FingerEyes-Xr for HTML5 Tutorials - 03

WFS 활용하기

2015년 3월 2일, 1차 배포

㈜지오서비스

WFS 활용하기

WFS(Web Feature Service)는 OGC의 표준으로써 클라이언트에서 요청한 피쳐(Feature)에 대한 셋(Set)을 서버가 조회하여 조회된 피쳐의 좌표값과 속성값을 XML이나 JSON, KML, ShapeFile 등과 같은 포맷으로 구성하여 클라이언트에게 서비스되는 방식입니다. 단순히 피쳐에 대한 조회 뿐만 아니라 편집도 가능하며 이러한 편집에 대한 안정적인 수행을 위한 트랜젝션(Transaction) 기능까지 제공합니다. 이러한 WFS에 대한 지원을 FingerEyes-Xr에서는 WFSLayer을 통해 활용 가능합니다.

이 글에서는 WFSLayer를 이용하여 WFS 형태로 서비스 되는 지도 데이터를 가져와 화면에 표시하는 내용에 대해 설명합니다. FingerEyes-Xr은 Flex 버전과 HTML5 버전이 존재하며 이 글은 HTML5에 대해 글입니다. FingerEyes-Xr for HTML5에 대한 소스 코드는 GitHub에서 다운로드 받을 수 있으며 URL은 https://github.com/FingerEyes-Xr/src 입니다. 아래의 화면은 FingerEyes-Xr for HTML5에 대한 GitHub 웹 화면입니다.

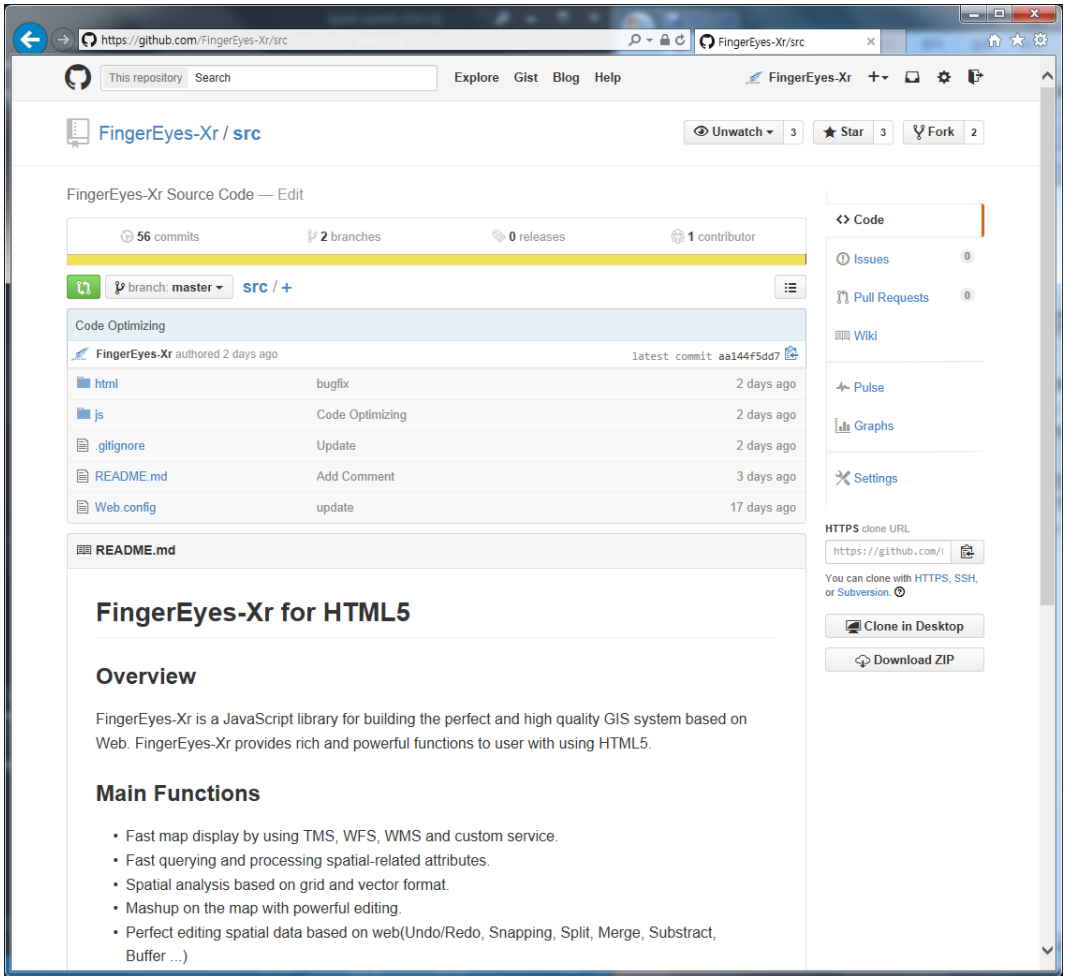

그림 1. FingerEyes-Xr for HTML에 대한 GitHub

필자는 이 글에 대한 예제 코드를 VisualStudio 2012를 이용하여 작성 하였습니다. 독자 여러분들이 어떤 툴을 사용하든 문제는 없겠으나 VisualStudio 2012에서 제공하는 JavaScript 디버깅과 기본적으로 제공하

는 웹서버(IIS)를 통해 웹 기반의 프로그래밍을 편리하고 효율적으로 진행할 수 있었기에 선택하였습니다. 참고로 웹 기반의 프로그래밍은 웹서버를 이용하여 URL 형태로 접근하지 않으면 올바르게 실행되지 않으므로 반드시 웹서버를 통해 URL 형태로 접근하시기 바랍니다.

WFSLayer를 FingerEyes-Xr에서 사용하기에 앞서 먼저 WFS 형태로 서비스할 공간 데이터에 대해 언급하겠습니다. 여기서는 GeoServer에서 서비스되는 WFS를 이용하겠습니다. GeoServer에서 공간 데이터를 관리하는 내용은 GeoServer 매뉴얼을 참고하시기 바랍니다. 여기서는 아래와 같은 레이어들 중 하나를 선택하여 FingerEyes-Xr에서 WFSLayer를 통해 표시해 보도록 하겠습니다.

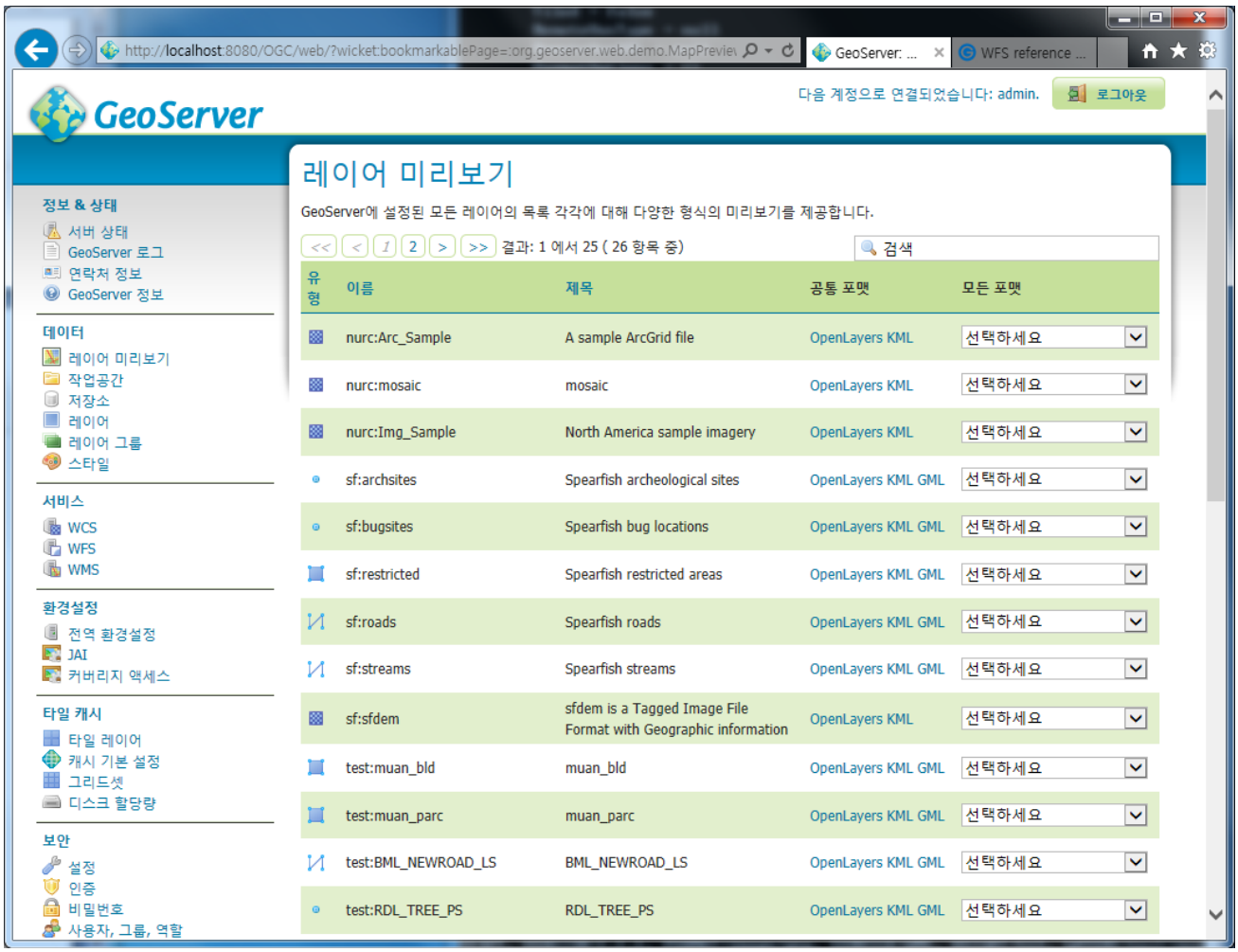

그림 2. GeoServer에서 서비스되는 레이어 구성

위의 레이어 목록 중 test:muan_parc 라는 이름의 레이어를 사용하겠습니다. 이 레이어는 지적 필지로써 폴리곤 타입입니다.

필자는 GeoServer를 Local에 실행해 두고 예제를 작성했습니다. 여러분들은 각자의 상황에 맞게 사용하시기 바랍니다.

이제 본격적으로 WFSLayer를 FingerEyes-Xr에서 사용해 보는 코드를 작성해 보겠습니다. 먼저 map.html

파일을 웹서버에서 접근할 수 있는 곳에 생성합니다. 필요하다면 map.html이 아닌 다른 파일명으로 생성하여도 진행에 문제는 없습니다. 다만, URL을 통해 웹브라우저에서 실행할 수 있는 경로에 생성한다는 것이 중요합니다. 일반 html을 생성하고 다음처럼 입력합니다.

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03    <title>FingerEyes-Xr : WFSLayer</title>
04 </head>
05 <body>
06
07 </body>
08 </html>
```

코드 1. 초기 map.html

위의 코드에서 3번 라인 밑에 다음 코드를 추가합니다.

```
01 <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
02
03 <script type="text/javascript">
04
05 </script>
```

코드 2. 기본 Script

1번 코드에서 CDN(Content Delivery Network)을 사용하여 FingerEyes-Xr의 JavaScript 라이브러리에 대한 소스 코드를 추가 하고 있습니다. 그리고 4번의 빈 공백 부분에 앞으로 추가할 JavaScript 코드가 입력될 것입니다.

UI를 구성하기 위해 <body> 부분을 다음처럼 변경합니다.

```
01 <body>
02    <div id="mainLayout">
03       <div id="mapDiv"></div>
04       <div id="title">
05          FingerEyes-Xr for HTML5 :
06          <font color="#349bd6">WFSLayer Example</font>
07       </div>
08    </div>
09 </body>
```

코드 3. 기본 UI 구성

특히 Id가 mapDiv인 DIV 요소에 맵(Map)이 표시될 것입니다. 일단 여기까지 작성하고 실행해 보면 다음과 같은 화면이 나타나는 것을 볼 수 있습니다.

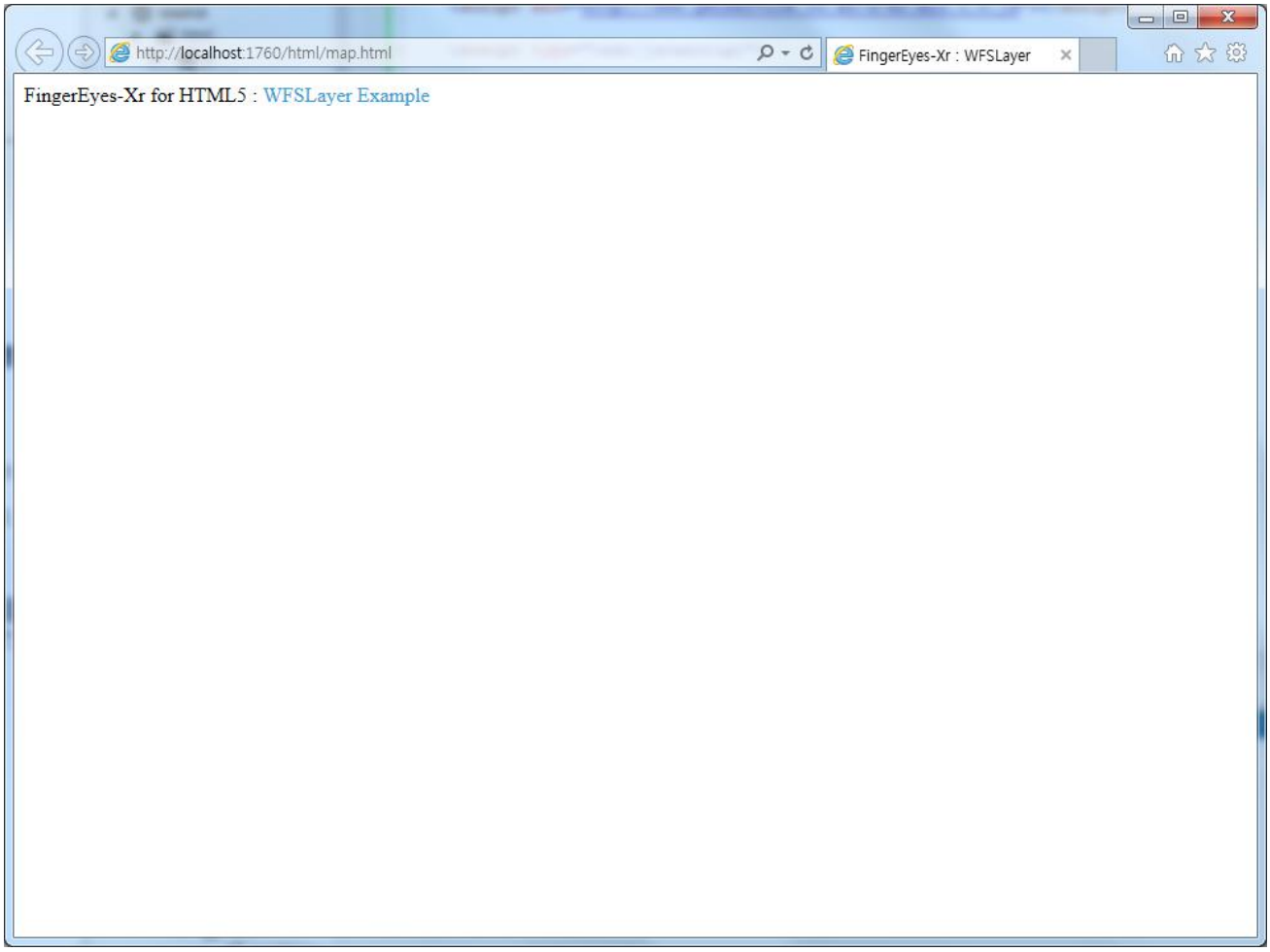

그림 3. 기본 UI 구성에 대한 실행 화면

WFS 레이어를 지도 화면에 표시하는 코드를 살펴보고 지도를 조작하는데 필요한 버튼 UI를 추가 하겠습니다.

이제 만든 페이지에 스타일을 적용해 보도록 하겠습니다. <head> 바로 밑에 아래의 스타일 코드를 추가합니다.

```
01  <style>
02  body
03  {
04      margin:0px;
05      padding:0px;
06  }
07
08  #mainLayout
09  {
10      position:relative;
11      width:100%;
12      height:100%;
13      border: none;
14  }
15
16  #mapDiv {
17      top:0px;
18      left:0px;
```

```
19      position:relative;
20      width:100%;
21      height:100%;
22      border: none;
23      overflow:auto;
24  }
25
26  #title {
27      top:12px;
28      left:12px;
29      padding: 12px;
30      position:absolute;
31      background:rgba(0,0,0,0.7);
32      border:none;
33      overflow:auto;
34      border-radius: 12px;
35      font-size: 24px;
36      color: #ffffff;
37      font-family: "Arial";
38  }
    </style>
```

코드 4. UI에 대한 스타일 적용

스타일이 적용된 페이지는 다음과 같습니다.

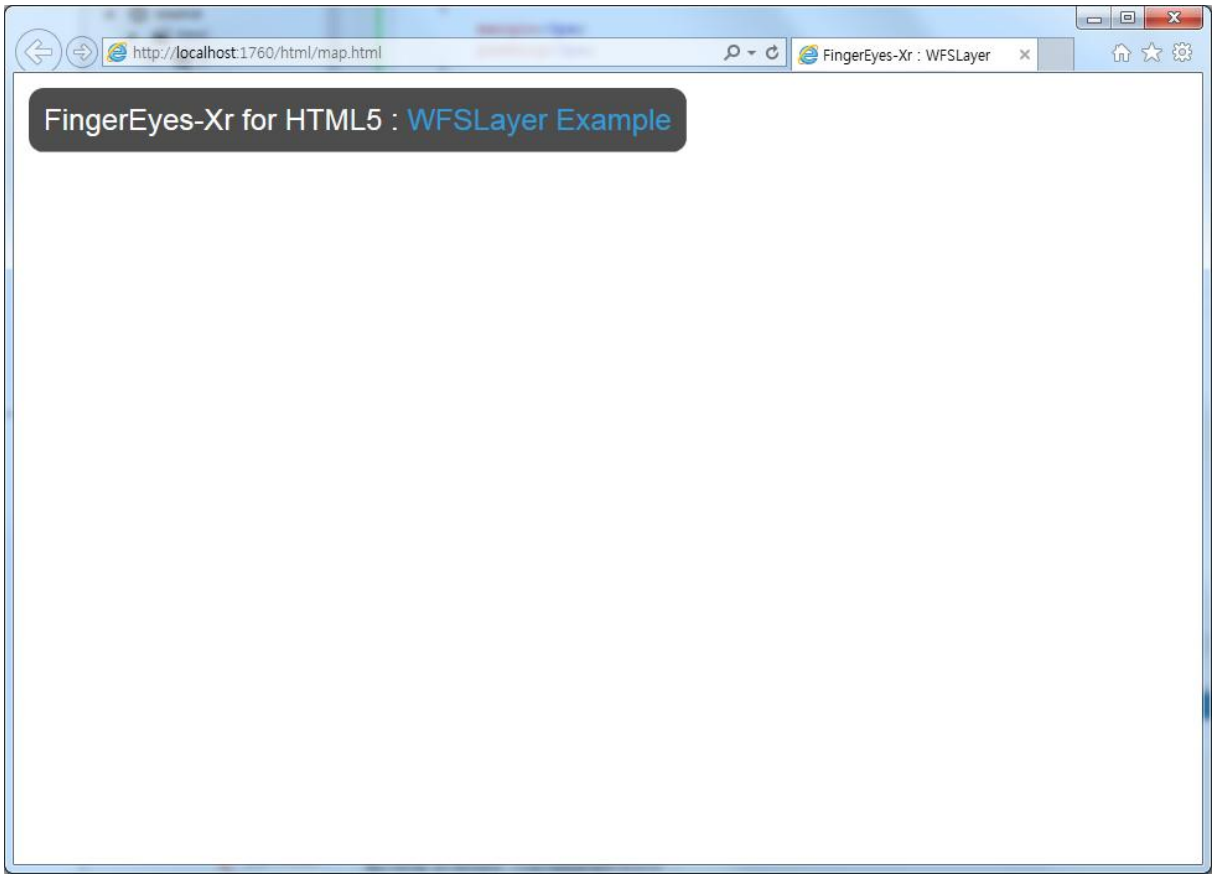

그림 4. 스타일이 적용된 UI

이제 UI와 스타일 적용이 완료되었으므로, 코드를 작성해 보겠습니다. **코드 2. 기본 Script**에서 4번 줄에 다음 코드를 추가합니다.

```
01  var map = null;
```

코드 5. DIV와 연결될 지도 객체 정의

이 map 변수는 지도 객체에 대한 참조로 사용됩니다. 이제 <body>에 대한 onload 이벤트를 load() 함수로 지정합니다.

```
01  <body onload="load()">
```

코드 6. Body 요소의 onload 이벤트 지정

그리고 load 함수를 다음처럼 추가합니다.

```
01  function load() {
02      map = new Xr.Map("mapDiv", {});
03
04
05
06  }
```

코드 7. Body의 onload 이벤트 함수

2번 코드는 id가 mapDiv인 DIV를 이용하여 지도 객체를 생성하고 이를 앞서 정의해 둔 map 변수에 저장하고 있습니다.

이제 WFS 레이어를 표시하기에 앞서 항공영상 지도를 배경지도로 깔고 그 위에 WFS 레이어를 표시하도록 하겠습니다. 항공영상 지도를 표시하는 것은 다음 코드를 통해 가능하며 load 함수의 마지막 부분에 추가합니다.

```
01  var lyr = new Xr.layers.TileMapLayer("basemap",
02      {
03          proxy: "http://222.237.78.208:8080/Xr",
04          url: "http://222.237.78.208:8080/muan_tilemaps",
05          ext: "jpg"
06      }
07  );
```

코드 8. 항공영상 배경지도 레이어 생성

위의 레이어는 항상영상 지도를 제공하는 공간 서버로부터 항공영상 지도를 서비스 받아 화면에 표시해 주기 위한 레이어를 생성하는 코드입니다.

이제 WFS 레이어를 표시하기 위해 WFSLayer를 생성하겠습니다. 다음 코드를 load 함수의 마지막에 추가합니다.

```
01  var parcLyr = new Xr.layers.WFSLayer(
02      "wfs_parc",
03      {
04          url: "http://localhost:8080/OGC/test/ows",
05          typeName: "test:muan_parc"
```

```
06      }
07  );
```

코드 9. 지적도에 대한 WFS 레이어 생성

위의 코드를 살펴보면, 먼저 1번 코드는 WFS 레이어를 사용하기 위해 WFSLayer 객체를 생성하여 parcLyr이라는 변수에 담고 있습니다. 이 클래스의 생성자는 2개의 인자를 갖습니다. 첫 번째는 레이어에 대한 고유한 이름입니다. 이 이름 값을 이용하여 언제든지 해당 이름에 대한 레이어 객체를 참조할 수 있습니다. 두 번째 인자는 Object 타입의 객체로 이 객체에는 url와 typeName이라는 프로퍼티를 가지고 있어야 합니다. url 프로퍼티를 통해 WFS 레이어에 대한 서버 URL을 지정합니다. 그리고 typeName에는 공간서버에서 WFS로 서비스되고 있는 레이어의 이름을 지정합니다.

이제 앞서 생성한 항공영상 레이어 객체에 대한 변수 lyr과 지적도에 대한 WFS 레이어 객체에 대한 변수 parcLyr을 레이어 관리자에 추가해야 합니다. 이에 대한 코드는 아래와 같습니다. load 함수의 마지막 줄에 추가합니다.

```
01  var lm = map.layers();
02  lm.add(lyr);
03  lm.add(parcLyr);
```

코드 10. 생성한 레이어 추가

이렇게 레이어를 최종적으로 추가 했으니 실행하면 지도가 표시될 것 같지만, 아직 한가지가 더 남아 있습니다. 그것은 레이어를 추가하고 난 뒤에 사용자에게 지도를 표시할 때 표시할 지도의 위치와 축척에 대한 것 입니다. 이에 대한 코드는 다음과 같으며 load 함수의 마지막 줄에 추가합니다.

```
01  map.onLayersAllReady(function () {
02      var cm = map.coordMapper();
03
04      cm.moveTo(151531, 246679);
05      cm.zoomByMapScale(766);
06
07      map.update();
08  });
```

코드 11. 레이어 추가 완료가 되면 지도의 표출

위의 코드에서 1번의 onLayersAllReady는 앞서 추가한 두 개의 레이어가 문제없이 추가가 되면 호출되는 이벤트 함수(Event Function 또는 Callback Function)를 지정하기 위한 것 입니다. 편의상 익명 함수를 사용하여 이벤트를 지정하였습니다. 해당 익명 함수의 코드 부분을 살펴보면, 먼저 2번 코드는 지도의 이동, 확대, 축소, 회전하는 기능과 지도 좌표와 화면 좌표계 간의 변환 기능을 제공하는 CoordMapper 객체를 가져와 cm 이라는 변수에 저장하고 있습니다. 이 cm 변수를 이용하여 4번 코드에서 화면 상에 표시될 지도의 중심 좌표를 지정하고 5번 코드에서 지도의 축척을 1:766으로 지정하기 위해 축척의 분모값인 766을 인자로 호출하고 있습니다. 최종적으로 7번 코드에서 update 함수를 호출하여 지도를 그리도록 하고 있습니다.

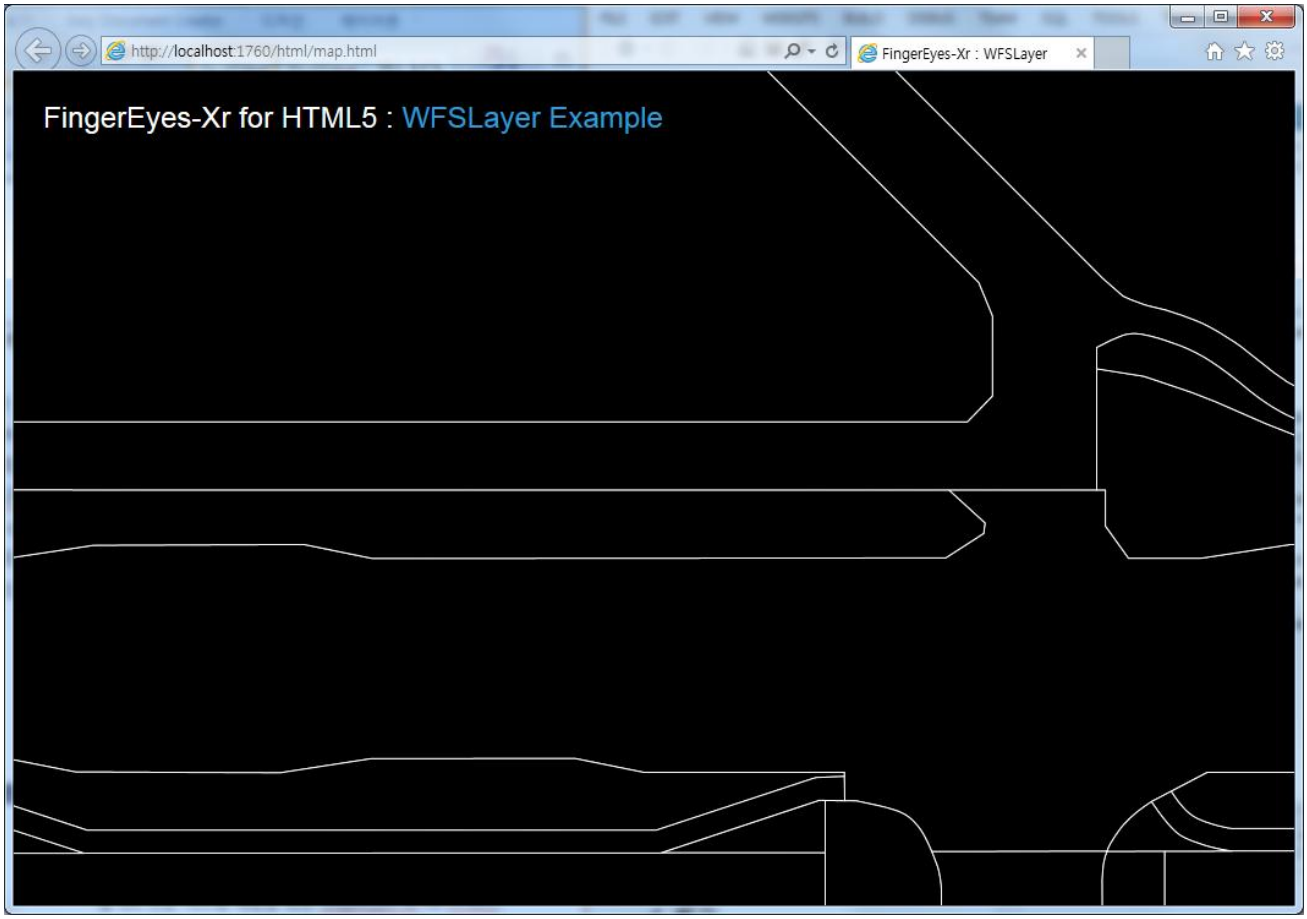

그림 5. 항공영상과 WFS 레이어 추가에 대한 실행 화면

실행해 보면 지적 필지가 검정색 채움과 하얀색 선으로 표시 되는 WFS 레이어를 볼 수 있습니다. 참고로 이 수치지도 밑에는 항공 영상이 표시되고 있는데, 지적 필지로 인해 가려져 보이지 않습니다. 이제 지적 필지의 그리기 심벌을 알맞게 변경하여 항공영상도 화면에서 보이도록 하겠습니다.

먼저 아래의 코드를 **코드 9. 지적도에 대한 WFS 레이어 생성** 부분의 WFS 레이어 객체를 생성하는 코드 바로 밑에 추가합니다.

```
01  var parcTheme = parcLyr.theme();
02  var parcPen = parcTheme.penSymbol();
03  var parcBrush = parcTheme.brushSymbol();
04
05  parcPen.color('#ffff00');
06  parcPen.width(2);
07
08  parcBrush.color('#ff0000');
09  parcBrush.opacity(0.0);
```

코드 12. 지적도에 대한 WFS 레이어 그리기 심벌 설정

위의 코드를 설명하면, 먼저 1번에서 해당 WFS 레이어의 테마(Theme) 객체를 얻고 이 테마 객체로부터 선에 대한 심벌을 설정할 수 있는 펜(Pen) 객체와 채움색을 설정할 수 있는 브러쉬(Brush) 객체를 얻어와 각각 parcPen과 parcBrush 변수에 담고 있습니다. 먼저 parcPen 변수를 통해 5번과 6번 코드에서처럼 선

의 색상을 노란색(#ffff00)으로 지정하고 선의 굵기를 2px로 지정합니다. 그리고 8번과 9번 코드처럼 채움색을 빨간색(#ff0000)으로 지정하고 불투명도를 0으로 지정하여 완전히 투명하게 그리도록 합니다. 실행하면 아래와 같은 결과를 볼 수 있습니다.

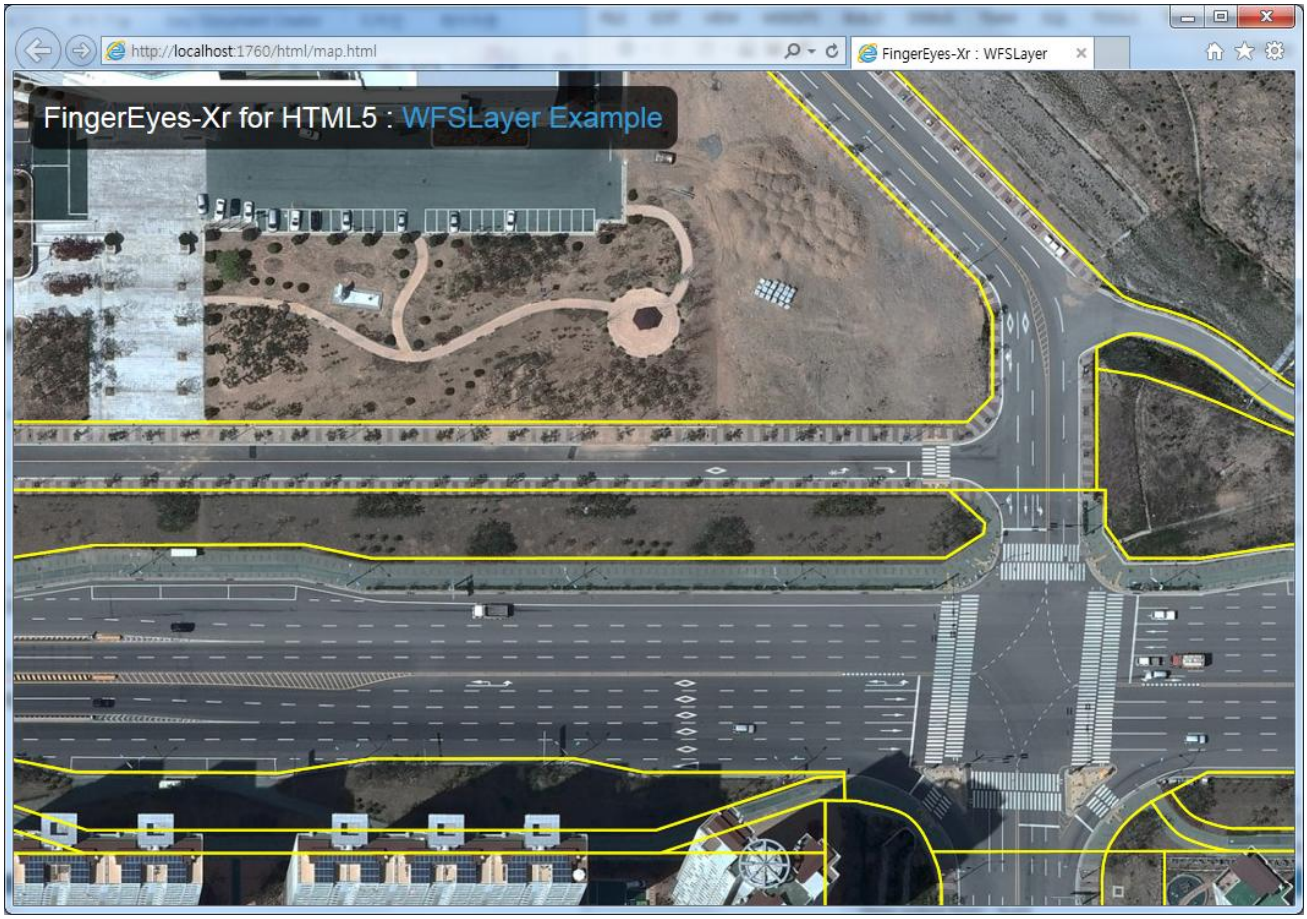

그림 6. 수치지도 그리기 심벌 설정

위의 화면을 보면 지적필지의 채움색이 투명하게 되어 아래에 깔린 항공영상까지 볼 수 있습니다. 이처럼 WFS 레이어는 클라이언트 단에서 자유롭게 그리기 심벌을 설정할 수 있어 효율적입니다.

이제 지도를 조작하는 4개의 버튼을 추가하겠습니다. **코드 3. 기본 UI 구성**에서 6번째 줄 바로 아래에 다음 코드를 추가합니다.

```
01  <br />
02  <input type="button" value="ZoomIn" onclick="onZoomInMap();">
03  <input type="button" value="ZoomOut" onclick="onZoomOutMap();">
04  <input type="button" value="Rotate" onclick="onRotateMap();">
05  <input type="button" value="RevRotate" onclick="onRevRotateMap();">
```

이제 이 4개의 Button에 대한 코드를 추가 하겠습니다. 이 4개의 버튼은 다음과 같은 기능을 수행합니다.

버 튼	기 능
ZoomIn	지도의 확대

ZoomOut	지도의 축소
Rotate	지도를 10도 단위로 회전
RevRotate	지도를 -10도 단위로 회전

표 1. 버튼의 기능

이미 이 4개의 버튼 컨트롤에는 사용자가 클릭했을 때에 대한 이벤트 함수가 지정 되어 있습니다. 그러나 아직 이 이벤트 함수의 정의가 되어 있지 않았는데, 여기서 이벤트 함수를 추가 하겠습니다.

먼저 ZoomIn 버튼에 대한 클릭 이벤트 함수로 지정한 onZoomInMap 함수에 대한 코드는 다음과 같습니다.

```
01 function onZoomInMap() {
02     var cm = map.coordMapper();
03     cm.zoomByMapScale(cm.mapScale() * 0.5);
04     map.update();
05 }
```

코드 13. 지도 확대 버튼에 대한 클릭 이벤트 코드

그리고 ZoomOut 버튼에 대한 클릭 이벤트 함수로 지정한 onZoomOutMap 함수에 대한 코드는 다음과 같습니다.

```
01 function onZoomOutMap() {
02     var cm = map.coordMapper();
03     cm.zoomByMapScale(cm.mapScale() * 2);
04     map.update();
05 }
```

코드 14. 지도 축소 버튼에 대한 클릭 이벤트 코드

10도 Rotate 버튼에 대한 클릭 이벤트 함수로 지정한 onRotateMap 함수에 대한 코드는 다음과 같습니다.

```
01 function onRotateMap() {
02     var cm = map.coordMapper();
03     cm.rotate(10);
04     map.update();
05 }
```

코드 15. 지도 회전 버튼에 대한 클릭 이벤트 코드

RevRotate 버튼에 대한 클릭 이벤트 함수로 지정한 onRevRotateMap 함수에 대한 코드는 다음과 같습니다.

```
01 function onRevRotateMap() {
02     var cm = map.coordMapper();
03     cm.rotate(-10);
04     map.update();
05 }
```

코드 16. 지도 반대 회전 버튼에 대한 클릭 이벤트 코드

이미 CoordMapper 객체에 대해 언급 하였듯이, 이 객체를 이용하여 지도를 확대하고 축소할 수 있으며 이동 및 회전이 가능합니다.

실행하여 버튼을 통해 지도의 확대, 축소 및 회전을 자유롭게 할 수 있습니다. 그러나 마우스 휠을 통해서 지도의 확대 및 축소가 되지 않는데 이는 줌레벨 컨트롤(Zoom Level Control)을 지도에 붙이게 되면 가능합니다. 이 줌레벨 컨트롤 부분에 대해서는 이후에 자세히 살펴보겠습니다.

그런데, 지도를 계속 축소해 나가면 속도 상에 문제가 발생합니다. 아래처럼 지도를 계속 축소 하다 보면 화면상에 표시되는 필지의 개수가 매우 많아 지게 되고 너무 많은 필지를 화면 상에 표출함에 있어 속도가 상당히 떨어지게 됩니다.

그림 7. 너무 많은 필지의 표현으로 지도 표출 속도가 크게 떨어짐

이러한 문제에 대한 개선 방안은 해당 WFS 레이어를 지정된 축척 범위에서만 표현되도록 하는 것입니다. 사실 위의 지도 화면의 축척에서는 지적도 레이어를 볼 필요가 없다고 할 수 있습니다. 이처럼 수치지도 레이어를 지정된 범위에서만 표출되도록 하는 방법에 대해 설명하겠습니다.

아래의 코드를 앞서 작성한 **코드 12. 수치지도 레이어 그리기 심벌**에 대한 코드 마지막에 추가합니다.

```
01  var visibility = parcLyr.visibility();
02  visibility.visibleByScale(true);
```

```
03  visibility.fromScale(1);
04  visibility.toScale(2001);
```

코드 17. 수치지도 레이어 표시 축척 범위 지정

위의 코드에서 1번은 해당 수치지도의 가시성(Visibility)을 조정할 수 있는 객체를 얻어 와 임시 변수에 저장하고 있습니다. 그리고 2번은 이 변수를 통해 축척에 대해 가시성 여부를 지정하겠다는 의미로 true를 인자로 하여 visibleByScale 함수를 호출하고 있습니다. 가시성에 대한 축척 범위를 지정하기 위해 3번과 4번 코드를 실행하는데, 위의 코드는 (1:2001 ~ 1:1]의 범위를 지정하고 있습니다. 1:2001은 범위에 표함되지 않으며 1:1은 범위에 포함된다는 점에 유의하시기 바랍니다.

이제 WFS 레이어에 대한 라벨(Label)을 표시해 보도록 하겠습니다. WFS 레이어는 수치지도로써 공간 데이터와 속성 데이터로 구성되는데, 라벨은 이 속성 데이터를 이용해 1:1로 연관되는 공간 데이터에 대한 도형에 텍스트로 표시되는 정보입니다. 속성 데이터는 필드 정보가 있습니다. 이미 우리가 추가한 수치지도에 대한 필드 중 PAR_LBL이라는 필드를 사용해 라벨을 표출하도록 하겠습니다. 아울러 라벨의 표시도 지정된 축척의 범위 안에서만 표출되도록 하겠습니다. 이러한 라벨 표시에 대한 코드는 다음과 같으며 **코드 17. 수치지도 레이어 표시 축척 범위 지정**에 대한 코드 다음에 추가합니다.

```
01  var label = parcLyr.label();
02  label.visibility().visibleByScale(true);
03  label.visibility().fromScale(0);
04  label.visibility().toScale(2001);
05  label.enable(true);
06  label.formatter().fieldName("PAR_LBL");
07
08  var labelTheme = label.theme();
09  labelTheme.symbol().strokeColor("#000000");
10  labelTheme.symbol().strokeWidth(2);
11
12  labelTheme.symbol().size(14);
13  labelTheme.symbol().fontFamily('맑은 고딕');
14  labelTheme.symbol().color("#ff0000");
```

코드 18. 라벨 설정

코드를 설명하면, 1번은 해당 수치지도 레이어의 라벨 객체를 얻고 2번은 라벨 객체에서 가시화 설정을 위한 객체를 얻습니다. 가시화 설정을 위한 객체를 이용해서 4번 ~ 6번 코드처럼 해당 축척 범위를 지정합니다. 7번 코드는 라벨을 표시하겠다는 것이고 8번은 라벨 표시를 위해 사용할 필드명을 지정합니다. 그리고 10번과 11번은 라벨을 표시할 때 사용하는 텍스트 심벌을 지정하기 위해 테마(Theme)와 심벌(Symbol) 객체를 얻는 코드입니다. 이렇게 얻은 코드를 통해 12번 ~ 17번까지 외곽선의 색상과 굵기 그리고 폰트의 크기와 폰트명, 텍스트 색상을 지정합니다. 실행을 하면 다음과 같습니다.

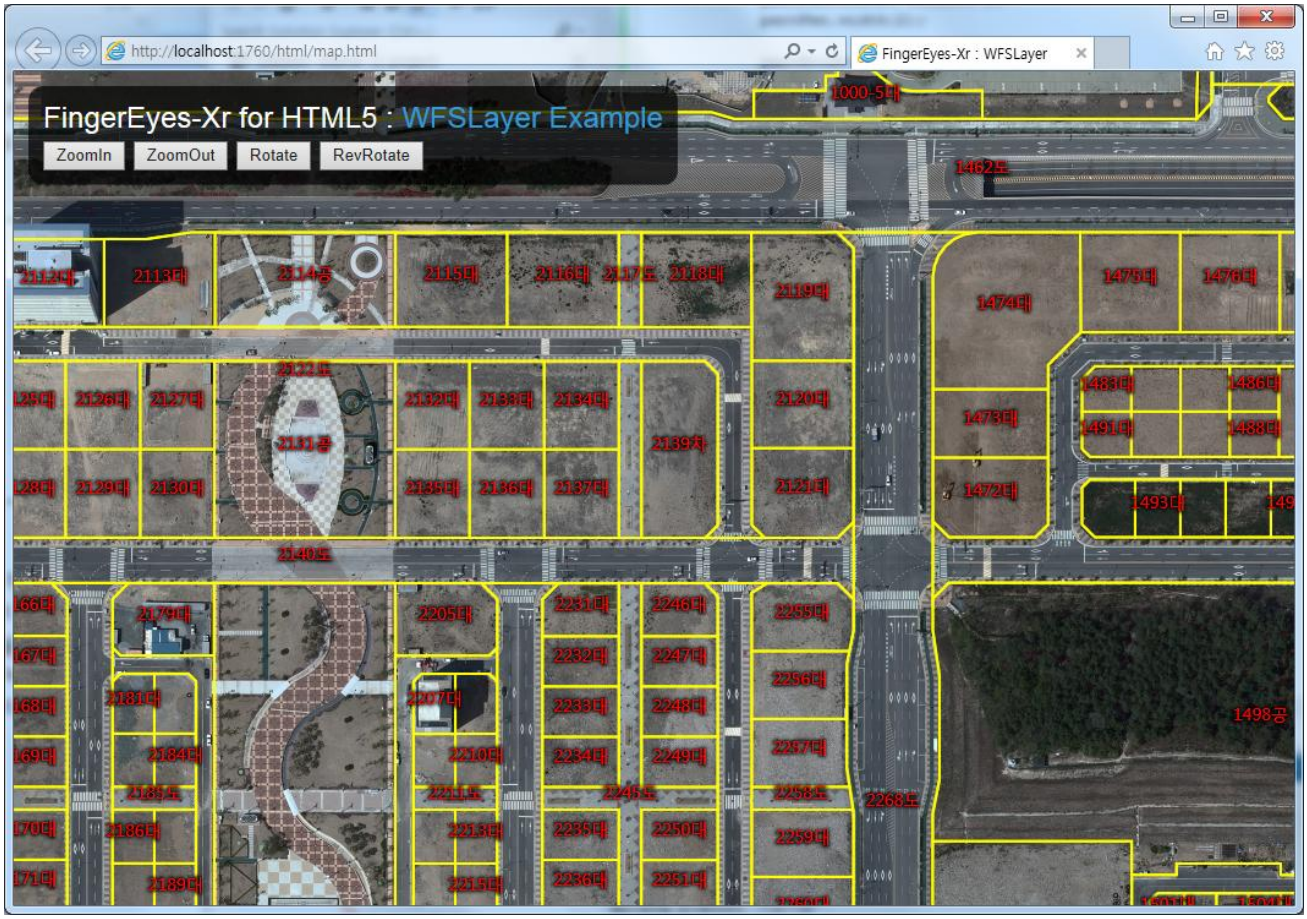

그림 8. 라벨 표시

이제 지도를 조작할 수 있는 인덱스 맵, 줌 레벨 컨트롤, 축척바 컨트롤을 화면에 배치해 보도록 하겠습니다. 먼저 축척바 컨트롤을 화면 상에 배치하도록 하겠습니다. 아래의 코드를 load 함수의 마지막 줄에 추가합니다.

```
01  var ctrl = new Xr.ui.ScaleBarControl("sbc", map);
02  map.userControls().add(ctrl);
```

코드 19. 축척바 컨트롤 추가

축척바 컨트롤 생성을 위해 ScaleBarControl 객체를 생성합니다. 첫번째 인자는 이 컨트롤에 대한 고유한 이름입니다. 이 이름을 통해 컨트롤에 접근이 가능합니다. 실행하면 아래의 그림처럼 화면 좌측 하단에 축척바 컨트롤이 표시됩니다.

그림 9. 축척바 컨트롤 표시

다음으로 줌 레벨 컨트롤을 지도 화면에 표시하도록 하겠습니다. 이 줌 레벨 컨트롤을 표시하기 위해 load 함수의 마지막 줄에 아래의 코드를 추가합니다.

```
01  var ctrl2 = new Xr.ui.ZoomLevelControl("zlc", map);
02
03  ctrl2.mapScales([
04      383.06022560915073, 766.1204512466478, 1915.3011281307926,
05      3830.6022556379635, 7661.204511956243, 19153.011279550446,
06      38305.7388117497, 76612.04511893881, 153224.09023821776]);
07
08  map.userControls().add(ctrl2);
```

코드 20. 줌 레벨 컨트롤 추가

위의 코드에서 3번 코드에서 마우스 휠 조작이나 컨트롤 조작을 통해 반영되는 지도의 축척에 대한 분모 값을 배열 형태로 지정합니다. 실행하면 아래의 화면처럼 줌 레벨 컨트롤이 화면에 표시되는 것을 볼 수 있습니다.

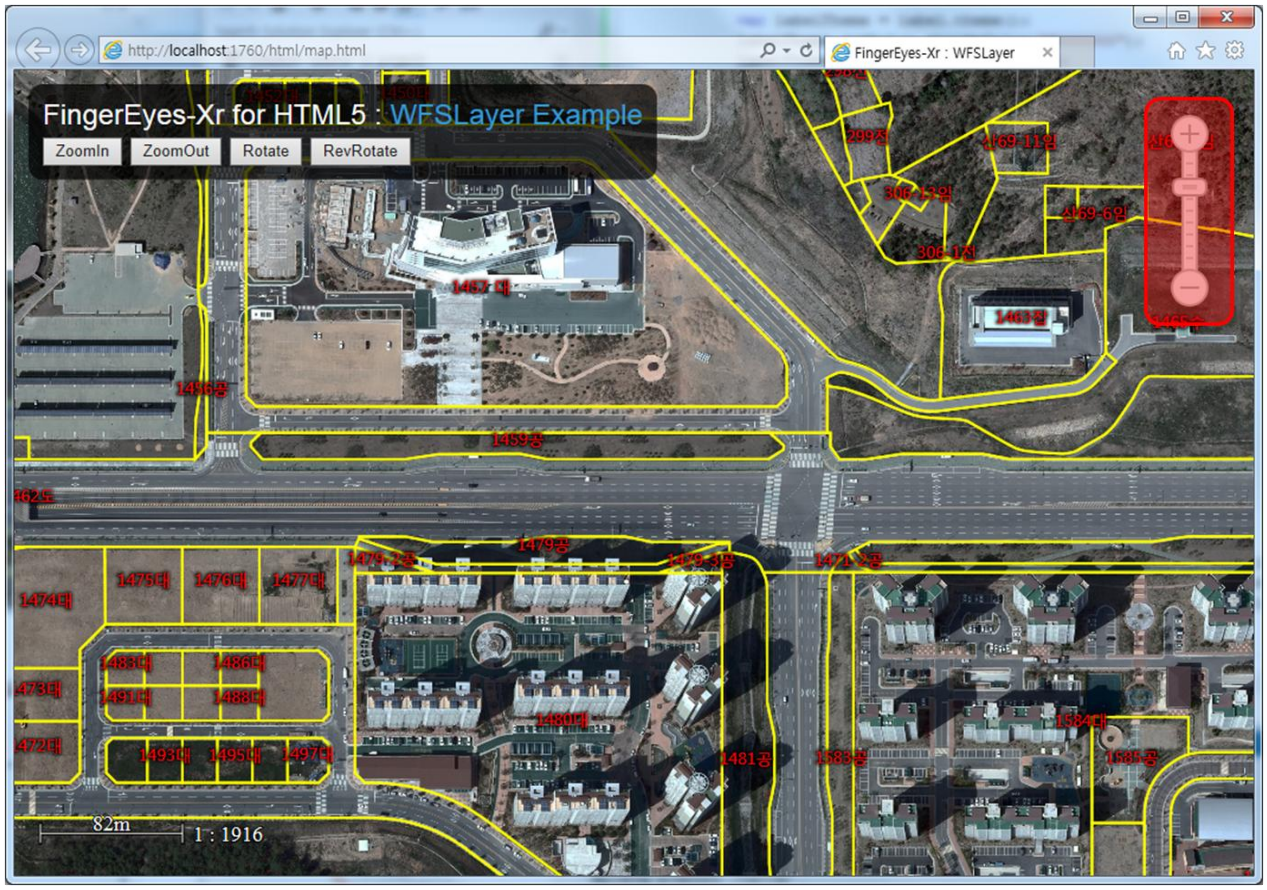

그림 10. 줌 레벨 컨트롤 추가

끝으로 인덱스 맵 컨트롤을 화면에 표시하도록 하겠습니다. 아래의 코드를 load 함수의 가장 마지막 줄에 추가합니다.

```
01  var ctrl3 = new Xr.ui.IndexMapControl("imc", map,
02      "http://222.237.78.208:8080/Xr?layerName=INDEXMAP");
03
04  ctrl3.size(200, 270);
05  map.userControls().add(ctrl3);
```

코드 21. 인덱스맵 컨트롤 추가

인덱스 맵 컨트롤 생성을 위해 IndexMapControl 객체를 생성하는데, 이때 생성자의 세번째 인자에 인덱스 맵에 표시되는 지도에 대한 Connection String을 지정하게 됩니다. 그리고 3번 코드에서 인덱스맵 컨트롤의 크기를 160px, 세로 300px로 지정하고 있습니다. 실행하면 다음처럼 인덱스 맵이 화면에 표시되는 것을 볼 수 있습니다.

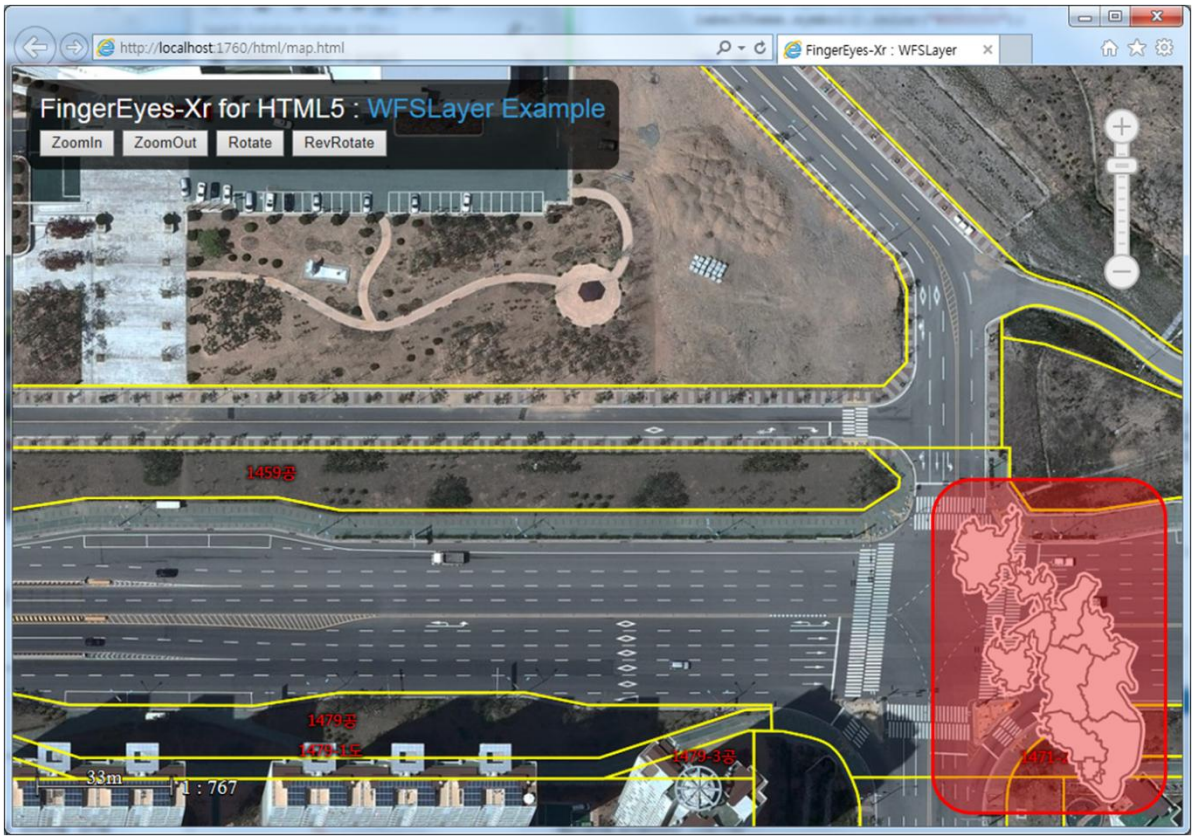

그림 11. 인덱스맵 컨트롤 표시

이상으로 WFS로 좌표와 속성 데이터를 서비스 받아 화면에 표시할 수 있는 WFSLayer에 대한 내용과 다양한 지도 조작 컨트롤을 추가하는 내용에 대해 살펴보았습니다. 끝으로 지금까지 작성한 코드에 대한 전체 내용은 아래와 같습니다.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
    body
    {
        margin:0px;
        padding:0px;
    }

    #mainLayout
    {
        position:relative;
        width:100%;
        height:100%;
        border: none;
    }
```

```
    #mapDiv {
        top:0px;
        left:0px;
        position:relative;
        width:100%;
        height:100%;
        border: none;
        overflow:auto;
    }

    #title {
        top:12px;
        left:12px;
        padding: 12px;
        position:absolute;
        background:rgba(0,0,0,0.7);
        border:none;
        overflow:auto;
        border-radius: 12px;
        font-size: 24px;
        color: #ffffff;
        font-family: "Arial";
    }
    </style>


    <title>FingerEyes-Xr : WFSLayer</title>

    <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>

    <script type="text/javascript">
        var map = null;

        function load() {
            map = new Xr.Map("mapDiv", {});

            var lyr = new Xr.layers.TileMapLayer("basemap",
                {
                    proxy: "http://222.237.78.208:8080/Xr",
                    url: "http://222.237.78.208:8080/muan_tilemaps",
                    ext: "jpg"
                }
            );
```

```
var parcLyr = new Xr.layers.WFSLayer(
    "wfs_parc",
    {
        url: "http://localhost:8080/OGC/test/ows",
        typeName: "test:muan_parc"
    }
);

var parcTheme = parcLyr.theme();
var parcPen = parcTheme.penSymbol();
var parcBrush = parcTheme.brushSymbol();

parcPen.color('#ffff00');
parcPen.width(2);

parcBrush.color('#ff0000');
parcBrush.opacity(0.0);

var visibility = parcLyr.visibility();
visibility.visibleByScale(true);
visibility.fromScale(1);
visibility.toScale(2001);

var label = parcLyr.label();
label.visibility().visibleByScale(true);
label.visibility().fromScale(0);
label.visibility().toScale(2001);
label.enable(true);
label.formatter().fieldName("PAR_LBL");

var labelTheme = label.theme();
labelTheme.symbol().strokeColor("#000000");
labelTheme.symbol().strokeWidth(2);

labelTheme.symbol().size(14);
labelTheme.symbol().fontFamily('맑은 고딕');
labelTheme.symbol().color("#ff0000");

var lm = map.layers();
lm.add(lyr);
lm.add(parcLyr);
```

```
    map.onLayersAllReady(function () {
        var cm = map.coordMapper();

        cm.moveTo(151531, 246679);
        cm.zoomByMapScale(766);

        map.update();
    });

    var ctrl = new Xr.ui.ScaleBarControl("sbc", map);
    map.userControls().add(ctrl);

    var ctrl2 = new Xr.ui.ZoomLevelControl("zlc", map);

    ctrl2.mapScales([
        383.06022560915073, 766.1204512466478, 1915.3011281307926,
        3830.6022556379635, 7661.204511956243, 19153.011279550446,
        38305.7388117497, 76612.04511893881, 153224.09023821776]);

    map.userControls().add(ctrl2);

    var ctrl3 = new Xr.ui.IndexMapControl("imc", map,
        "http://222.237.78.208:8080/Xr?layerName=INDEXMAP");

    ctrl3.size(200, 270);
    map.userControls().add(ctrl3);


}

function onZoomInMap() {
    var cm = map.coordMapper();
    cm.zoomByMapScale(cm.mapScale() * 0.5);
    map.update();
}

function onZoomOutMap() {
    var cm = map.coordMapper();
    cm.zoomByMapScale(cm.mapScale() * 2);
    map.update();
}
```

```
        function onRotateMap() {
            var cm = map.coordMapper();
            cm.rotate(10);
            map.update();
        }

        function onRevRotateMap() {
            var cm = map.coordMapper();
            cm.rotate(-10);
            map.update();
        }

    </script>

</head>
<body onload="load()">
    <div id="mainLayout">
        <div id="mapDiv"></div>
        <div id="title">
            FingerEyes-Xr for HTML5 :
            <font color="#349bd6">WFSLayer Example</font>
            <br />
            <input type="button" value="ZoomIn" onclick="onZoomInMap();">
            <input type="button" value="ZoomOut" onclick="onZoomOutMap();">
            <input type="button" value="Rotate" onclick="onRotateMap();">
            <input type="button" value="RevRotate" onclick="onRevRotateMap();">
        </div>
    </div>
</body>

</html>
```