FingerEyes-Xr for Flex를 이용한 공간 데이터 편집 API

(주)지오서비스

2015

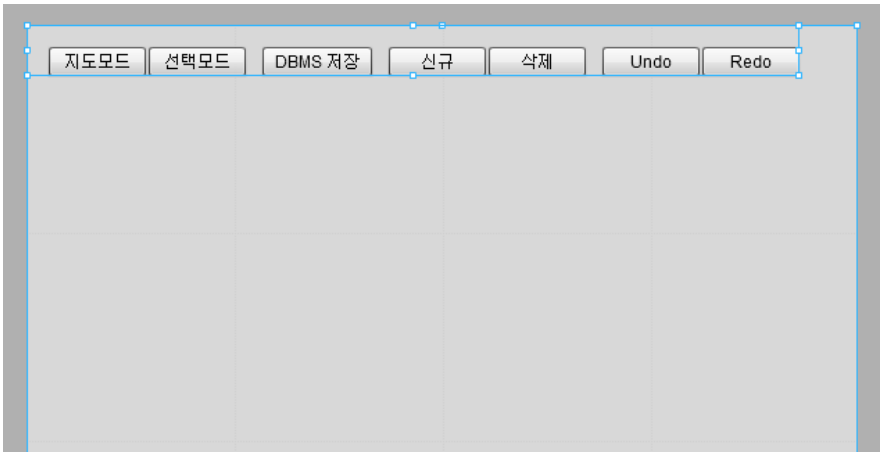
목 차


geoservice
Good Solution
Good Partner

FingerEyes-Xr for Flex 를 이용하여 웹에서 공간 데이터를 편집하는 기능에 대한 개발 문서입니다. 편집 대상이 되는 공간 데이터는 PostgreSQL DBMS 에 저장되어 있으며 GeoService-Xr 공간서버를 통해 서비스 됩니다.

1. UI 구성

지도를 살펴보고 편집할 도형을 선택할 수 있는 있으며, 선택된 도형을 편집하여 DBMS 에 저장할 수 있는 버튼 등을 배치합니다. 아래와 같은 화면처럼 모두 7 개의 버튼을 배치합니다.



화면에는 표시되어 있지 않으나 FingerEyes-Xr 의 XrMap 컴포넌트도 배치되어 있습니다. 아래는 위의 UI 에 대한 코드입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
                            xmlns:s="library://ns.adobe.com/flex/spark"
                            xmlns:mx="library://ns.adobe.com/flex/mx"
                            xmlns:xr="geoservice.*">
<xr:XrMap id="map" width="100%" height="100%" x="0" y="0" />
<s:HGroup paddingTop="16" paddingLeft="16" gap="2">
   <mx:Button label="지도모드" click="onMapMode(event)" />
   <mx:Button label="선택모드" click="onSelectMode(event)" />

   <s:Spacer width="8" />
   <mx:Button label="DBMS 저장" click="onCommit(event)" />

   <s:Spacer width="8" />
   <mx:Button label="신규" click="onNew(event)" />
   <mx:Button label="삭제" click="onDelete(event)" />

   <s:Spacer width="8" />
   <mx:Button label="Undo" click="onUndo(event)" />
   <mx:Button label="Redo" click="onRedo(event)" />
</s:HGroup>
</s:Application>
```

2. 레이어 구성

아래처럼 Application 의 initialize 이벤트에 onInit 함수를 지정합니다. 이 onInit 함수에 레이어를 구성하는 코드를 입력할 것입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
    xmlns:mx="library://ns.adobe.com/flex/mx"
    xmlns:xr="geoservice.*"
    initialize="onInit(event)">
```

onInit 함수는 아래와 같습니다.

```
1  protected function onInit(event:FlexEvent):void
2  {
3      var basemap:XrTileMapLayer = new XrTileMapLayer("basemap",
4          "http://222.237.78.208:8080/muan_tilemaps");
5      basemap.saturation = 0.42;
6      map.layers.addLayer(basemap);
7
8      var bldLyr:XrShapeMapLayer = new XrShapeMapLayer("bld",
9          "http://222.237.78.208:8080/Xr?layerName=MUAN_BLD");
10     bldLyr.theme.properties = {
11         lineThickness:2.0, lineAlpha:1.0, lineColor:0xffff00,
12         fillAlpha:0.25, fillColor:0xffff00
13     };
14
15     bldLyr.requestAttributeAlways = true;
16     bldLyr.noCache = true;
17     map.layers.addLayer(bldLyr);
18
19     var ml:XrMashupLayer = new XrMashupLayer("mashup");
20     map.layers.addLayer(ml);
21
22     map.edit.targetLayer = ml;
23
24     map.viewControls.scaleLevels =
25         [200000,100000,50000,25000,10000,5000,2500,1000,500];
26     map.moveMap(new XrCoordinate(151531,246679));
27     map.viewControls.scaleLevel = 7;
28 }
```

위의 코드에서 편집 대상이 되는 레이어는 8 번 코드에 있는 XrShapeMapLayer 클래스 타입인 bldLyr 이라는 변수로써, 레이어 명이 "bld"입니다. 이 편집 대상 레이어는 건물에 대한 수치지도 레이어입니다. 앞으로 이 편집 대상 레이어를 찾을 때는 항상 레이어 명인 "bld"를 이용합니다. 또한 19 번 코드에 매쉬업 레이어를 추가했는데, 이 매쉬업 레이어의 이름은 "mashup"이고, 이 매쉬업 레이어를 이용해 실제 편집 기능이 이루어집니다. 그래서 22 번 코드에 편집에 대한 대상 레이어를 이 매쉬업 레이어 변수로 지정하고 있습니다.

실제 편집은 "bld" 레이어이지만, 논리적으로는 "mashup" 레이어에서 이루어진다는 것이 이해하기 어려울 수 있습니다. 이 부분을 좀더 자세히 설명하면 다음과 같습니다. 사용자는 마우스를 이용해 편집하고자 하는 "bld" 레이어의 도형을 클릭합니다. 클릭하여 선택된 "bld" 레이어의 도형은 "mashup" 레이어에 복사됩니다. 이렇게 복사된 "mashup" 레이어의 도형에 대해 편집이 이루어 지는 것입니다. 이렇게 "mashup" 레이어에 대해 편집된 내용은 공간 DBMS 에 반영되고, 이렇게 반영된 DBMS 에 의해 자연스럽게 "bld" 레이어에 편집된 내용이 반영됩니다.

위의 코드에서 갑작스럽게 많은 클래스가 언급되어 있습니다. 이러한 클래스에 대한 import 구문이 필요한데요. 아래의 코드가 이러한 import 구문입니다.

```
1  import geoservice.base.XrCoordinate;
2  import geoservice.controls.*;
3  import geoservice.data.IXrShape;
4  import geoservice.data.XrPolygonShape;
```

```
5   import geoservice.events.XrEditFeatureAddedNewEvent;
6   import geoservice.events.XrEditFeatureSelectChangedEvent;
7   import geoservice.events.XrMapMouseEvent;
8   import geoservice.services.XrProposalFIDService;
9   import geoservice.services.XrUpdateShapeMapFromMashupService;
10  import geoservice.view.layers.IXrLayer;
11  import geoservice.view.layers.XrMashupLayer;
12  import geoservice.view.layers.XrShapeMapLayer;
13  import geoservice.view.layers.XrTileMapLayer;
14  import geoservice.view.mashups.IXrMashup;
15  import geoservice.view.mashups.XrPolygonMashup;
16  import geoservice.view.symbols.XrFillSymbol;
17  import geoservice.view.symbols.XrLineSymbol;
18
19  import mx.controls.Alert;
20  import mx.events.CloseEvent;
21  import mx.events.FlexEvent;
```

위의 코드는 실제 코딩을 하면서 자연스럽게 입력이 되지만, 이렇게 한번에 입력을 하는 것은 학습하는 분들이 혼란스럽지 않도록 하기 위함입니다. 이제 실행하면 다음과 같은 화면을 볼 수 있습니다.

항공 영상 배경 지도 위에 노란색의 건물 도형이 보입니다. 바로 이 건물 도형을 가지고 있는 "bld" 레이어가 실제 편집 대상입니다.

3. 선택 도형 편집

이제 편집할 도형을 선택하는 버튼에 대한 코드를 살펴보겠습니다. UI 버튼 중에 "지도모드"와 "선택모드"에 대해 살펴보겠습니다. "지도모드"는 마우스를 이용해 지도를 이동하는 일반적인 지도 보기

모드(Mode)입니다. 반면 "선택모드"는 마우스를 이용해 지도를 이동할 수 있으면서 클릭을 통해 편집하고자 하는 건물 도형을 선택할 수 있는 모드(Mode)입니다.

먼저 "지도모드" 버튼의 클릭 이벤트에 대한 코드는 다음과 같습니다.

```
1  protected function onMapMode(event:MouseEvent):void
2  {
3      _selectMode = false;
4      setMapViewMode();
5  }
```

전역 변수로 선언되어 있는 _selectMode 를 false 로 지정하고 setMapViewMode 함수를 호출하고 있습니다. _selectMode 변수와 setMapViewMode 함수는 다음과 같습니다.

```
1  private var _selectMode:Boolean = false;
2
3  private function setMapViewMode():void
4  {
5      map.edit.editMode = false;
7      map.edit.selectNone();
8      map.edit.editHistoryReset();
9
10     var ml:XrMashupLayer = map.layers.getLayer("mashup") as XrMashupLayer;
11     ml.removeAllMashups();
12 }
```

setMapViewMode 함수를 좀더 살펴보면 5 번 코드에서 편집 모드를 false 로 지정하고 7 번 코드에서 편집을 위해 선택된 도형에 대한 컨트롤 포인트(Control Point)가 표시되고 있다면 제거합니다. 8 번 코드는 편집 이력을 모두 제거 합니다. 10 ~ 12 번 코드는 "mashup" 레이어의 항목을 모두 제거합니다. 편집하고자 하는 도형을 마우스로 선택하면 선택 도형을 "mashup" 레이어의 항목으로 추가하게 됩니다. 이렇게 추가된 항목을 제거 하는 것입니다.

다음으로 "선택모드" 버튼의 클릭 이벤트에 대한 코드는 다음과 같습니다.

```
1  protected function onSelectMode(event:MouseEvent):void
2  {
3      _selectMode = true;
4  }
```

단순히 _selectMode 를 true 로 지정하고 있습니다. 지도뷰 위에서 마우스를 클릭했을때 발생하는 클릭 이벤트에서 _selectMode 가 true 일 경우 별도의 처리를 해주게 됩니다. 지도뷰에 대한 마우스 클릭 이벤트를 다음처럼 지정합니다.

```
1  <xr:XrMap id="map" width="100%" height="100%" x="0" y="0"
2      mapMouseClickEvent="onMapClick(event)" />
```

지도뷰의 마우스 클릭 이벤트 처리 함수인 onMapClick 는 다음과 같습니다.

```
   protected function onMapClick(event:XrMapMouseEvent):void
   {
1      if(_selectMode)
       {
2          var ml:XrMashupLayer = map.layers.getLayer("mashup") as XrMashupLayer;
3          var fids:Array = ml.getIDListByMousePoint(mouseX, mouseY, true);
4          if(fids.length == 0) {
5              if(map.edit.history.undoable) {
6                  mx.controls.Alert.show(
```

```
                    "편집한 내용이 있습니다. 무시하시겠습니까?",
                    "확인",
                    Alert.YES|Alert.NO|Alert.CANCEL,
                    null,
                    function(e:CloseEvent):void {
                        if (e.detail == Alert.YES)
                        {
                            toMashup(event.mouseEvent.localX, event.mouseEvent.localY);
                        }
                        else
                        {
                            var item:IXrMashup;
                            for each (item in ml.items) {
                                map.edit.setSketch(item.id);
                                break;
                            }
                        }
                    }
                );
            } else {
                toMashup(event.mouseEvent.localX, event.mouseEvent.localY);
            }
        }
    }
}
```

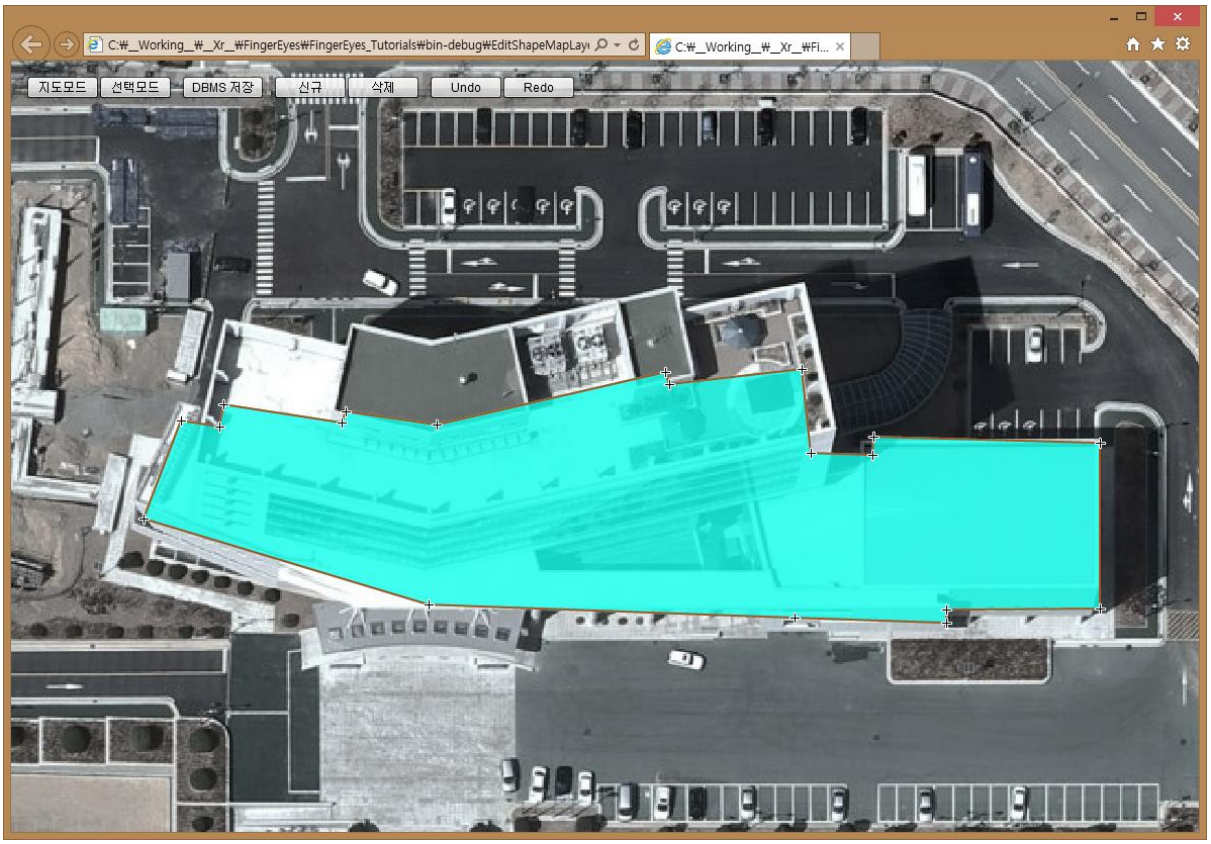
위의 코드를 살펴보면, 1 번에서 _selectMode 가 true 일 경우에 도형의 선택을 처리하고 있습니다. 2 번 코드는 "mashup" 레이어를 얻어와 ml 객체에 저장하고 있습니다. 3 번 코드는 마우스를 클릭한 지점에 "mashup" 레이어의 도형이 있다면 해당 도형의 fid 를 가져옵니다. 4 번 코드에서 만약 클릭한 지점에 "mashup" 레이어의 도형이 없다면 5 번 코드를 실행합니다. 5 번 코드는 이전에 편집한 이력이 있다면(Undo 가 가능하다면 편집한 이력이 있다는 의미) 6 번 코드를 실행합니다. 6 번 코드는 사용자에게 편집한 내용이 있는데 이를 무시하고 다른 도형을 선택할 것인지를 묻습니다. 만약 사용자가 예(YES) 버튼을 클릭하면 클릭한 X, Y 좌표를 인자로 넘겨 toMashup 함수를 실행하고 아니오(NO) 또는 취소 버튼을 클릭하면 10 ~ 13 번 코드를 실행합니다. 이 코드들은 기존에 선택된 도형을 다시 편집 가능한 상태로 유지시켜 줍니다. 14 번 코드는 이전에 편집된 내역이 없을 때 실행하는 코드로써 클릭한 X, Y 좌표를 인자로 넘겨 toMashup 함수를 실행합니다.

클릭한 지점에 대한 좌표를 받는 toMashup 함수는 클릭한 지점에 "bld" 레이어의 도형이 있다면 이 도형을 "mashup" 레이어의 항목으로 복사합니다. 이 함수의 코드는 다음과 같습니다.

```
1  private function toMashup(mouseX:int, mouseY:int):void
2  {
3     var lyr:XrShapeMapLayer = map.layers.getLayer("bld") as XrShapeMapLayer;
4     var fid:int = lyr.getFIDByMousePoint(mouseX, mouseY);
5     if(fid != -1)
6     {
7        var shp:IXrShape = lyr.shapeSet.rows[fid];
8        var polygon:XrPolygonShape = shp as XrPolygonShape;
9        if(polygon != null)
10       {
11          map.edit.selectNone();
12          map.edit.editHistoryReset();
13
14          var ml:XrMashupLayer = map.layers.getLayer("mashup") as XrMashupLayer;
15          ml.removeAllMashups();
16
17          var srcPolygons:Vector.<Vector.<XrCoordinate>> = polygon.polygons;
18          var mashup:IXrMashup = new XrPolygonMashup(fid, srcPolygons, false);
19
20          mashup.lineSymbol = new XrLineSymbol(
21             {lineThickness:2.0, lineAlpha:1.0, lineColor:0x00ffff});
22          mashup.fillSymbol = new XrFillSymbol(
23             {fillColor:0x00ffff, fillAlpha:0.5});
24
```

```
25              ml.addMashup(mashup);
26              ml.updateItem(fid, false);
27
28              map.edit.editMode = true;
29              map.edit.setSketch(fid);
30
31              return;
32          }
33      }
34
35      setMapViewMode();
36  }
```

위의 코드를 자세히 설명하면 다음과 같습니다. 먼저 3 번 코드에서 "bld" 레이어를 얻고, 4 번 코드에서 이 레이어의 getFIDByMousePoint 함수를 이용하여 클릭한 지점의 마우스 좌표를 이용해, 클릭한 지점에 건물 도형이 있다면 해당 도형의 id 를 fid 변수에 저장합니다. 5 번 코드에서 만약 선택된 도형이 있다면(-1 인 경우 선택된 도형이 없다는 의미), 7 번 코드에서 fid 에 해당하는 도형을 얻고 8 번 코드에서 해당 도형의 구체적인 좌표값을 얻기 위한 XrPolygonShape 타입의 객체를 얻습니다. 이러한 좌표값들을 이용해 18 번 코드에서 "mashup" 레이어에 추가할 항목(IXrMashup 타입)을 생성합니다. 이때 선택된 도형의 fid 값과 "mashup" 레이어에 추가하기 위해 생성한 항목의 id 값을 동일하게 합니다. 20 ~ 23 번 코드는 "mashup" 레이어에 추가한 항목의 그리기 심벌을 지정합니다. 25 번 코드와 26 번은 생성한 항목을 "mashup" 레이어에 추가하고 업데이트합니다. 그리고 28 번 코드에서 편집 모드로 전환하고 29 번에 "mashup" 레이어에 추가한 항목을 스케치화 합니다. 스케치화되면 아래의 화면처럼 마우스를 이용해 편집할 수 있는 컨트롤 포인트가 표시됩니다.



위의 화면처럼 일단 편집하고자 하는 도형이 스케치화 되면 마우스를 이용해서 전체를 이동할 수 있습니다. 또한 개별적인 좌표에 대해 이동 및 생성 그리고 삭제가 가능합니다. 생성 및 삭제는 Ctrl

키를 이용합니다. 이처럼 편집된 도형을 실제 공간 DBMS 에 저장하기 위해서 "DBMS 저장" 버튼을 클릭하면 됩니다. 이 "DBMS 저장" 버튼의 클릭 이벤트는 다음과 같습니다.

```
1  private function onCommit(event:Event):void
2  {
3     if(map.edit.select.length == 0)
      {
         showMsg("편집한 내용이 없습니다.");
         return;
      }
4
      _selectMode = false;
      mx.controls.Alert.show(
         "편집한 내용을 DBMS에 저장하시겠습니까?",
         "확인",
         Alert.YES|Alert.NO|Alert.CANCEL,
         null,
5        function(e:CloseEvent):void {
6           if (e.detail == Alert.YES)
            {
7              var id:uint = map.edit.select.getFID(0);
8              var shapeMapLyr:XrShapeMapLayer = map.layers.getLayer("bld")
                  as XrShapeMapLayer;
9              var mashupLyr:XrMashupLayer = map.layers.getLayer("mashup")
                  as XrMashupLayer;

10             var svc:XrUpdateShapeMapFromMashupService =
                  new XrUpdateShapeMapFromMashupService(
                     map.edit, shapeMapLyr, mashupLyr,
                     onSinkCompleted, onSinkError);

11             var params:Object = {targetId:id, sourceId:id };

12             if(!svc.run(params))
               {
                  showMsg("편집 서비스 파라메터가 옳바르지 않습니다.");
               }
            }
         }
      );
   }
```

"DBMS 저장" 버튼에 대한 클릭 이벤트인 onCommit 함수입니다. 먼저 3 번 코드에서 사용자가 선택한 도형이 있는지를 검사하여 만약 없다면, "편집 내용이 없습니다."라는 메세지를 표시하고 함수를 종료합니다. 만약 편집한 내용이 있다면 4 번 코드에서 _selectMode 를 false 로 지정하고 마우스 조작으로 다른 도형을 선택하지 못하게 합니다. 그리고 편집한 내용을 DBMS 에 저장할 것인지를 사용자에게 묻고 6 번에서 예(YES)를 클릭했는지 확인하고, 7 번 코드에서 현재 사용자가 편집하고 있는 도형의 id 값을 얻습니다. 그리고 편집을 위해 사용한 "bld" 레이어와 "mashup" 레이어의 객체를 얻습니다. 10 번 코드에서 이 두 레이어 객체를 사용하는데요, XrUpdateShapeMapFromMashupService 클래스를 통해 실제 공간 서버 DBMS 에 편집한 데이터를 반영시키기 됩니다. 이 XrUpdateShapeMapFromMashupService 클래스의 생성자에는 편집한 데이터를 실제 DBMS 에 반영시킬때 성공과 실패에 따라 각기 다른 콜백함수를 지정할 수 있습니다. 성공하면 onSinkCompleted 함수를, 실패하면 onSinkError 함수가 호출되도록 하고 있습니다. 12 번 코드에서 이 XrUpdateShapeMapFromMashupService 클래스 객체의 run 함수를 호출하여 서비스를 호출할 수 있는데요. run 함수는 반영시킬 id 값을 지정해 줘야 합니다. 바로 11 번 코드가 반영시킬 id 값에 대한 객체를 생성하고 있는데, targetId 는 공간 DBMS 에 저장된 데이터 중 업데이트될 데이터의 id 값이고, sourceId 는 업데이트 시킬 변경된 데이터를 가지고 있는 "mashup" 레이어의 항목에 대한 id 입니다. 앞서 XrUpdateShapeMapFromMashupService 의 run 함수의 호출에 대한

DBMS 반영에 대한 성공과 실패에 대한 콜백함수인, onSinkCompleted 와 onSinkError 는 다음과 같습니다.

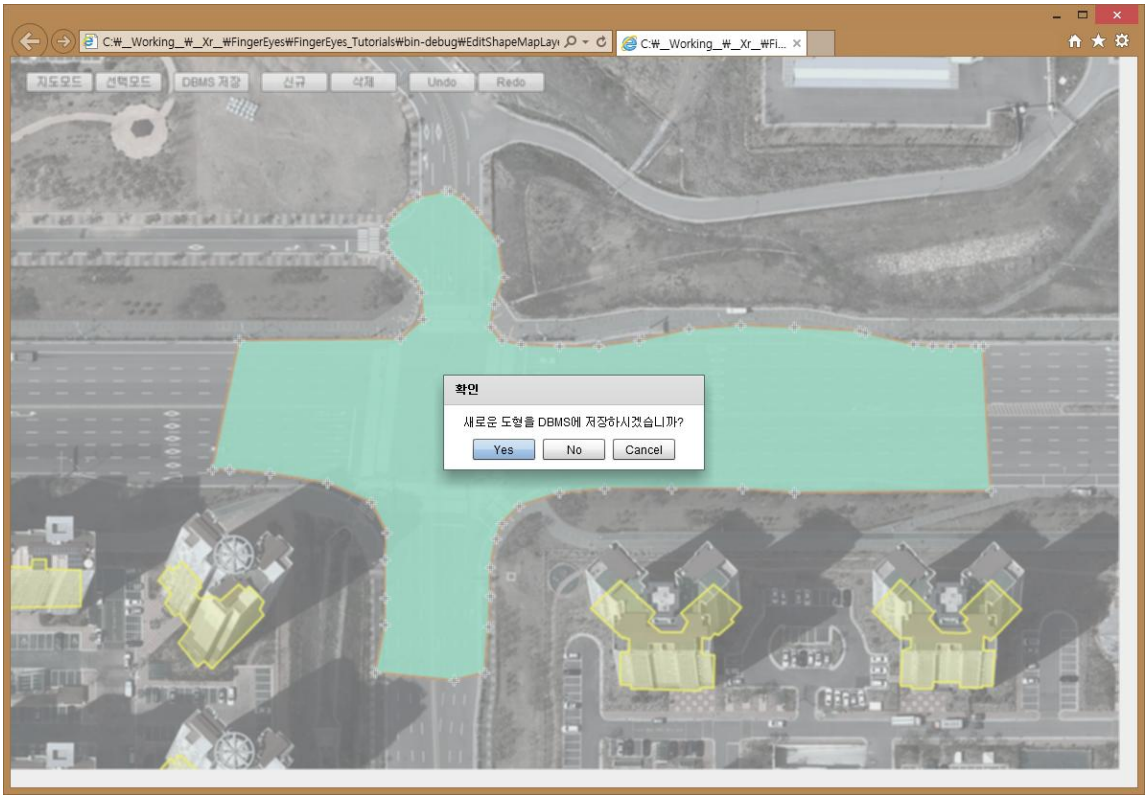
```
protected function onSinkCompleted():void
{
    setMapViewMode();
    map.update(true);
}

protected function onSinkError(msg:String):void
{
    showMsg(msg);
}
```

성공 함수에서는 선택모드에서 지도모드로 전환해주고 지도를 다시 그려주도록 하고 있습니다. 실패 함수에서는 실패 원인을 표시하기 위해 showMsg 함수를 실패 원인에 해당하는 문자열 값을 인자로 하여 호출하고 있습니다. showMsg 함수는 다음과 같습니다.

```
private function showMsg(msg:String):void
{
    mx.controls.Alert.show(msg);
}
```

4. 신규 도형 생성

새로운 공간 데이터로써의 도형을 생성하기 위해서는 "신규" 버튼을 클릭하면 됩니다. "신규" 버튼을 클릭하는 순간 마우스를 이용해 지도 화면에서 폴리곤을 그릴 수 있게 됩니다. 더블 클릭을 통해 폴리곤을 모두 다 그리게 되면 다음과 같이 새로운 도형을 DBMS 에 저장하겠냐고 사용자에게 묻습니다.



여기서 YES 버튼을 클릭하면 생성한 도형을 바로 공간 DBMS 에 저장합니다.

이러한 "신규" 버튼에 대한 클릭 이벤트를 먼저 살펴보겠습니다.

```
      protected function onNew(event:MouseEvent):void
      {
1        if(map.edit.history.undoable) {
2           mx.controls.Alert.show(
               "편집한 내용이 있습니다. 무시하고 새로운 도형을 추가하시겠습니까?",
               "확인",
               Alert.YES|Alert.NO|Alert.CANCEL,
               null,
3              function(e:CloseEvent):void {
4                 if (e.detail == Alert.YES)
                  {
5                    addNew();
                  }
               }
           );
        } else {
6           addNew();
        }
      }
```

위의 코드를 살펴보면, 먼저 새로운 도형을 생성하기에 앞서 만약 이전에 편집한 내용이 있는지를 1 번 코드에서 검사합니다. 이전에 편집한 내역이 있다면 사용자에게 "편집한 내용이 있습니다. 무시하고 새로운 도형을 추가하겠습니까?"를 묻습니다. 만약 사용자가 YES 버튼을 누르면 addNew 함수를 호출합니다. 이 addNew 함수가 새로운 도형을 마우스로 지도 위에 그릴 수 있게 하는 함수입니다. 6 번 코드는 이전에 편집한 내역이 없을 경우로 바로 addNew 함수를 호출합니다.

addNew 함수는 다음과 같습니다.

```
1   private function addNew():void
2   {
3      _selectMode = false;
4
5      var ml:XrMashupLayer = map.layers.getLayer("mashup") as XrMashupLayer;
6      ml.removeAllMashups();
7
8      map.edit.selectNone();
9      map.edit.editMode = true;
10     map.edit.addPolygon(0);
11  }
```

예상보다 코드가 짧습니다. 3 번에서 _selectMode 를 false 로 지정하여 도형 선택 모드를 벗어나게 합니다. 그리고 5~6 번은 "mashup" 레이어의 모든 항목을 제거합니다. 그리고 8 번에서는 기존에 스케치화된 내용을 지웁니다. 9 번은 편집 모드로 전환하고 10 번 코드에서 id 가 0 인 폴리곤을 마우스로 그리라는 함수를 호출합니다. 10 번 코드가 호출되는 시점부터 지도 화면에서 마우스를 이용해 폴리곤을 그릴 수 있습니다. 폴리곤을 그릴 때 더블클릭으로 폴리곤 생성을 완료할 수 있습니다. 이처럼 새로운 도형의 생성이 완료되는 시점에서 특정 이벤트가 발생하게 됩니다. 이 특정 이벤트는 editFeatureAddedNew 로써 다음처럼 이벤트가 지정됩니다.

```
1   <xr:XrMap id="map" width="100%" height="100%" x="0" y="0"
2      mapMouseClickEvent="onMapClick(event)"
3      editFeatureAddedNew="onEditNewFeatureAdded(event)" />
```

editFeatureAddedNew 이벤트에 할당된 onEditNewFeatureAdded 함수는 다음과 같습니다.

```
protected function onEditNewFeatureAdded(event:XrEditFeatureAddedNewEvent):void
{
1   var svc:XrProposalFIDService = new XrProposalFIDService(map.edit,
        onGetFIDCompleted, onGetFIDError);

2   var shapeMapLyr:XrShapeMapLayer = map.layers.getLayer("bld") as XrShapeMapLayer;

3   if(!svc.run({targetLayer: shapeMapLyr})) {
        showMsg("XrProposalFIDService 호출 에러");
    }
}
```

새로운 도형이 생성되면 호출되는 위의 코드를 살펴 보기에 앞서 먼저 이해해야 할 것이 있습니다. 새로운 도형을 생성해서 DBMS 에 저장하기 전에 저장될 도형의 ID 값을 결정해야 합니다. 이 ID 값은 실제로 DBMS 에 해당 도형의 FID 필드값에 저장됩니다. 이 ID 값을 얻기 위해서 1 번 코드에서와 같이 XrProposalFIDService 클래스를 사용합니다. 이 클래스의 run 함수는 targetLayer 에 실제 편집 대상이 되는 수치지도 레이어를 지정해야 하는데, 2 번 코드에서 "bld" 레이어가 해당 레이어입니다. XrProposalFIDService 의 run 이 호출되어 성공적으로 ID 값을 얻어 올 수 있다면 onGetFIDCompleted 함수가 호출되고, 실패하면 onGetFIDError 함수가 호출됩니다. 이 두 함수는 다음과 같습니다.

```
protected function onGetFIDCompleted(fid:uint):void
{
1   _selectMode = false;
2   mx.controls.Alert.show(
        "새로운 도형을 DBMS에 저장하시겠습니까?",
        "확인",
        Alert.YES|Alert.NO|Alert.CANCEL,
        null,
3       function(e:CloseEvent):void {
4           if (e.detail == Alert.YES)
            {
5               var shapeMapLyr:XrShapeMapLayer =
                    map.layers.getLayer("bld") as XrShapeMapLayer;

6               var mashupLyr:XrMashupLayer =
                    map.layers.getLayer("mashup") as XrMashupLayer;

7               var svc:XrUpdateShapeMapFromMashupService =
                    new XrUpdateShapeMapFromMashupService(map.edit, shapeMapLyr,
                        mashupLyr, onSinkCompleted, onSinkError);

8               var params:Object = { targetId:fid, sourceId:0 };

9               if(!svc.run(params))
                {
                    showMsg("편집 서비스 파라메터가 옳바르지 않습니다.");
                }
            }
            else
            {
10              setMapViewMode();
            }
        }
    );
}

protected function onGetFIDError():void
{
    showMsg("XrProposalFIDService 호출 응답 에러");
}
```

신규 도형에 대한 ID 값을 성공적으로 얻을 경우 호출되는 onGetFIDCompleted 함수는 신규 도형을 위한 ID 값이 fid 인자로 넘어옵니다. 이 함수의 내용은 다음과 같습니다.

먼저 1 번에서 _selectMode 를 false 로 지정하여 마우스 클릭으로 다른 도형이 선택되지 않도록 합니다. 그리고 2 번에서 사용자가 그린 새로운 도형을 DBMS 에 저장할 것인지를 묻습니다. 예(YES) 버튼을 클릭했는지를 4 번 코드에서 확인하고, 만약 그렇다면, 5 번과 6 번에서 "bld" 레이어와 "mashup"레이어를 얻습니다. 그리고 7 번 코드에서 XrUpdateShapeMapFromMashupService 클래스의 객체를 생성하여, 이 객체를 이용해 새로운 도형 데이터를 실제 DBMS 에 저장할 수 있도록 준비합니다. 9 번에서 이 XrUpdateShapeMapFromMashupService 클래스의 객체의 run 함수를 호출하기 위해 준비해야할 객체는 8 번 코드에 있는 params 로써 targetId 에는 신규 도형에 대해 얻은 ID 값인 fid 를 지정하고 sourceId 에는 0 을 지정합니다. sourceId 에 0 을 지정한 이유는 앞서 "mashup" 레이어를 통해 폴리곤 항목을 생성할 때 id 값을 0 으로 지정했기 때문입니다. 10 번 코드는 새롭게 그린 도형을 DBMS 에 저장하지 않도록 NO 나 Cancel 버튼을 클릭할 때 호출됩니다. 단지 지도모드로 전환하기 위해서 setMapViewMode 함수를 호출합니다.

5. 선택 도형 삭제

이제 도형을 선택하고 "삭제" 버튼을 클릭하면, 선택된 도형이 실제 DBMS에서 제거되는 기능에 대해 살펴보겠습니다. 선택모드에서 마우스로 도형을 선택하고, "삭제" 버튼을 클릭했을 때 이 "삭제" 버튼에 대한 클릭 이벤트의 코드는 다음과 같습니다.

```
    protected function onDelete(event:MouseEvent):void
    {
1       if(map.edit.select.length == 0) {
            showMsg("삭제할 도형을 선택하지 않았습니다. 삭제할 도형을 먼저 선택하세요.");
            return;
        }

2
3       _selectMode = false;
        mx.controls.Alert.show(
            "선택한 도형을 DBMS에서 삭제하시겠습니까?",
            "확인",
            Alert.YES|Alert.NO|Alert.CANCEL,
            null,
            function(e:CloseEvent) {
5           if (e.detail == Alert.YES)
            {
6               var id:uint = map.edit.select.getFID(0);

7               var shapeMapLyr:XrShapeMapLayer =
                    map.layers.getLayer("bld") as XrShapeMapLayer;

8               var mashupLyr:XrMashupLayer =
                    map.layers.getLayer("mashup") as XrMashupLayer;

9               var svc:XrUpdateShapeMapFromMashupService =
                    new XrUpdateShapeMapFromMashupService(map.edit, shapeMapLyr,
                        mashupLyr, onSinkCompleted, onSinkError);

10              var params:Object = { targetId:id, removal:true  };

11              if(!svc.run(params))
                {
                    showMsg("편집 서비스 파라메터가 옳바르지 않습니다.");
                }
            } else {
12              setMapViewMode();
            }
        });
```

```
}
```

선택된 도형에 대한 "삭제" 버튼의 클릭 이벤트 코드에 대해 살펴보면 다음과 같습니다. 먼저 1번 코드에서 선택된 도형이 있는지를 검사합니다. 만약 선택된 도형이 없다면 메세지 표시와 함께 함수는 종료됩니다. 선택된 도형이 있다면 2번 코드에서 _selectMode를 false로 지정하여 다른 도형이 선택되지 않도록 합니다. 그리고 3번 코드에서 선택된 도형을 DBMS에서 삭제할 것인지를 사용자에게 묻습니다. 만약 사용자가 YES 버튼을 클릭하면 현재 사용자가 선택한 도형의 ID 값을 얻기 위해서 6번 코드를 실행합니다. 7번과 8번은 "bld" 레이어와 "mashup" 레이어의 객체를 얻습니다. 9번은 XrUpdateShapeMapFromMashupService 클래스의 객체를 생성하는데, 이 객체를 이용해 선택된 도형을 실제 DBMS에서 삭제하게 됩니다. 11번 코드에서 XrUpdateShapeMapFromMashupService 클래스의 객체의 run 함수를 호출할 때 인자로써 객체를 주게 됩니다. 이 객체는 10번 코드에서 살펴볼 수 있습니다. targetId에는 삭제할 도형의 ID 값이 저장되어 있고 removal에는 true 값이 저장되어 있는 개체입니다.

6. Undo/Redo

Undo와 Redo는 편집 이력에 대한 되돌리기와 재실행 기능입니다. 이 편집 이력에 대한 Undo와 Redo의 범위는 DBMS에 반영되기 이전까지입니다. DBMS에 반영되면 Undo와 Redo는 되지 않습니다. 이 Undo와 Redo에 대한 버튼의 클릭 이벤트는 다음과 같습니다.

```
private function onRedo(event:MouseEvent):void
{
    map.edit.redo();
}

private function onUndo(event:MouseEvent):void
{
    map.edit.undo();
}
```