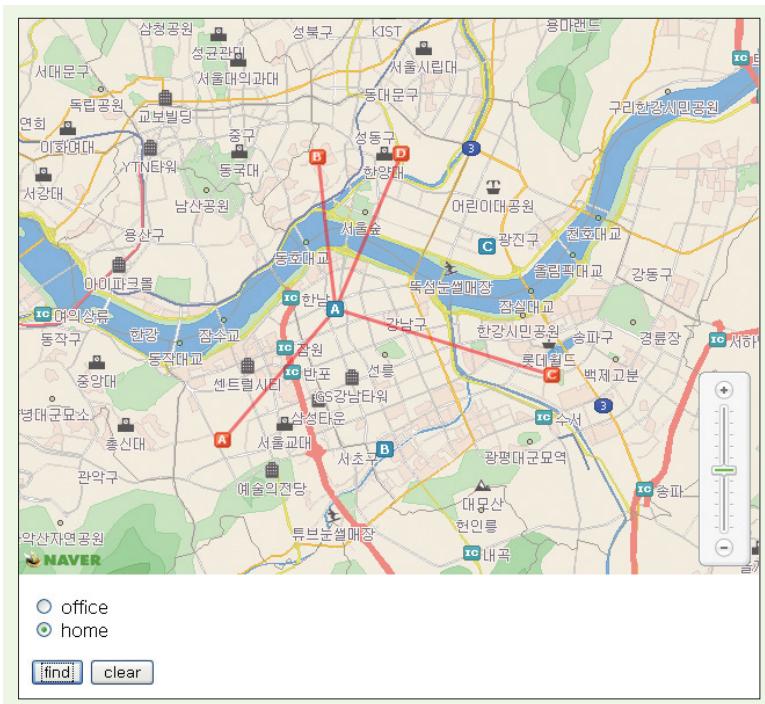


그림 2.4 최적 지점 찾기 예제 실행결과 화면

## 전체 소스코드

### ● map.html 파일

```
<!DOCTYPE HTML SYSTEM>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv="Cache-Control" content="No-Cache">
<meta http-equiv="Pragma" content="No-Cache">
<title>최적지점 찾기 예제</title>
</head>
<body style="margin: 0px 0px 0px 0px">
<!-- 지도-->
<div id='mapContainer' style='width:640px; height:480px'></div>
<div>
<br>
<!-- 마커 선택 라디오 버튼-->
<form name="markerForm" action="">
<input type="radio" name="markerGroup" value="office">
office<br>
<input type="radio" name="markerGroup" value="home">
home<br>
</form>
<!-- 찾기 버튼과 마커 지우기 버튼-->
<input type="button" value="find" onclick="find();">
<input type="button" value="clear" onclick="clearMarkers();">
</div>
<!-- 자바스크립트 불러오기 -->
<script type="text/javascript" src="http://maps.naver.com/js/naverMap.naver?key=YOURMAPKEY">
</script>
<script type="text/javascript" src="map.js"></script>
</body>
</html>
```

### ● map.js 파일

```
var mapObj = new NMap(document.getElementById('mapContainer'), 640, 480);
var zoom = new NZoomControl();
zoom.setAlign("right");
zoom.setVAlign("bottom");
mapObj.addControl(zoom);
mapObj.setBound(0, 999999, 999999, 0);
NEvent.addListener(mapObj, "click", clickMap);

var markers = {
    office: {
        prefix: 'http://sstatic.naver.com/search/local/icon3/icos_L',
        offset: 64,
        points: []
    },
    home: {
```

```
prefix: 'http://static.naver.com/local/map_img/set/icos_free_',
offset: 96,
points: []}
};

var MARKER_KEY = 'marker', LINE_KEY = 'line';

function clickMap(pos)
{
    var buttons = document.markerForm.markerGroup;

    for (var i = 0; i < buttons.length; i++) if (buttons[i].checked) {
        marker = markers[buttons[i].value];

        var cnt = marker.points.length
        if (cnt >= 10) {
            alert('이 예제에서 마커는 10개까지만 허용됩니다.');
            continue;
        }

        var ch = String.fromCharCode(marker.offset + cnt + 1);
        var iconUrl = marker.prefix + ch + '.gif';
        var newMarker = new NMark(pos, new NIIcon(iconUrl, new NSize(15, 14)));
        mapObj.addOverlay(newMarker,MARKER_KEY);
        marker.points.push(newMarker.getPoint());
    }
}

function find() {
    var near = null;
    var nearDist = Number.MAX_VALUE;
    var offices = markers['office'].points;
    var homes = markers['home'].points;

    for (var i = 0; i < homes.length; i++) {
        var home = homes[i];
        var dist = 0;
        for (var j in offices) {
            dist += offices[j].distance(home);
        }
        if (dist < nearDist) {
            near = home;
            nearDist = dist;
        }
    }

    if (!near) {
        return;
    }

    var minx = near.getX(), maxx = near.getX();
    var miny = near.getY(), maxy = near.getY();
```

```
for (var i = 0; i < offices.length; i++) {
    office = offices[i];
    minx = Math.min(office.getX(), minx);
    maxx = Math.max(office.getX(), maxx);
    miny = Math.min(office.getY(), miny);
    maxy = Math.max(office.getY(), maxy);
}

mapObj.setBound(minx, maxy, maxx, miny);

mapObj.clearOverlays(LINE_KEY);

for (var i = 0; i < offices.length; i++) {
    line = new NPolyline();
    line.addPoints(near, offices[i]);
    mapObj.addOverlay(line, LINE_KEY);
}
}

function clearMarkers() {
    for (var attr in markers) {
        markers[attr].points = [];
    }
}

mapObj.clearOverlays(MARKER_KEY);
mapObj.clearOverlays(LINE_KEY);
}
```

## 약도 만들기 예제

### 문제 상황

브랜드샵 웹페이지를 만들다가 약도를 넣으면 고객이 찾아오기 편리하겠다는 생각을 했습니다. 그래서 네이버 지도검색 서비스를 이용해서 브랜드샵 위치를 검색한 후에 화면캡처 프로그램을 써서 약도 이미지를 저장했습니다. 그런 다음 웹페이지에 이미지를 업로드합니다. 그런데 이건 웬지 복잡하고 불편합니다. 좀 더 편한 방법이 있을 것 같습니다.

이런 경우 약도를 HTML 코드로 만들어주는 애플리케이션이 있다면, 웹페이지 HTML 문서에 약도 코드를 간단히 넣어주기만 하면 될 것 같습니다. 네이버 지도API를 이용하면 약도를 HTML 코드로 생성해 주는 애플리케이션을 만들 수 있습니다.

### 준비하기

약도 만들기 예제를 따라해 보기 전에 준비해야 할 것이 있습니다.

#### A. 필수사항

- 네이버 OpenAPI 키가 필요합니다.
- 네이버 지도 API 키가 필요합니다.
- 지도 API를 사용하려면 도메인 주소 또는 IP가 필요하므로 서버가 필요합니다.
- PHP5 이상 지원하는 웹 서버가 필요합니다.
- 자바스크립트 실행을 위해서 FireFox 1.5 이상 또는 Microsoft Internet Explorer 6 이상을 사용해야 합니다.

#### B. 권장사항

- 예제는 HTML과 자바스크립트와 PHP로 작성되어 있습니다. 따라서 HTML과 자바스크립트와 PHP로 작성된 코드를 이해할 수 있어야 합니다. 네이버 지도 API를 다룰 수 있어야 합니다. 지도 API가 처음이신 분은 이 문서의 두 번째 예제를 먼저 읽어보시기 바랍니다.
- 네이버 지도 API에 대한 사용자 문서를 읽어두면 예제를 이해하는데 도움이 됩니다. 지도 API 사용자 문서는 네이버 OpenAPI 사이트(<http://openapi.naver.com/>)에서 찾아볼 수 있습니다.

### 구현하기

네이버 지도API를 이용하여 약도 화면에 목적지를 표시한 후, 그 화면을 HTML 코드로 생성해 주는 애플리케이션을 구현하겠습니다.



#### STEP 1\_ 걸모습 만들기

약도가 출력되도록 화면을 구성합니다.

##### ● sketchmap.php 파일

```
<div id="mapContainer" style="width:640px; height:480px;"></div>
<script type="text/javascript" src="http://maps.naver.com/js/naverMap.naver?key=YOURMAPKEY">
</script>
<script type="text/javascript">
var mapObj = new NMap(document.getElementById('mapContainer'));
```

①

②

③

```

var zoom = new NZoomControl();
zoom.setAlign("right");
zoom.setValign("bottom");
mapObj.addControl(zoom);
</script>

```

- ① 약도를 출력할 컨테이너를 설정합니다. 위 예제에서는 div 엘리먼트를 사용하여 설정하였습니다.
- ② 네이버 지도 API 자바스크립트를 불러 옵니다. 이때 key 파라미터로 여러분이 발급받은 지도 API 키를 입력해야 합니다. 위 예제에서 YOURMAPKEY라고 표현된 부분에 여러분이 발급받은 지도API 키를 입력합니다.
- ③ *NMap* 클래스로 약도 객체를 생성합니다. 약도 객체를 생성할 때는 약도를 출력할 컨테이너 정보를 첫 번째 파라미터로 전달해야 합니다.
- ④ 사용자가 약도 화면을 축소 또는 확대할 수 있도록 줌 컨트롤 객체를 생성합니다. *NZoomControl* 클래스로 줌 컨트롤 객체를 생성한 후에, *NZoomControl.setAlign()*와 *NZoomControl.setValign()* 메소드를 사용하여 줌 컨트롤 객체의 표시 위치를 설정합니다. 그런 다음 *NMap.addControl()* 메소드로 줌 컨트롤 객체를 약도에 추가합니다. 이렇게 하면 축척 슬라이드 바가 약도 화면에 표시됩니다.



## STEP 2\_ 마커 표시하기

사용자가 약도 위에 목적지를 표시하면 약도에 마커 아이콘이 표시되도록 자바스크립트 코드를 작성합니다.

### ● sketchmap.php 파일

```

var markerUrl = 'http://ssstatic.naver.com/search/local/icon3/icos_LA.gif';

function makeMarker(x, y) {
    var icon = new NIIcon(markerUrl, new NSize(15, 14));
    var newMarker = new NMark(new NPoint(x, y), icon);
    mapObj.addOverlay(newMarker);
    return newMarker;
}

function clickMap(pos) {
    if (marker) {
        marker.setPoint(pos);
    }
    else {
        marker = makeMarker(pos.getX(), pos.getY());
    }
}

NEvent.addListener(mapObj, "click", clickMap);

```

- ① *NMark* 클래스로 마커 객체를 생성합니다. *NMark*의 첫 번째 파라미터로 마커 위치를 지정합니다. 마커의 위치 정보는 *clickMap()*의 매개변수로 받은 'pos'와 같습니다. *NMark*의 두 번째 파라미터로 마커 아이콘을 지정합니다. 마커 아이콘은 *NIIcon*의 인스턴스를 만들어서 정의하는데, *NIIcon*의 입력 파라미터로 아이콘 그림 파일의 URL과 아이콘 크기를 입력합니다. 위 예제 코드에서 마커 아이콘의 URL은 *markerUrl*로 선언하였습니다.
- ② *clickMap()* 함수를 정의합니다. 이미 표시된 마커가 있는 경우에는 *NMark.setPoint()* 메소드로 마커의 위치만 변경

하고, 마커가 없는 경우에는 makeMarker() 함수를 호출해서 마커를 새로 만듭니다.

- ③ `NEvent.addListener()` 메소드를 사용하여 click이벤트를 등록하고, 이벤트가 발생할 때 실행할 코드를 등록합니다.  
즉 약도 객체를 클릭했을 때 clickMap() 함수가 호출되도록 등록합니다. `NEvent.addListener()` 메소드는 매개변수로 클릭 위치를 전달합니다.

## ▼ STEP 3\_ 지역정보/주소정보 검색창 만들기

약도 밑에 검색창을 추가하여 지역검색 또는 주소검색을 할 수 있도록 구현합니다.

● `sketchmap.php` 파일

```
<form id="search" method="get" action="=$_SERVER['PHP_SELF']?">
  addr
  <input type="text" name="query">
  <input type="submit" value="search">
</form>
```

사용자가 상호명 또는 주소로 약도를 검색할 수 있도록 검색창을 구성합니다. 이 예제에서는 ‘addr’이라는 글상자 한 개와 ‘search’라는 버튼 한 개로 구성된 폼을 만들었습니다.

## ▼ STEP 4\_ 지역정보/주소정보 검색 처리하기

검색창에서 입력 받은 단어를 검색하도록 아래와 같이 코드를 작성합니다.

● `sketchmap.php` 파일

```
<?php
define('NAVERKEY', 'YOURNAVERKEY'); ❶
define('MAPKEY', 'YOURMAPKEY'); ❷

$names = "";
$addresses = "";
$query = null;

function makeItem($title, $mapx, $mapy) { ❸
    return "<a href='#' onclick='javascript:map0obj.setCenterAndZoom(new NPoint($mapx, $mapy),1);'$title<br></a>";
}

if (isset($_GET["query"])) { ❹
    $query = trim($_GET["query"]);
    if (strlen($query) > 0) { ❺
        $encodedquery = urlencode($query);
        $url1 =
            "http://openapi.naver.com/search?query=$encodedquery&target=local&sort=vote&key=" ❻
            .NAVERKEY;
        $name = simplexml_load_file($url1)->channel; ❼
    }
    foreach ($name->item as $item) {
        $names .= makeItem($item->title, $item->mapx, $item->mapy); ❽
    }
    $encodedquery = urlencode(iconv('utf-8', 'euc-kr', $query)); ❾
}
```

```

$url2 = "http://maps.naver.com/api/geocode.php?query=$encodedquery&key=".MAPKEY;
$address = simplexml_load_file($url2);

foreach ($address->item as $item) {
    $point = $item->point;
    $addresses .= makeItem($item->address, $point->x, $point->y);
}
}

?>

<div>
<?php
echo $names;
echo $addresses;
?>
</div>

```

- ❶ 네이버 검색 API 키를 정의합니다. define문의 두 번째 파라미터(위 예제에서 YOURNAVERKEY로 표현)로 여러분이 발급 받은 검색 API 키를 입력합니다.
- ❷ 지도API키를 정의합니다. define문의 두 번째 파라미터(위 예제에서 YOURMAPKEY로 표현)로 여러분이 발급받은 지도 API키를 입력합니다.
- ❸ makeItem() 함수를 작성합니다. 사용자가 지도검색 또는 주소검색 결과를 클릭할 때 해당 위치를 약도에 나타내도록 구현해야 합니다. 즉, 검색 결과에 대한 링크를 만들어주고, NMap.setCenterAndZoom() 메소드를 사용하여 약도의 중심과 축척을 설정함으로써 검색 결과가 최적의 상태로 약도에 나타날 수 있게 설정합니다.
- ❹ isset() 함수로 \$\_GET["query"]라는 값이 있는지 확인하고, 값이 있는 경우 그 값을 얻어옵니다.
- ❺ 얻어온 값에서 불필요한 공백을 trim() 함수를 사용하여 제거한 후 \$query 변수에 저장합니다.
- ❻ \$query의 크기가 0보다 크면 지역검색과 주소검색을 수행합니다.
- ❼ 먼저 지역검색을 하려면 urlencode() 함수로 \$query를 처리한 후, \$url1을 만들어서 지역검색API를 호출합니다. simplexml\_load\_file() 함수를 사용해서 지역검색 결과를 얻어옵니다. simplexml\_load\_file() 함수는 지역검색 결과로 얻어온 XML 문서를 객체로 만들어 줍니다. 이 객체의 channel->item 멤버에 상호명과 좌표가 저장되어 있습니다. channel->item 멤버에 저장된 상호명과 좌표를 읽어올 때마다 makeItem() 함수를 호출합니다. 모든 상호명과 좌표명을 읽어와야 하므로, makeItem() 함수는 반복적으로 호출됩니다. makeItem()에서 반환받은 값을 붙여서 \$names 변수에 저장합니다.

**참고** ● 네이버 OpenAPI의 검색 결과로 반환되는 XML 문서는 모두 RSS(Really Simple Syndication) 형식으로 되어 있습니다. RSS 문서에 대한 자세한 설명은 이 문서의 “부록. RSS 이해하기”를 참조하시기 바랍니다.

- ❽ 주소검색을 하려면 검색어를 euc-kr로 인코딩해야 합니다. 주소검색 URL을 생성하기 전에 iconv() 함수를 사용해서 검색어의 인코딩을 변경합니다. 그런 다음 \$url2를 생성하여 주소검색을 수행합니다. simplexml\_load\_file() 함수를 사용하여 주소검색 결과로 얻어온 XML 문서를 객체로 만듭니다. 이 객체의 item->address 멤버에서 주소검색 결

과를 얻어오고, `item->point` 멤버에서 좌표를 읽어옵니다. 모든 주소와 좌표에 대해서 반복적으로 `makeItem()` 함수를 호출합니다. `makeItem()`에서 반환받은 값을 붙여서 `$addresses` 변수에 저장합니다.

- ⑨ 검색 결과 중 사용자가 선택한 결과는 `$names`와 `$addresses` 변수에 저장됩니다. 이 변수의 내용이 출력될 수 있도록 약도 아래에 품을 생성합니다.

## STEP 5\_ 약도 크기 조절 기능 구현하기

약도의 크기를 조절할 수 있는 기능을 구현하겠습니다. 이 단계는 선택사항이므로, 사용하지 않을 경우에는 약도 컨테이너 생성 시 품 크기를 확정합니다. 이 기능을 사용할 경우에는 Step1의 컨테이너 생성 코드를 다음 코드로 대체합니다.

### ● sketchmap.php 파일

```
<div id="mapContainer" style="width:<?=$mapWidth?>px; height:<?=$mapHeight?>px;"></div>
```

## STEP 5-1\_ 크기 조절용 품 생성하기

사용자가 약도 크기를 입력할 수 있도록 품을 생성합니다.

### ● sketchmap.php 파일

```
<form id="resize" method="get" action=<?=$_SERVER['PHP_SELF']?>>
  size
  <input type="text" name="mapWidth" style="width: 40px;">
  <input type="text" name="mapHeight" style="width: 40px;">
  <input type="submit" value="resize">
</form>
```

Step3에서 만든 검색창 위에 크기 조절용 품을 생성합니다. 즉 약도의 너비와 높이를 설정할 수 있는 글상자 2개와 `resize` 버튼을 생성합니다.

## STEP 5-2\_ 약도 크기 제어하기

크기 조절을 처리해 줄 PHP 코드를 작성합니다.

### ● sketchmap.php 파일

```
$mapWidth = 640; ①
$mapHeight = 480;
if (isset($_GET["mapWidth"])) {
    $mapWidth = trim($_GET["mapWidth"]);
    $mapHeight = trim($_GET["mapHeight"]);
}
} ③
```

① `$mapWidth`, `$mapHeight` 변수를 선언합니다.

② `isset()` 함수로 `$_GET["mapWidth"]`라는 값이 있는지 확인합니다. 만약 있다면 약도 크기에 대한 정보가 모두 있는 것으로 간주할 수 있습니다.

③ `$_GET["mapWidth"]`와 `$_GET["mapHeight"]`를 얻어오고 불필요한 공백을 `trim()` 함수를 사용하여 제거한 후, `$mapWidth`와 `$mapHeight` 변수를 얻어온 값으로 갱신합니다.

### STEP 5-3\_ 크기 설정용 글상자에 현재 크기 표시하기

크기 설정용 글상자에도 현재 크기가 표시되도록 작성합니다.

#### ● sketchmap.php 파일

```
<form id="resize" method="get" action=<?=$_SERVER['PHP_SELF']?>>
  size
  <input type="text" name="mapWidth" value=<?=$mapWidth?>" style="width: 40px;">
  <input type="text" name="mapHeight" value=<?=$mapHeight?>" style="width: 40px;">
  <input type="submit" value="resize">
</form>
```



### STEP 6\_ 축척, 위치, 마커 정보 저장하기

약도의 크기를 재설정하면 페이지를 다시 불러오게 되므로 약도의 위치와 축척, 마커가 리셋됩니다. 이미 찾은 정보가 없어지면 불편하므로 정보가 사라지지 않도록 만들어줄 필요가 있습니다. 이런 경우에는 submit 시 이벤트 핸들러를 추가하여 정보를 서버에 보내주고, 크기 재설정 등의 이유로 페이지를 다시 불러올 때 서버가 페이지에 이 정보를 적용하도록 구현합니다. 약도 정보는 약도 크기 조절 시에도 리셋되지만, 지역검색/주소검색 시에도 리셋됩니다. 따라서 약도 크기 재설정 기능을 사용하지 않을 경우에도 이 단계는 수행하는 것이 좋습니다.

### STEP 6-1\_ 이벤트 핸들러 추가하기

약도 크기 조절 시에 약도 정보를 보존하기 위하여 크기 조절용 폼에 이벤트 핸들러를 추가합니다.

#### ● sketchmap.php 파일

```
<form id="resize" method="get" action=<?=$_SERVER['PHP_SELF']?>" onsubmit="javascript:addCommons(this);> ①
```

① 크기 조절용 폼에서 resize 버튼을 누르면 addCommons() 함수를 호출하여 폼에 약도의 위치와 축척에 대한 정보를 추가합니다.

```
function addCommons(form) {
  var center = mapObj.getCenter();
  addHiddens(form, {'zoom': mapObj.getZoom(), 'x': center.getX(), 'y': center.getY()});
  if (marker) {
    var point = marker.getPoint();
    addHiddens({'markerX': point.getX(), 'markerY': point.getY()});
  }
}
```

```
function addHiddens(form, dict) {
  for (name in dict) {
    hidden = document.createElement('input');
    hidden.type = 'hidden';
    hidden.name = name;
    hidden.value = dict[name];
    form.appendChild(hidden);
  }
}
```

②

③

- ② `addCommons()` 자바스크립트 함수를 작성합니다. `addCommons()` 함수는 품에 정보를 추가합니다. 우선 축척과 약도의 중심점에 대한 정보를 `addHiddens()` 메소드를 이용해 추가합니다. 중심점은 `NMap.getCenter()` 메소드로 얻을 수 있습니다. 마커가 있는 경우에는 마커의 위치도 추가합니다. 마커의 위치는 `NMark.getPoint()` 메소드로 얻습니다.
- ③ `addHiddens()` 자바스크립트 함수를 작성합니다. `addHiddens()` 함수는 dict 배열에 들어있는 내용을 `hidden` 태입의 `input` 엘리먼트로 만들어서 품에 추가해 주는 역할을 합니다.
- ④ 이런 과정을 통해 품에 추가된 정보는 사용자 지정 크기 값과 함께 cgi parameter 형태로 서버에 전송됩니다. URL로 표현하면 다음과 같은 형식입니다.

```
sketchmap.php?mapWidth=640&mapHeight=480&zoom=11&x=499712&y=499712&markerX=381952&markerY=450560
```

## STEP 6-2\_ 약도 정보 읽어오기

cgi parameter로 전송했던 약도 정보들을 다시 읽어오는 코드를 작성합니다.

### ● sketchmap.php 파일

```
$mapWidth= 640;
$mapHeight = 480;
$zoom = 11;
$x = 500000;
$y = 500000;
$marker = false;
$markerX = -1;
$markerY = -1;

if (isset($_GET["mapWidth"])) {
    $mapWidth = trim($_GET["mapWidth"]);
    $mapHeight = trim($_GET["mapHeight"]);
    $zoom = trim($_GET["zoom"]);
    $x = trim($_GET["x"]);
    $y = trim($_GET["y"]);
    if (isset($_GET["markerX"])) {
        $marker = true;
        $markerX = trim($_GET["markerX"]);
        $markerY = trim($_GET["markerY"]);
    }
}
```

- ① 약도의 정보를 담고 있는 변수를 초기화합니다.
- ② `array_key_exists()` 함수로 `$_GET` 배열에 “`mapWidth`”라는 키가 들어있는지 확인하고, 키가 들어있을 경우 약도 정보를 담고 있는 파라미터를 받은 것으로 간주하고 모든 정보를 읽습니다.
- ③ `$marker`에는 마커 생성 여부를 저장합니다.

## STEP 6-3\_ 약도 정보 적용하기

읽어온 약도 정보를 적용하도록 코드를 작성합니다.

### ● sketchmap.php 파일

```

var marker = null;
</php
if ($marker) {
    echo "marker = makeMarker($markerX, $markerY);";
}
?>
mapObj.setZoom(<?=$zoom?>);
mapObj.setCenter(new NPoint(<?=$x?>, <?=$y?>));

```

- ① \$marker가 true인 경우에만 마커를 생성합니다. 그렇지 않은 경우에는 마커를 null 상태로 유지합니다.
- ② *NMap.setZoom()* 메소드를 사용하여 약도의 축척 정보를 화면에 반영합니다.
- ③ *NMap.setCenter()* 메소드를 사용하여 약도의 중심점 정보를 화면에 반영합니다. 이때 *NPoint* 클래스를 사용하여 좌표를 새로 생성합니다.

**참고** ● 지역검색/주소검색 시에도 위와 같은 절차로 약도 정보를 보존할 수 있습니다. 즉, 지역검색/주소검색을 위해 search 버튼을 클릭하면 다음과 같이 find() 함수를 호출하도록 구현합니다.

```
<form id="search" method="get" action="sketchmap.php" onsubmit="javascript:find(this);">
그런 다음 find() 함수를 구현합니다. 약도 위치, 축척, 마커 정보 외에 약도 너비, 높이 정보도 담아야 합니다. 약도 위치, 축척, 마커 정보 외에 지도의 너비와 높이 정보도 저장합니다. 다음과 같이 구현합니다.
```

```
function find(form) {
    addCommons(form);
    addHiddens(form, {'mapWidth': <?=$mapWidth?>, 'mapHeight': <?=$mapHeight?>});
}
```



## STEP 7\_ HTML 코드 생성하기

이제 만들어진 약도를 HTML코드로 생성합니다.

### STEP 7-1\_ HTML 생성 결과 출력창 만들기

#### ● sketchmap.php 파일

```

<input type="button" value="html" onclick="javascript:genHtml();"><br>
<textarea cols="75" rows="20" id="result">
</textarea>

```

- ① HTML 코드 생성 버튼을 만들고, 버튼을 클릭했을 때 genHtml() 함수를 호출하도록 합니다.
- ② 생성된 HTML 코드가 출력될 글상자를 생성합니다. 글상자 위치는 지역검색/주소검색 검색 결과 출력 부분 아래가 적절합니다.

## STEP 7-2\_ 기존 코드 재활용하기

HTML코드를 생성할 때 최대한 기존에 작성했던 코드를 재활용합니다. 다음 예제에서는 지도 API 스크립트를 불러오는 코드와 지도를 초기화하는 코드를 재활용하도록 구현했습니다.

### ● sketchmap.php 파일

```
<div id="mapapi">
<script type="text/javascript" src="http://maps.naver.com/js/naverMap.naver?key=<?=MAPKEY?>">
</script>
</div>
```

① 앞에서 작성했던 지도 API 자바스크립트를 불러오는 부분을 div 태그로 감싸고 엘리먼트 ID를 부여합니다.

### ● sketchmap.php 파일

```
<script type="text/javascript" id="initMap">
var mapObj = new NMap(document.getElementById('mapContainer'));
var zoom = new NZoomControl();
zoom.setAlign("right");
zoom.setValign("bottom");
mapObj.addControl(zoom);
</script>

<script type="text/javascript">
NEvent.addListener(mapObj, "click", clickMap);
var marker = null;
<?php
if ($marker) {
    echo "marker = makeMarker($markerX, $markerY);";
}
?>
mapObj.setZoom(<?=$zoom?>);
mapObj.setCenter(new NPoint(<?=$x?>, <?=$y?>));
</script>
```

② 그 외 재활용이 가능한 부분에 엘리먼트 ID를 부여합니다. 위 예제에서는 지도 API 스크립트 파일을 불러오는 부분에 mapapi라는 ID를, 지도 초기화 부분에는 initMap이라는 ID를 부여했습니다. 이렇게 ID를 부여하면 나중에 HTML 코드를 생성할 때 document.getElementById() 메소드를 이용해 해당 엘리먼트를 가져와서 사용할 수 있습니다.

③ 지도 클릭 이벤트 등록 및 마커 생성 부분처럼 재활용할 수 없는 부분은 따로 분리합니다.

## STEP 7-3\_ HTML 코드 생성 함수 작성하기

약도에 대한 HTML 코드를 생성하는 getHtml() 함수를 작성합니다. 이 함수에서 생성하는 HTML 코드에는 다음과 같은 내용이 포함됩니다.

- 지도 API 불러오기
- 약도 위치 및 축척 설정

- 마커 설정
- 약도 크기 설정 및 그리기

### ● sketchmap.php 파일

```

function genHtml() {
    var code =
        "mapObj.setCenterAndZoom(new NPoint(" + mapObj.getCenter() + "), " +
        mapObj.getZoom() + ");\n" ] 1

    if (marker) {
        var point = marker.getPoint();
        code += "var pos = new NPoint(" +
            point.getX() + ", " + point.getY() + ");\n" +
            "marker = new NMarker(pos, new NIIcon('" + markerUrl +
            "', new NSize(15, 14)));;\n" +
            "mapObj.addOverlay(marker);";
    } ] 2

    mapStyle = document.getElementById('mapContainer').style;
    res = document.createTextNode(
        '<div id="mapContainer" style="width:' + mapStyle.width +
        '; height:' + mapStyle.height + '";></div>' +
        document.getElementById('mapapi').innerHTML +
        '<script type="text/JavaScript">' +
        document.getElementById('script').innerHTML +
        code + '\n</script>');
    ] 3

    var textarea = document.getElementById('result');
    child = textarea.childNodes[0];
    ] 4

    if (child) {
        textarea.replaceChild(res, child);
    } ] 5
    else {
        textarea.appendChild(res);
    }
}

```

- ① 약도의 위치와 축척을 설정하는 코드를 작성합니다. *NMap.setCenterAndZoom()* 메소드를 사용하면 한번에 설정할 수 있습니다.
- ② 마커가 있다면 마커를 약도에 표시하는 코드도 작성합니다. Step 2에서 작성한 코드와 거의 같습니다.
- ③ 약도가 들어가 있는 컨테이너의 크기를 알아내기 위해서 그 컨테이너의 style을 가져와야 합니다. 약도가 들어있는 컨테이너의 ID는 *mapContainer* 이므로 *document.getElementById()* 메소드로 사용해서 style을 얻습니다.
- ④ 위에서 작성한 코드와 Step 7-2)에서 재활용하기로 한 코드를 이어 붙여서 최종적으로 HTML 코드를 생성합니다.
- ⑤ 생성한 HTML 코드는 약도 화면 최하단의 *textarea*에 들어갑니다. *textarea*에 기존 텍스트가 들어 있었다면 *replaceChild()* 메소드로 교체하고 아무것도 없었다면 *appendChild()* 메소드로 추가합니다.

## 예제 코드 실행 결과



그림 2.5 약도 만들기 예제 실행 결과 화면

## 전체 소스코드

```
<?php
define('MAPKEY', 'YOURMAPKEY');
define('NAVERKEY', 'YOURNAVERKEY');

$names = "";
$addresses = "";
$mapWidth= 640;
$mapHeight = 480;
$zoom = 11;
$x = 500000;
$y = 500000;
$marker = false;
$markerX = -1;
$markerY = -1;
$query = null;

function makeItem($title, $mapx, $mapy) {
    return "<a href='#' onclick='javascript:mapObj.setCenterAndZoom(new NPoint($mapx, $mapy), 1);'>$title<br></a>";
}

if (isset($_GET["query"])) {
    $query = trim($_GET["query"]);
    if(strlen($query) > 0){
        $encodedquery = urlencode($query);
        $url =
            "http://openapi.naver.com/search?query=$encodedquery&target=local&sort=vote&key=".NAVERKEY;
        $name = simplexml_load_file($url)->channel;

        foreach($name->item as $item) {
            $names .= makeItem($item->title, $item->mapx, $item->mapy);
        }

        // 주소 검색의 처리
        $encodedquery = urlencode(iconv('utf-8', 'euc-kr', $query));
        $url2 = "http://maps.naver.com/api/geocode.php?query=$encodedquery&key=".MAPKEY;
        $address = simplexml_load_file($url2);

        foreach($address->item as $item) {
            $point = $item->point;
            $addresses .= makeItem($item->address, $point->x, $point->y);
        }
    } // endif
} // endif

// 지도 너비, 높이, 측척, 위치, 마커 위치 설정
if (isset($_GET["mapWidth"])) {
    $mapWidth = trim($_GET["mapWidth"]);
    $mapHeight = trim($_GET["mapHeight"]);
```

```
$zoom = trim($_GET["zoom"]);
$x = trim($_GET["x"]);
$y = trim($_GET["y"]);
if (isset($_GET["markerX"])) {
    $marker = true;
    $markerX = trim($_GET["markerX"]);
    $markerY = trim($_GET["markerY"]);
}
?>

<!DOCTYPE HTML SYSTEM>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv="Cache-Control" content="No-Cache">
<meta http-equiv="Pragma" content="No-Cache">
<title>약도 예제</title>
</head>
<body style="margin: 0px 0px 0px 0px">

<div id="mapContainer" style="width:<?=$mapWidth?>px; height:<?=$mapHeight?>px"></div>
<div id="mapapi">
<script type="text/javascript" src="http://maps.naver.com/js/naverMap. naver?key=<?=$MAPKEY?>">
</script>
</div>

<script type="text/javascript">
var markerUrl = 'http://sstatic.naver.com/search/local/icon3/icos_LA.gif';

// 마커를 새로 만들어서 반환하는 함수
function makeMarker(x, y) {
    if (x < 0 || y < 0) {
        return null;
    }
    var icon = new NIIcon(markerUrl, new NSize(15, 14));
    var newMarker = new NMark(new NPoint(x, y), icon);
    mapObj.addOverlay(newMarker);
    return newMarker;
}

// 약도를 HTML 코드로 만들어서 textarea에 출력하는 함수
function genHtml() {
    // HTML 코드를 생성해서 TextNode로 만듦
    var code =
        "mapObj.setCenterAndZoom(new NPoint(" + mapObj.getCenter() + "), " +
        mapObj.getZoom() + ");\n";

    if (marker) {
        var point = marker.getPoint();
        code += "var pos = new NPoint(" +
            point.getX() + ", " + point.getY() + ");\n" +
            "mapObj.addOverlay(pos);\n"
    }
}
```

```
        "marker = new NMark(pos, new NIcon('" + markerUrl +
        "', new NSize(15, 14)));\\n" +
        "mapObj.addOverlay(marker);";
    }

    mapStyle = document.getElementById('mapContainer').style;
    res = document.createTextNode(
        '<div id="mapContainer" style="width: ' + mapStyle.width +
        '; height: ' + mapStyle.height + ';"></div>' +
        document.getElementById('mapapi').innerHTML +
        '<script type="text/JavaScript">' +
        document.getElementById('initMap').innerHTML +
        code + '\\n</script>');

    // textarea를 얹어서 만든 TextNode를 삽입 혹은 기존의 child와 교체
    var textarea = document.getElementById('result');
    child = textarea.childNodes[0];
    if (child) {
        textarea.replaceChild(res, child);
    }
    else {
        textarea.appendChild(res);
    }
}

// form에 공통적으로 필요한 지도 위치, 축척, 마커 위치의 정보를 넣어주는 함수
function addCommons(form) {
    var center = mapObj.getCenter();
    if (marker) {
        var point = marker.getPoint();
        addHiddens(form, {'markerX': point.getX(), 'markerY': point.getY()});
    }
    addHiddens(form, {'zoom': mapObj.getZoom(), 'x': center.getX(), 'y': center.getY()});
}

// 주소/지역검색 버튼을 눌렀을때 호출되는 함수.
// 지도 관련 정보들을 모두 form에 넣어 줌.
function addHiddens(form, dict) {
    for (name in dict) {
        hidden = document.createElement('input');
        hidden.type = 'hidden';
        hidden.name = name;
        hidden.value = dict[name];
        form.appendChild(hidden);
    }
}

// 지역검색 버튼을 눌렀을때 호출되는 함수.
// 지도 관련 정보들을 모두 form에 넣어 줌
function find(form) {
    addCommons(form);
    addHiddens(form, {'mapWidth': <?=$mapWidth?>, 'mapHeight': <?=$mapHeight?>});
}
```

```
}
```

```
// 지도를 눌렀을 때 호출되는 함수. 마커를 새로 지정.
function clickMap(pos) {
    if (marker) {
        marker.setPoint(pos);
    }
    else {
        marker = makeMarker(pos.getX(), pos.getY());
    }
}
</script>

<div>
<!-- 크기 조절 폼--&gt;
&lt;form id="resize" method="get" action=&lt;?=$_SERVER['PHP_SELF']?&gt;" onsubmit="javascript:
addCommons(this);"&gt;
    size
    &lt;input type="text" name="mapWidth" value=&lt;?=$mapWidth?&gt;" style="width: 40px;"&gt;
    &lt;input type="text" name="mapHeight" value=&lt;?=$mapHeight?&gt;" style="width: 40px;"&gt;
    &lt;input type="submit" value="resize"&gt;
&lt;/form&gt;
<!-- 상호/주소 검색 폼--&gt;
&lt;form id="search" method="get" action=&lt;?=$_SERVER['PHP_SELF']?&gt;" onsubmit="javascript:
find(this);"&gt;
    addr
    &lt;input type="text" name="query"&gt;
    &lt;input type="submit" value="search"&gt;
&lt;/form&gt;
&lt;/div&gt;
<!-- 상호/주소 검색 결과 출력 --&gt;
&lt;div&gt;
&lt;?php
echo $names;
echo $addresses;
?&gt;
&lt;/div&gt;
<!-- html 코드 출력 버튼/글상자 --&gt;
&lt;input type="button" value="html" onclick="javascript:genHtml();"&gt;&lt;br&gt;
&lt;textarea name="result" cols="75" rows="20" id="result"&gt;
&lt;/textarea&gt;

<!-- 지도 초기화--&gt;
&lt;script type="text/javascript" id="initMap"&gt;
    var mapObj = new NMap(document.getElementById('mapContainer'));
    var zoom = new NZoomControl();
    zoom.setAlign("right");
    zoom.setValign("bottom");
    mapObj.addControl(zoom);
&lt;/script&gt;
<!-- 지도 클릭 이벤트 등록, 마커 생성 --&gt;
&lt;script type="text/javascript"&gt;</pre>
```

```
NEvent.addListener(mapObj, "click", clickMap);
var marker = null;
<?php
if ($marker) {
    echo "marker = makeMarker($markerX, $markerY);";
}
?>
mapObj.setZoom(<?=$zoom?>);
mapObj.setCenter(new NPoint(<?=$x?>, <?=$y?>));
</script>

</body>
</html>
```

## 지도/블로그 검색 예제

### 문제 상황

분당 정자동에서 커피 맛이 좋은 카페를 찾아서 친구와 만나려고 합니다. 그런데 친구는 분당 지리를 잘 모릅니다. ‘정자동 커피’로 검색한 결과를 지도 위에 표시한 다음 지도를 친구에게 공유하면 참 좋겠다는 생각을 합니다.

이런 경우 네이버 지도 서비스와 네이버 지역검색 서비스, 그리고 네이버 블로그 검색을 통합한 서비스를 만드는 겁니다. 네이버 지도를 화면에 나타낸 다음, 맛있는 커피 집을 검색하고, 그 결과를 지도 위에 표시해 주는 것입니다.

네이버 지도API와 지역검색 API, 그리고 블로그 검색 API를 사용하면 이런 일을 하는 애플리케이션을 만들 수 있습니다.

### 준비하기

지도/블로그 검색 예제를 따라해 보기 전에 준비해야 할 사항이 있습니다.

#### A. 필수사항

- 네이버 OpenAPI 키가 필요합니다.
- 네이버 지도 API 키가 필요합니다.
- 지도 API를 사용하려면 도메인 주소 또는 IP가 필요하므로 서버가 필요합니다.
- PHP5 이상 지원하는 웹 서버가 필요합니다.
- 자바스크립트 실행을 위해서 FireFox 1.5 이상 또는 Microsoft Internet Explorer 6 이상을 사용해야 합니다.

#### B. 권장사항

- 예제는 PHP(또는 Python)와 자바스크립트로 작성되었습니다. 따라서 PHP(또는 Python)과 자바스크립트로 작성된 코드를 이해할 수 있어야 합니다.
- 네이버 지도 API에 대한 사용자 문서와 지역검색API/블로그검색API에 대한 사용자 문서를 읽으면 예제를 이해하는데 도움이 됩니다. 사용자 문서는 네이버 OpenAPI 사이트(<http://openapi.naver.com/>)에서 찾아볼 수 있습니다.

#### C. 개발환경 구성

개발언어로 Python을 사용할 경우 다음 사항을 설정해야 합니다. Python은 아파치 웹 서버의 Python모듈(Mod Python)을 이용하여 CGI를 구성하므로, 아파치 웹 서버에서 다음 사항을 설정합니다.

```
LoadModule python_module modules/mod_python.so

<IfModule python_module>
  AddHandler mod_python .psp .psp_
  PythonHandler mod_python.psp | .psp .psp_
  PythonHandler mod_python.cgihandler | .py
</IfModule>
```

### 구현하기

네이버 지역/블로그 검색 결과를 지도에 표시하는 애플리케이션을 만들려면 다음과 같은 기능을 구현해야 합니다. 본 예제 프로그램은 세 개의 소스 파일로 구성됩니다. 프로그램의 중심이 되는 mapsearch 파일과 블로그 검색 결과를 전달해 줄 blogsearch 파일, XML 문서를 파싱해 줄 xml2obj.js 파일로 구성됩니다. 개발언어에 따른 소스 파일 명칭은 다음과 같습니다.

- PHP: mapsearch.php, blogsearch.php, xml2obj.js
- Python: mapsearch.py, blogsearch.py, xml2obj.js

**참고** 이 예제는 PHP 예제 코드를 기준으로 설명하고 있습니다. Python 예제 코드는 이 문서의 “2.5.5 전체 소스 코드” 절을 참조하시기 바랍니다.



## STEP 1\_ 지도 띄우기

네이버 지도와 지역/블로그 검색을 위한 검색창을 출력하도록 화면을 구현합니다.

### ● mapsearch.php 파일

```
<!DOCTYPE HTML SYSTEM>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta http-equiv="Cache-Control" content="No-Cache">
    <meta http-equiv="Pragma" content="No-Cache">
    <title>지도/블로그 검색예제 </title>
  </head>
  <body style="margin: 0px 0px 0px 0px">
    <script type="text/javascript" src="http://maps.naver.com/js/naverMap.naver?key=지도API 키"> ] 1
    </script>
    <div id='mapContainer' style='width: 960px; height: 550px'></div> ] 2
    <script type="text/javascript">
      var mapObj = new NMap(document.getElementById('mapContainer'), 960, 550); ] 3
      mapObj.setBound(319198, 531730, 323198, 527730); ] 4
      mapObj.zoomOut();
      var zoom = new NZoomControl();
      zoom.setAlign("right");
      zoom.setValign("bottom");
      mapObj.addControl(zoom);
    </script>
    <div id='searchform'>
      <form name="mapsearch" method="get" action=<?=$_SERVER['PHP_SELF']?>>
        <input type="text" name="query">
        <input type="submit" value="Map Search">
      </form>
    </div>
  </body>
</html>
```

- ① 지도API를 사용하려면 네이버에서 제공하는 자바스크립트를 포함해야 합니다. 이때 key 파라미터로 지도 API 키를 입력해야 합니다.
- ② 지도를 출력할 컨테이너를 설정합니다. 위 예제에서는 div 엘리먼트를 사용했습니다.
- ③ NMap 클래스로 지도 객체를 생성합니다. 지도 객체를 생성할 때는 지도를 출력할 컨테이너 정보를 파라미터로 전달해야 합니다.
- ④ NMap.setBound() 메소드를 사용하여 컨테이너에 표시할 지도 영역을 설정합니다. 위 예제에서는 특정 영역을 기준으로 지도를 표시하기 위하여 NMap.setBound() 메소드를 사용했습니다. 만약 특정한 점을 중심으로 지도를 표시하려

면 *NMap.setCenter()* 또는 *NMap.setCenterAndZoom()* 메소드를 사용하면 편리합니다.

```
mapObj.setCenter(new NPoint(321198, 529730));  
mapObj.setCenterAndZoom(new NPoint(321198, 529730), 3);
```

⑤ ① ~ ④ 절차를 수행하면 기본 지도가 출력되는 것을 확인할 수 있습니다.

⑥ 지도를 손쉽게 확대 또는 축소하기 위해서는 줌 컨트롤 객체를 추가합니다. *NZoomControl* 클래스로 줌 컨트롤 객체를 생성한 후에, *NZoomControl.setAlignment()*와 *NZoomControl.setVertical()* 메소드를 사용하여 줌 컨트롤 객체의 표시 위치를 설정합니다. 그런 다음 *NMap.addControl()* 메소드로 줌 컨트롤 객체를 지도에 추가합니다. 이렇게 하면 축척 슬라이드 바가 지도 화면에 표시됩니다.

질의를 받을 수 있는 검색창을 생성합니다.

**참고** ● 지도가 출력되는 영역의 크기를 변경하려면 *div* 엘리먼트의 사이즈(style 속성값)을 변경하고, *NMap* 객체를 생성할 때 넘겨주는 세 번째/네 번째 파라미터를 변경합니다. 만약 이때 크기 정보를 전달하지 않으면 디폴트 값을 컨테이너의 너비와 높이로 사용합니다.

## 실행 화면

여기까지의 예제 코드를 실행하면 다음과 같이 화면에 지도가 출력됩니다

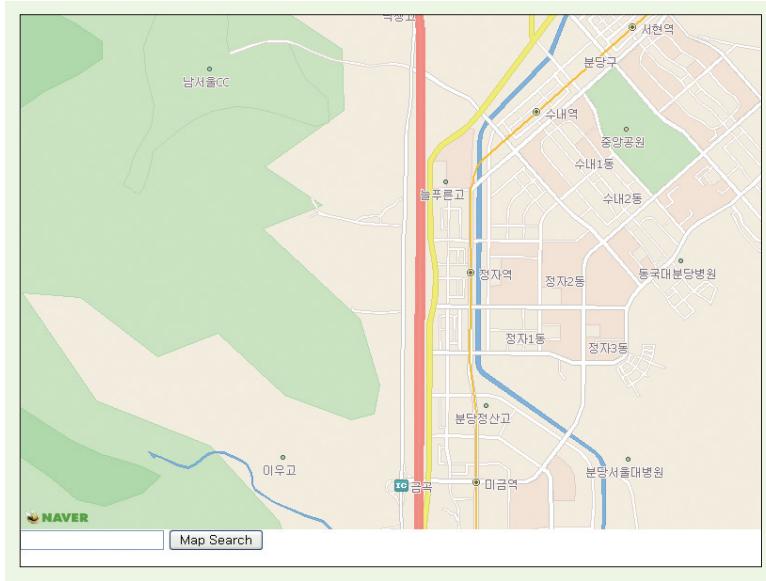


그림 2.6 지도 출력 화면 (1)

## STEP 2\_ 지역 정보를 검색하여 지도에 표시하기

검색창에 입력된 질의문으로 지역검색 API를 호출한 후, 그 결과를 지도 위에 반영하는 기능을 구현합니다. 지도 위에 마커 아이콘을 나타내고 마커 위에 마우스를 대면 정보창이 나타나도록 구현합니다.



## STEP 2-1\_ 지역검색 결과 파싱하기

먼저 질의문을 검색하여 결과를 파싱하는 과정입니다.

### ● mapsearch.php 파일

```
<?
define('MAPKEY', '지도 API 키');
define('NAVERKEY', '네이버 API 키');
$minx = 999999;
$maxx = 0;
$miny = 999999;
$maxy = 0;

if (isset($_GET["query"])) {
    $query = trim($_GET["query"]);
    if (strlen($query) > 0) {
        $encodedquery = urlencode($query);
        $url = "http://openapi.naver.com/search?query=$encodedquery&target=local&sort=vote&key=" ①
            .NAVERKEY;
        $result = simplexml_load_file($url); ②
        $list = array();
        $result = $result->channel;
        foreach ($result->item as $item) {
            $title = $item->title;
            $link = $item->link;
            $desc = $item->description;
            $tel = $item->telephone;
            $addr = $item->address;
            $mapx = intval($item->mapx);
            $mapy = intval($item->mapy);
            $minx = min($minx, $mapx);
            $maxx = max($maxx, $mapx);
            $miny = min($miny, $mapy);
            $maxy = max($maxy, $mapy);
            $tmparr = array($title, $link, $desc, $tel, $addr, $mapx, $mapy);
            array_push($list, $tmparr);
        }
    }
}
?>
```

- ① 검색창에서 질의문을 받으면 \$url을 만들어서 지역검색API를 호출합니다. 검색 결과는 추천수를 기준으로 정렬하도록 옵션을 설정하였습니다. (sort 파라미터에 vote로 입력하였습니다)

**참고** ● 위 예제에서는 utf-8 문자 셋을 사용하도록 설정되어 있기 때문에 질의문에 대해서 처리하지 않았습니다. 그러나 문자 셋이 utf-8이 아닌 경우에는 질의문을 utf-8로 변환해야 합니다. PHP는 iconv() 함수를, Python은 encode() 함수를 사용하여 변환합니다. 이 경우 변환은 \$url을 만들기 전에 수행해야 합니다.

- ② `simplexml_load_file()` 함수를 사용해서 XML 문서를 파싱합니다. `simplexml_load_file()` 함수는 지역검색 결과로 얻어온 XML 문서를 객체(SimpleXMLElement)로 만들어 줍니다. 이 객체를 사용하면 손쉽게 XML 문서에 접근할 수 있습니다.

**참고** ● Python 예제 코드에서는 XML 파싱을 위해 `getText()` 함수와 `getItem()` 함수, `getItemList()` 함수를 구현했습니다. `getText()` 함수는 노드의 텍스트를 얻기 위해 사용하며, `getItem()` 함수는 `getText()`로 얻은 텍스트를 dictionary 타입으로 구성하기 위해 사용합니다. `getItemList()` 함수는 DOM으로 파싱한 객체에서 원하는 엘리먼트만 얻어내기 위해 사용합니다. Python 예제 코드는 이 문서의 “2.5.5 전체 소스 코드”절을 참조하시기 바랍니다.

**참고** ● 네이버 OpenAPI의 검색 결과로 반환되는 XML 문서는 모두 RSS(Really Simple Syndication) 형식으로 되어 있습니다. RSS 문서에 대한 자세한 설명은 이 문서의 “부록. RSS 이해하기”를 참조하시기 바랍니다.

③ 각 item 엘리먼트를 확인하여 검색 결과를 배열에 저장합니다.

④ 좌표를 이용하여 지도에 표시할 영역을 결정합니다.

## STEP 2-2\_마커 아이콘과 정보창 표시하기

마커 아이콘을 지도 위에 표시하고, 마커 위에 마우스를 놓으면 정보창이 나타나도록 구현합니다.

### ● mapsearch.php 파일

```
<script type="text/javascript">
var mapObj = new NMap(document.getElementById('mapContainer'), 960, 550);
mapObj.setBound(<?=$minx?, <?=$maxy?, <?=$maxx?, <?=$miny?>);

var zoom = new NZoomControl();
zoom.setAlign("right");
zoom.setVAlign("bottom");
mapObj.addControl(zoom);
var infowin = new NIInfoWindow();
mapObj.addOverlay(infowin);
```

1

```
mapObj.addOverlay(infowin);
```

2

---

```
function setInfowin(title, tel, pos) {
    var body = ' \
<div style="width:220px; border:dotted 1px #000000; background:#ffffff; padding:5px 5px 5px 5px;"> \
<div style="font-size:10pt; font-family:굴림, tahoma; font-weight:bold; float:left;">' + title + '</div> \
<div style="font-size:8pt; font-family:굴림, tahoma;">' + tel + '</div> \
<div style="margin:10px 3px 3px 80px;"><a href="javascript:infowin.hideWindow()"> \
<img src=http://static.naver.com/n/cmn/btn_close.gif border=0 width=48 height=17></a></div> \
    infowin.set(pos, body);
    infowin.showWindow();
}
```

3

---

```
function addMarker(pos, count, title, tel) {
    var iconUrl = 'http://sstatic.naver.com/search/local/icon3/icos_L' +
        String.fromCharCode(64 + count) + '.gif';
    var marker = new NMark(pos, new NIIcon(iconUrl, new NSize(16, 15)));
```

4

```
NEvent.addListener(marker, "mouseover",
    function(pos) {
        setInfoWin(title, tel, pos);
    }
);
mapObj.addOverlay(marker);
}

function setMarkers() {
<?php
for ($i = 0; $i < count($list); $i++) {
    $cnt = $i + 1;
    $title = $list[$i][0];
    $tel = $list[$i][3];
    $mapx = $list[$i][5];
    $mapy = $list[$i][6];
    echo 'addMarker(new NPoint($mapx, $mapy), $cnt, '$title', '$tel');\n';
}
?>
}
setMarkers();
</script>
```

- ① *NMap* 클래스로 지도 객체를 만들고 *NMap.setBound()* 메소드를 호출하여 화면에 보여줄 영역을 정해줍니다. 그리고 측척 슬라이드 바를 지도에 추가합니다.
- ② *NInfoWin* 클래스로 정보창 객체를 만들고 지도에 등록합니다. 이 정보창에는 가게의 이름과 전화번호가 들어갑니다.
- ③ 정보창에 이름과 전화번호를 넣기 위해 *setInfoWin()* 함수를 작성합니다. 이 함수는 가게 이름과 전화번호, 정보창의 위치를 받아서 *infoWin.set()* 메소드로 정보창에 내용을 넣고 위치를 설정해 줍니다. 그리고 *NInfoWindow.showWindow()* 메소드를 호출해서 설정한 내용대로 화면에 출력합니다.
- ④ 지도상에 마커를 추가하는 *addMarker()* 함수를 작성합니다. 마커 객체는 *NMark* 클래스의 인스턴스이고, 위치와 아이콘을 지정해야 합니다. 아이콘 객체는 *NIcon* 클래스로 만들 수 있습니다. 마커 생성 후에 *NEvent.addListener()* 메소드로 마커에 이벤트 리스너를 등록합니다. 이 이벤트 리스너는 마커 위에 마우스 커서가 위치할 때, 위에서 정의한 *setInfoWin()* 함수를 호출해 정보창을 보여줍니다. 마커를 만들었다면 *NMap.addOverlay()* 메소드로 지도에 등록합니다.
- ⑤ 이제 *setMarkers()* 함수를 작성합니다. 이 함수는 *addMarkers()* 함수를 이용해서 검색된 모든 가게들에 대한 마커를 만들어서 지도에 표시합니다.
- ⑥ 마지막으로 *setMarkers()* 함수를 호출합니다.

## 실행 화면

여기까지 구현하면 다음과 같이 지도 화면에 마커와 정보창이 나타납니다.



그림 2.7 지도 출력 화면 (2)

## STEP 3\_ 블로그 검색결과 얻어오기/출력하기

블로그 검색결과를 얻어와서 정보창에 출력하는 기능을 구현합니다.

### ● blogsearch.php 파일

```
<?php
define('NAVERKEY', '네이버 API 키');

if (isset($_GET["query"])) {
    $query = trim($_GET["query"]);
    if (strlen($query) > 0) {
        $encodedquery = urlencode($query);
        $url = "http://openapi.naver.com/search?query=$encodedquery&target=blog&key=".NAVERKEY;
        $data = file_get_contents($url);
        echo $data;
    }
}
?>
```

### ● mapsearch.php 파일

```
<script type="text/JavaScript" src="xml2obj.js"></script>
<script type="text/javascript">
function processBlog(xml) {
    var obj = xml2obj(xml);
    var result = "";
    var items = obj.channel.item || [];
    for(var i = 0; i < items.length; i++) {
        var title = items[i].title;
        var link = items[i].link;
        if (title.length > 15) {
            title = title.substring(0, 15) + "...";
        }
    }
}
```

```

        }
        var con = "<li><a href=\"" + link + "\" target=_blank>" + title + "</a><br>";
        result += con;
    }
    return result;
}

function setInfowin(xml, title, tel, pos) {
    var body = ' \
<div style="width:220px; border:dotted 1px #000000; background:#ffffff; padding:5px 5px 5px 5px;"> \
<div style="font-size:10pt;font-family:굴림, tahoma;font-weight:bold;float:left;">' + title + '</div> \
<div style="font-size:8pt;font-family:굴림, tahoma;">' + tel + '</div> \
<div style="width:20px;font-size:8pt;padding:5px 0px 0px 0px;">' + processBlog(xml) + '</div> \
<div style="margin:10px 3px 3px 80px;"><a href="javascript:infowin.hideWindow()"> \
<img src=http://static.naver.com/n/cmn/btn_close.gif border=0 width=48 height=17></a></div> \
';
    infowin.set(pos, body);
    infowin.showWindow();
}

function addMarker(pos, count, title, tel) {
    var iconUrl = 'http://sstatic.naver.com/search/local/icon3/icos_L' +
        String.fromCharCode(64 + count) + '.gif';
    var marker = new NMark(pos, new NIIcon(iconUrl, new NSize(16, 15)));
    NEvent.addListener(marker, "mouseover",
        function(pos) {
            var url = "blogsearch.php?query=" + encodeURIComponent(title);
            var xmlhttp = new XMLHttpRequest();
            xmlhttp.setType("text");
            xmlhttp.load(url, setInfowin, title, tel, pos);
        }
    );
    mapObj.addOverlay(marker);
}

setMarkers();
</script>

```

- ① blogsearch.php를 작성합니다. blogsearch.php는 주어진 가게 이름으로 블로그 검색을 수행하고 그 결과를 출력합니다.
- ② mapsearch.php 파일에서 processBlog() 함수를 작성합니다. 이 함수는 블로그 검색 결과가 담긴 XML 문서에서 각 포스트의 제목과 링크를 얻고, 얻어낸 정보를 정보창에 출력할 수 있도록 HTML 형식으로 구성하여 반환합니다. 이 함수에서 xml2obj()라는 함수를 사용하고 있는데 이것은 XML 문서를 자바스크립트 객체로 바꿔주는 역할을 합니다. 이 함수를 불러오기 위해 xml2obj.js파일을 사용하였습니다. xml2obj.js 파일의 소스 코드는 이 문서의 Step4에 있으므로, 복사해서 사용하셔도 됩니다. processBlog() 함수는 addMarker() 함수 뒤에 작성하는 것이 적당합니다.
- ③ Step2에서 작성한 setInfowin() 함수를 수정합니다. 블로그 검색 결과를 받아오기 위한 매개변수 xml을 추가합니다. 블로그 검색 결과는 XML 포맷으로 되어 있으므로 processBlog() 함수를 호출하여 HTML 형식으로 변환하고 정보창에 추가합니다.
- ④ Step2에서 작성한 addMarker() 함수에서 마커에 이벤트 리스너를 등록하는 부분을 수정합니다. blogsearch.php를 이용해서 블로그 검색 결과를 얻어오는 URL(위 예제의 url변수)을 작성합니다. XML문서를 읽어오는 XMLHttpRequest 클래스를 사용해서 블로그 검색 결과 XML 문서를 읽어오는 객체를 생성합니다. XML 문서를 읽은 뒤 그 문서를 setInfowin()

함수에 매개변수로 넘겨줍니다. 이때 `NXml/http.loadhttp()` 메소드의 세 번째 파라미터로 가게 이름을, 네 번째 파라미터로 전화번호를, 다섯 번째 파라미터로 마커의 위치를 전달합니다.



## STEP 4\_ xml2obj.js 코드

이 예제에서는 XML 문서를 빠르게 파싱하기 위하여 `xml2obj()` 함수를 사용하고 있습니다. 이 함수는 자바스크립트에서 제공하지 않기 때문에 여러분이 `xml2obj()` 함수를 사용할 수 있도록 다음의 소스 코드를 제공합니다. 다음 소스 코드를 `xml2obj.js`라는 파일 이름으로 `mapsearch.php` 파일(또는 `mapsearch.py` 파일)과 같은 디렉터리에 저장하기 바랍니다.

### ● `xml2obj.js` 파일

```
function xml2obj(xml)
{
    var obj = {}, que = [], depth = 0;

    // attribute를 해석하기 위한 함수
    var parse_attr = function(oobj, str) {
        str.replace(/([^\s]+)\s*=\s*(["]*)"/g,
            function(a0,a1,a2) {
                oobj[a1] = a2;
            }
        );
    }

    // 주석, XML선언, 테그 사이 공백 등의 의미 없는 코드를 삭제
    xml = xml.replace(/<(\?|\!-)[^>]*>/g,'').replace(/>\s+</g, '><');

    // 하위 노드가 없는 태그는 하나의 닫힌 태그로 수정
    xml = xml.replace(/<(![\^>]+)(\s[^>]*)?></\1>/g, '<$1$2 />').replace(/^s+|\s+$|/g, '');

    // 함수 객체를 정규 표현식 처리의 인자로 줘서 iterator로 사용
    xml = xml.replace(/<\/?([^\!][^>]*)(\s[^>]*)?>(<\/\$1>|<!\[CDATA\[(:(.|\s)*?)\]\]>|^<>*)/g,
        function(a0,a1,a2,a3) {
            // IE에서 일치하는 내용이 없으면 undefined로 전달되므로
            // 빈 문자열로 변경해 다른 브라우저와의 호환성을 맞춤
            if (typeof a1 == 'undefined') a1 = '';
            if (typeof a2 == 'undefined') a2 = '';
            if (typeof a3 == 'undefined') a3 = '';

            if (a0.substr(1,1) == '/') { // 현재 태그가 닫는 태그라면,
                // 깊이를 1만큼 감소
                depth--;
            }
            else if (que.length == 0) { // 객체 큐에 객체가 없다면,
                que[depth] = obj; // 초기의 객체를 큐에 넣고
                parse_attr(obj, a2); // attribute를 해석
            }
            else {
                var k = a1, o = {}, is_closed = false;
                is_closed = (a2.substr(-1,1) == '/');
                if (a3.length > 0 || is_closed) { // 텍스트 노드가 있다면
                    o = a3; // 추가할 객체는 문자열 객체
                }
                else {
                    que[depth].push(o);
                }
            }
        }
    );
}
```

```

        // CDATA라면 전달받은 그대로 리턴하고
        // 그렇지 않다면 decode 해서 리턴
        if (o.substr(0,9) == '<![CDATA[' && o.substr(-3,3) == ']]>') o = o.substring(0, o.length-3);
        else o = o.replace(/</g, '<').replace(/>/g, '>').replace(/&/g, '&');
    }

    // 객체를 할당하기 전에 태그 이름이 이미 존재하는지 살펴보고
    // 이전에 존재하는 태그라면, 배열로 만든다. 이미 배열이라면 현재의 객체를 배열에 추가
    if (typeof que[depth][k] == 'undefined') {
        que[depth][k] = o;
    }
    else {
        var v = que[depth][k];
        if (que[depth][k].constructor != Array) que[depth][k] = [v];
        que[depth][k].push(o);
    }

    // attribute를 해석
    parse_attr(o, a2);

    if (!is_closed) que[++depth] = o;
}

return '';
};

return obj;
}

```

### 예제 코드 실행 결과

위 예제 코드를 실행하면 다음과 같이 지도에 지역검색, 블로그 검색 결과가 표시됩니다.

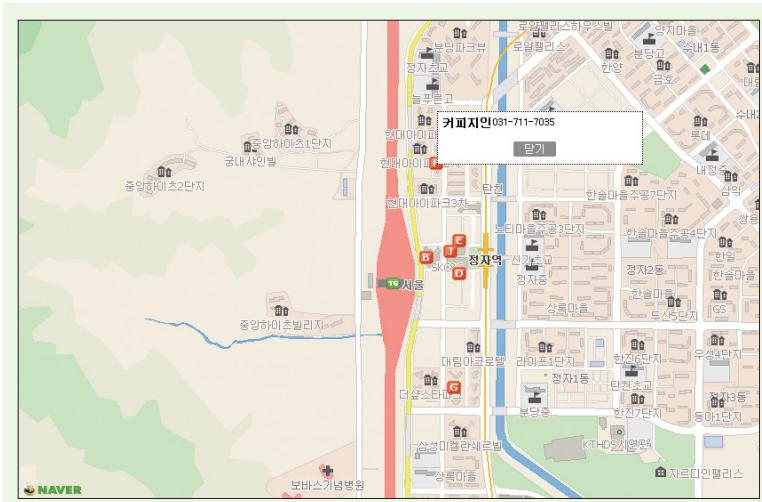


그림 2.8 지도/블로그 검색 예제 실행 결과 화면

## php용 전체 소스코드

### ● blogsearch.php 파일

```
<?php
define('NAVERKEY', '네이버 API 키');

if (isset($_GET["query"])) {
    $query = trim($_GET["query"]);
    if(strlen($query) > 0) {
        $encodedquery = urlencode($query);
        $url = "http://openapi.naver.com/search?query=$encodedquery&target=blog&key=".NAVERKEY;
        $data = file_get_contents($url);
        echo $data;
    }
}
?>
```

### ● mapsearch.php 파일

```
<?php
define('MAPKEY', '지도 API 키');
define('NAVERKEY', '네이버 API 키');

$minx = 999999;
$maxx = 0;
$miny = 999999;
$maxy = 0;
$query = "";

if (isset($_GET["query"])) {
    $query = trim($_GET["query"]);
    if (strlen($query) > 0) {
        $encodedquery = urlencode($query);
        $url =
            "http://openapi.naver.com/search?query=$encodedquery&target=local&sort=vote&key="
            .NAVERKEY;
        $result = simplexml_load_file($url);
        $list = array();
        $result = $result->channel;
        foreach ($result->item as $item) {
            $title = $item->title;
            $link = $item->link;
            $desc = $item->description;
            $tel = $item->telephone;
            $addr = $item->address;
            $mapx = intval($item->mapx);
            $mapy = intval($item->mapy);
            // 지도가 표시될 영역을 결정합니다.
            $minx = min($minx, $mapx);
            $maxx = max($maxx, $mapx);
            $miny = min($miny, $mapy);
            $maxy = max($maxy, $mapy);
        }
    }
}
```

```
$miny = min($miny, $mapy);
$maxy = max($maxy, $mapy);

$tmparr = array($title, $link, $desc, $tel, $addr, $mapx, $mapy);
array_push($list, $tmparr);
}

}

?>

<!DOCTYPE HTML SYSTEM>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv="Cache-Control" content="No-Cache">
<meta http-equiv="Pragma" content="No-Cache">
<title>지도/블로그 검색 예제</title>
</head>
<body style="margin:0px 0px 0px 0px">
<script type="text/javascript" src="http://maps.naver.com/js/naverMap.
naver?key=<?=MAPKEY?>">
</script>
<div id='mapContainer' style='width:960px; height:550px'></div>
<script type="text/javascript" src="xml2obj.js"></script>
<script type="text/javascript">
// 기본지도 생성
var map0bj = new NMap(document.getElementById('mapContainer'), 960, 550);
map0bj.setBound(<?=$minx?>, <?=$maxy?>, <?=$maxx?>, <?=$miny?>);

// 확대, 축소를 위한 컨트롤을 생성합니다.
var zoom = new NZoomControl();
zoom.setAlign("right");
zoom.setValign("bottom");
map0bj.addControl(zoom);

// 정보 출력을 위한 NInfoWindow 객체를 생성하여 등록합니다.
var infowin = new NInfoWindow();
map0bj.addOverlay(infowin);

// 검색 결과를 마커를 사용하여 지도에 표시합니다.
function addMarker(pos, count, title, tel) {
    var iconUrl =
        'http://sstatic.naver.com/search/local/icon3/icos_L' +
        String.fromCharCode(64 + count) + '.gif';
    // 마커 생성
    var marker = new NMark(pos, new NIIcon(iconUrl, new NSize(16, 15)));
    // marker 객체에 마우스가 오버했을 때, 실행할 함수를 등록합니다.
    NEvent.addListener(marker, "mouseover",
        function(pos) {
            var url = "blogsearch.php?query=" + encodeURIComponent(title);
            var xmlhttp = new XMLHttpRequest();
            xmlhttp.setType(0);
```

```
        xmlhttp.loadhttp(url, setInfowin, title, tel, pos);
    }
);

// 지도에 마커를 등록합니다.
mapObj.addOverlay(marker);
}

function processBlog(xml) {
    var target = document.getElementById("tmpl");
    var obj = xml2obj(xml);
    var result = "";

    var items = obj.channel.item || [];
    for (var i = 0; i < items.length; i++) {
        var title = items[i].title;
        var link = items[i].link;
        if (title.length > 15) {
            title = title.substring(0, 15) + "...";
        }
        var con =
            "<li><a href=\"" + link + "\" target=_blank>" + title + "</a><br>";
        result += con;
    }

    return result;
}

function setMarkers() {
<?php
for ($i = 0; $i < count($list); $i++) {
    $cnt = $i + 1;
    $title = $list[$i][0];
    $tel = $list[$i][3];
    $mapx = $list[$i][5];
    $mapy = $list[$i][6];
    echo "addMarker(new NPoint($mapx, $mapy), $cnt, '$title', '$tel');\n";
}
?>
}

function setInfowin(xml, title, tel, pos) {
    var body = ' \
<div style="width:220px; border:dotted 1px #000000; background:#ffffff; padding:5px 5px 5px;"> \
<div style="font-size:10pt;font-family:굴림, tahoma; font-weight:bold; float:left;">' + title + '</div> \
<div style="font-size:8pt;font-family:굴림, tahoma;">' + tel + '</div> \
<div style="width:200px; font-size:8pt; padding:5px 0px 0px 0px;">' + processBlog(xml) + '</div> \
<div style="margin:10px 3px 3px 80px;"><a href="javascript:infowin.hideWindow()"> \
<img src=http://static.naver.com/n/cmn/btn_close.gif border=0 width=48 height=17></a></div> \
';
```

```
        infowin.set(pos, body);
        infowin.showWindow();
    }

    // 검색된 결과를 지도에 표시합니다.
    setMarkers();
</script>

<div id="searchform">
    <form name="mapsearch" method="get" action=<?=$_SERVER['PHP_SELF']?>">
        <input type="text" name="query">
        <input type="submit" value="Map Search">
    </form>
</div>
</body>
</html>
```

## Python용 전체 소스코드

### ● blogsearch.py 파일

```
 #-*- coding: utf-8 -*-
import os, cgi, urllib

NAVERKEY = "네이버 API 키"
_GET = cgi.parse_qs(os.environ['QUERY_STRING'])

query = "";
if _GET.has_key("query"):
    query = _GET['query'][0].strip()

print "Content-Type: application/xml\n"
if len(query) > 0:
    params = {"query": query,
               "target": "blog",
               "key": NAVERKEY
             }
    queryString = urllib.urlencode(params)
    data = urllib.urlopen("http://openapi.naver.com/search?" + queryString).read()

    print data
```

### ● mapsearch.py 파일

```
 #-*- coding: utf-8 -*-
import os, cgi, urllib
import xml.dom.minidom

# 네이버 검색 및 지도OpenAPI 키
NAVERKEY = "네이버 API 키"
```

```
MAPKEY = "지도 API 키"

# 언어환경
LANG = "utf-8"

# XML Parsing을 위한 함수
def getText(nodelist):
    rc = ""
    for node in nodelist:
        if node.nodeType == node.TEXT_NODE:
            rc = rc + node.data

    return rc.encode(LANG)

def getItem(nodelist):
    items = {}
    for node in nodelist:
        items[node.tagName] = getText(node.childNodes)
    return items

def getItemList(data):
    items = xml.dom.minidom.parseString(data).getElementsByTagName("channel")[0]. \
        getElementsByTagName("item")
    itemlist = {}
    for i, item in enumerate(items):
        itemlist[i] = getItem(item.childNodes)
    return itemlist

# 지도에 사용 될 마커를 생성하는 함수
def getMarker(node, cnt):
    node['cnt'] = cnt
    return "addMarker(new NPoint(%(mapx)s, %(mapy)s), %(cnt)d, '%(title)s', '%(telephone)s');\n" \
        % node

def getMarkerList(nodelist):
    rc = ""
    for i in range(0, len(nodelist)):
        rc += getMarker(nodelist[i], i+1)
    return rc

content = {"minx": 999999,
           "maxx": 0,
           "miny": 999999,
           "maxy": 0,
           "LANG": LANG,
           "MAPKEY": MAPKEY,
           "QUERY": "",
           "MARKER": "",
           "FILE": os.path.basename(__file__)
         }

_GET = cgi.parse_qs(os.environ['QUERY_STRING'])
if _GET.has_key("query"):
```

```
content['QUERY'] = _GET['query'][0].strip()

# 검색어가 존재 할 경우
if len(content['QUERY']) > 0:
    params = {"query": content['QUERY'],
               "target": "local",
               "sort": "vote",
               "key": NAVERKEY
              }
    queryString = urllib.urlencode(params)

# 검색데이터를 가져옵니다.
data = urllib.urlopen("http://openapi.naver.com/search?" + queryString).read()

# XML 파일의 데이터를 읽습니다.
itemlist = getItemList(data)

# 지도가 표시될 영역을 결정합니다.
for i in range(0,len(itemlist)):
    content["minx"] = min(content["minx"], int(itemlist[i]["mapx"]))
    content["maxx"] = max(content["maxx"], int(itemlist[i]["mapx"]))
    content["miny"] = min(content["miny"], int(itemlist[i]["mapy"]))
    content["maxy"] = max(content["maxy"], int(itemlist[i]["mapy"]))

content['MARKER'] = getMarkerList(itemlist)

mapSearch = """<!DOCTYPE HTML SYSTEM>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=%(LANG)s">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="pragma" content="no-cache">
    <title>지도/블로그 검색 예제</title>
  </head>
  <body>
    <script type="text/javascript" src="xml2obj.js"></script>
    <script type="text/javascript" src="http://maps.naver.com/js/naverMap.naver?key=%(MAPKEY)s">
    </script>
    <div id="mapContainer" style="width:960px;height:550px;"></div>
    <script type="text/javascript">
      // 기본 지도 생성
      var mapObj = new NMap(document.getElementById('mapContainer'),960,550);
      mapObj.setBound(%(minx)s, %(maxy)s, %(maxx)s, %(miny)s);
      mapObj.zoomOut();

      // 확대, 축소를 위한 컨트롤을 생성합니다.
      var zoom = new NZoomControl();
      zoom.setAlign("right");
      zoom.setVAlign("bottom");
      mapObj.addControl(zoom);

      // 정보 출력을 위한 NInfoWindow 객체를 생성하여 등록합니다.
    </script>
  </body>
</html>"""
```

```
var infowin = new NInfoWindow();
mapObj.addOverlay(infowin);

// 검색 결과를 마커를 사용하여 지도에 표시합니다.
function addMarker(pos, count, title, tel) {
    var iconUrl =
        'http://sstatic.naver.com/search/local/icon3/icos_L' +
        String.fromCharCode(64 + count) + '.gif';
    // 마커 생성
    var marker = new NMark(pos, new NIcon(iconUrl, new NSize(16, 15)));
    // marker 객체에 마우스가 오버했을 때, 실행할 함수를 등록합니다.
    NEvent.addListener(marker, "mouseover",
        function(pos) {
            var url = "blogsearch.py?query=" + encodeURIComponent(title);
            var xmlhttp = new NXmlhttp();
            xmlhttp.setType(0);
            xmlhttp.loadhttp(url, setInfowin, title, tel, pos);
        }
    );
    // 지도에 마커를 등록합니다.
    mapObj.addOverlay(marker);
}

function processBlog(xml) {
    var target = document.getElementById("tmpl");
    var obj = xml2obj(xml);
    var result = "";

    var items = obj.channel.item || [];
    for(var i = 0; i < items.length; i++) {
        var title = items[i].title;
        var link = items[i].link;
        if (title.length > 15) {
            title = title.substring(0, 15) + "...";
        }
        var con =
            "<li><a href=\"\" + link + \"\" target=_blank>" + title + "</a><br>";
        result += con;
    }

    return result;
}

function setMarkers() {
    %(MARKER)s
}

function setInfowin(xml, title, tel, pos) {
    var body = ' \
<div style="width:220px; border:dotted 1px #000000; background:#ffffff; padding:5px 5px 5px;"> \
<div style="font-size:10pt; font-family:굴림, tahoma; font-weight:bold; float:left;">' +
```

```
title + '</div> \
<div style="font-size:8pt;font-family:굴림, tahoma;">' + tel + '</div> \
<div style="width:200px;font-size:8pt;padding:5px 0px 0px 0px;">' + processBlog(xml) + '</div> \
<div style="margin:10px 3px 3px 80px;"><a href="javascript:infowin.hideWindow()"> \
<img src=http://static.naver.com/n/cmn/btn_close.gif border=0 width=48 height=17></a></div> \
';

    infowin.set(pos, body);
    infowin.showWindow();
}

// 검색된 결과를 지도에 표시합니다.
setMarkers();
</script>
<div id="searchform">
<form name="mapsearch" method="get" action="%(FILE)s">
    <input type="text" name="query" value="%QUERY{s}">
    <input type="submit" value="Map Search">
</form>
</div>
</body>
</html>
"""
% content
print "Content-Type: text/html; charset=" + LANG + "\n"
print mapSearch
```

---

## Troubleshooting

예제 코드가 실행되지 않을 경우 다음 사항을 확인해 봅니다.

- 소스코드의 파일명을 예제대로 했는지 확인해 보시기 바랍니다. 파일 이름이 다르면 동작하지 않을 수 있습니다.
- 네이버 OpenAPI 키(검색 API 키, 지도 API 키)가 정확한지 확인해 보시기 바랍니다. 지도 API 키를 사용하신다면 키 발급 시 등록한 디렉터리가 맞는지 확인해 보시기 바랍니다.
- 소스 코드를 utf-8로 인코딩해서 저장하시기 바랍니다. euc-kr로 저장한 경우 자바스크립트 에러가 발생할 수 있습니다.
- 서버의 PHP 버전을 확인해 보시기 바랍니다. PHP 5 미만에서는 동작하지 않습니다.
- PHP 설정에서 short\_open\_tag = On 으로 되어있는지 확인해 보시기 바랍니다.
- 웹 브라우저가 자바스크립트를 실행할 수 있는지 확인해 보시기 바랍니다. 웹 브라우저가 자바스크립트를 지원하지 않거나 자바스크립트를 실행하지 않도록 설정되어 있을 수 있습니다. Firefox 1.5 이상, Microsoft Internet Explorer 6.0 이상을 사용할 것을 권장합니다.
- 위 항목을 모두 확인해도 문제가 해결되지 않는다면 네이버 OpenAPI FAQ (<http://openapi.naver.com/>)를 참고하시기 바랍니다.

# 부록



RSS 이해하기

RSS란?

RSS 리더 활용하기

RSS 문서의 구조



## RSS 이해하기

네이버 OpenAPI의 수행 결과는 RSS(Really Simple Syndication) 형식으로 반환되므로, OpenAPI수행 결과를 이용하기 위해서는 RSS 문서 형식을 이해해야 합니다. 다음 내용은 RSS에 대한 일반적인 설명입니다. 시작 전에 알아두도록 합시다.

### RSS란?

RSS란 뉴스나 블로그와 같이 자주 갱신되는 정보를 표현하기 위한 포맷입니다. RSS 문서에는 RSS문서를 제공하는 사이트에서 제공하는 콘텐츠 전체가 담겨 있거나 콘텐츠에 대한 요약이 담겨 있습니다.

예를 들어 네이버 카페에서 제공하는 RSS문서에는 그 카페에 올라온 최근 글의 목록이 담겨 있습니다.

다음 화면의 왼쪽 하단에 보이는 RSS 아이콘을 클릭하면 RSS 문서로 연결됩니다. 블로그라면 최근에 포스팅된 글의 목록을 RSS 문서로 제공할 것이며, 뉴스 사이트라면 최근 뉴스의 목록과 요약이 담긴 RSS 문서를 제공할 것입니다.

The screenshot shows the Naver Cafe RSS feed page. On the left, there's a sidebar with links like '자유게시판', '오픈API', and 'FAQ'. Below it is a search bar and a '검색' button. A large green arrow points from the 'RSS' icon in the sidebar to the right-hand panel, which displays the XML code of the RSS feed.

```

<?xml version="1.0" encoding="EUC-KR" ?>
- <rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
- <channel>
- <title><![CDATA[ 네이버 OpenAPI 풍식 카페 ]]>
</title>
<link>http://cafe.naver.com/openapi</link>
- <description><![CDATA[ 모두에게 열려있는 네이버 OpenAPI 풍식 카페입니다. ]]>
</description>
<language>ko</language>
- <item>
- <title><![CDATA[ 일본어 검색 서비스 내부 개편에 따른 일본어 검색 openapi 내부 변경 ]]>
</title>
<link>http://cafe.naver.com/openapi/1352</link>
<description><![CDATA[ 일본어 검색 서비스 내부 개편에 따른 일본어 검색 서비스 http://pdic.naver.com 이 개편되어 일본어 서비스 api 도 내부적으로 약간의 구조 변경이 있었습니다.외부에 보이는.. ]]>
</description>
<pubDate>Wed, 30 Apr 2008 12:05:00 +0900</pubDate>
- <item>
- <title><![CDATA[ 마커를 마우스로 끌어다가 다른곳으로 옮길수가있나요? ]]>
</title>
<link>http://cafe.naver.com/openapi/1351</link>
- <description><![CDATA[ 네이버맵에서 마커를 마우스로 끌어다가 다른곳으로 옮기고 그곳에 좌표값을 만원발고 싶은데.. 가능한가요?api를 접한지 이제갓 일주일?ㅎㅎ?,..해야될건.. ]]>
</description>
<pubDate>Wed, 30 Apr 2008 11:39:00 +0900</pubDate>
- <item>
- <title><![CDATA[ 네이버 map api 마우스 제어에 대책 ]]>
</title>

```

그림 2.9 RSS 링크

## RSS 리더 활용하기

RSS 문서를 직접 읽으려면 XML 엘리먼트를 이해해야 하므로 다소 불편합니다. RSS 리더를 활용하면 RSS 문서를 파싱해서 WYSIWYG(What You See Is What You Get) 문서 형태로 보여줍니다.

다음 그림은 Microsoft Outlook에 내장된 RSS리더를 활용해서 RSS 문서를 읽은 예제입니다. RSS 리더에 RSS 문서를 등록해 두면 RSS 문서는 항상 최신으로 갱신되므로, 사용자는 사이트를 일일이 방문하지 않고도 새로운 콘텐츠를 습득할 수 있습니다.

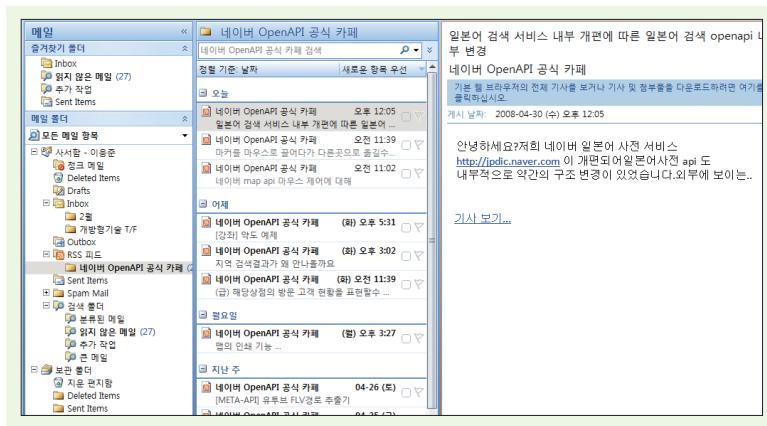


그림 2.10 RSS 리더

## RSS 문서의 구조

네이버 OpenAPI에서 RSS형식으로 반환해주는 수행결과를 해석하기 위해서는 RSS문서의 구조에 대해 알아야 할 필요가 있습니다. 실제 사례를 통해 알아보겠습니다. 다음은 네이버 OpenAPI로 검색 키워드를 'notebook'으로 설정하고 이미지 검색을 수행하여 그 결과를 얻어온 예입니다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
  <channel>
    <title>Naver Open API - image ::'notebook'</title>
    <link>http://search.naver.com</link>
    <description>Naver Search Result</description>
    <lastBuildDate>Tue, 29 Apr 2008 21:53:56 +0900</lastBuildDate>
    <total>869</total>
    <start>1</start>
    <display>1</display>
    <item>
      <title>The Notebook</title>
      <link>http://cafefiles.naver.net/data24/2007/5/20/176/00n-back-rda-taki320.jpg</link>
      <thumbnail>http://cafethumb2.naver.net/data24/2007/5/20/176/00n-back-rda-taki320.jpg?type=r2</thumbnail>
      <sizeheight>815</sizeheight>
      <sizewidth>550</sizewidth>
    </item>
  </channel>
</rss>
```

위 RSS 문서에서 산괄호로 둘러 쌓인 부분이 태그입니다. 태그는 태그 안에 담긴 내용이 무엇을 의미하는지 알려줍니다. 예를 들어 위 결과에서 <sizewidth>550</sizewidth>는 550이라는 숫자가 이미지의 너비를 뜻한다는 것을 나타냅니다. 이렇게 태그와 태그 안에 담긴 정보를 합쳐서 엘리먼트라고 부릅니다.

엘리먼트의 이름이 같다고 해서 그 의미가 어디서나 같은 것은 아닙니다. 예를 들어 위 RSS 문서에는 <title> 태그가 두 번 나오는데, 두 번의 쓰임이 다릅니다.

#### A. 첫 번째 <title>

```
<channel>
  <title>Naver Open API - image ::'notebook'</title>
...
</channel>
```

첫 번째 title은 channel 엘리먼트의 하위에 속해 있습니다. XML 문서에서는 이러한 관계를 부모/자식 관계라고 합니다. channel 엘리먼트는 title 엘리먼트의 부모가 되고, title 엘리먼트는 channel 엘리먼트의 자식이 됩니다. channel 엘리먼트는 검색 결과 전체를 포함하는 상위 노드가 되므로, channel/title 엘리먼트는 검색 결과 전체의 제목이 됩니다.

#### B. 두 번째 <title>

```
<item>
  <title>The Notebook</title>
...
</item>
```

두 번째 title은 item 엘리먼트의 자식 노드입니다. item 엘리먼트는 검색 결과 한 개를 나타내므로, item/title은 검색 결과의 제목이 됩니다.

네이버 OpenAPI가 생성해서 반환하는 결과에 들어있는 엘리먼트들에 대한 설명은 네이버 OpenAPI 공식 사이트 (<http://openapi.naver.com>)에서 확인할 수 있습니다. 또한 RSS에 대한 자세한 설명은 <http://cyber.law.harvard.edu/rss/rss.html>의 스펙을 참고하시기 바랍니다.



