## 通过wfs增删改查要素

本节将讲解ol5使用wfs同后台geoserver进行交互，从而实现要素的增删改查。由于geoserver需要服务器端，所以我们将从环境搭建开始讲解，使用公开的数据源，进行查询，修改，添加，删除操作的演示，使得大家最终学会。

### 配置数据源

为了后续进行wfs的各项使用演示，我就使用geoserver官方中使用的数据[nyc\_roads.zip](http://docs.geoserver.org/latest/en/user/_downloads/nyc_roads.zip)（文件夹中已经提供）。这份数据的配置，官网也提供了指导，参见[Publishing a shapefile](http://docs.geoserver.org/latest/en/user/gettingstarted/shapefile-quickstart/index.html)。

下面就以我的计算机配置为例，进行说明：

下载后解压，把压缩包里面的nyc\_roads.dbf, nyc\_roads.prj, nyc\_roads.shp, nyc\_roads.shx放在目录F:\apache-tomcat-8.5.4\webapps\geoserver\data\data\nyc\_roads下，nyc\_roads这个目录没有，就新建一个。

启动tomcat，在浏览器中打开geoserver的配置页面http://localhost:8080/geoserver，使用用户admin登录，密码为geoserver。

创建工作区：

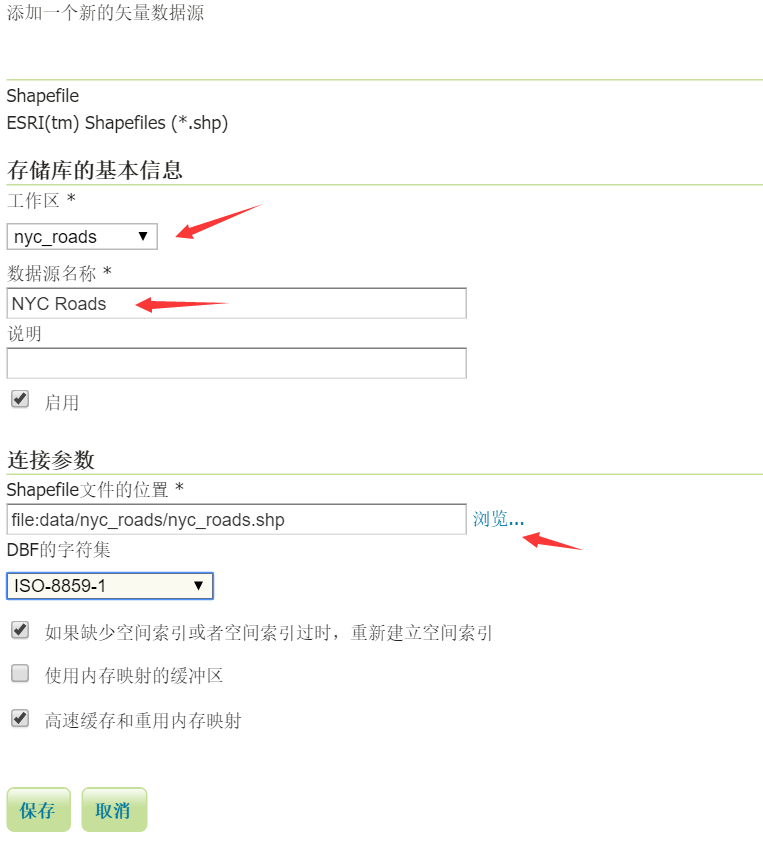




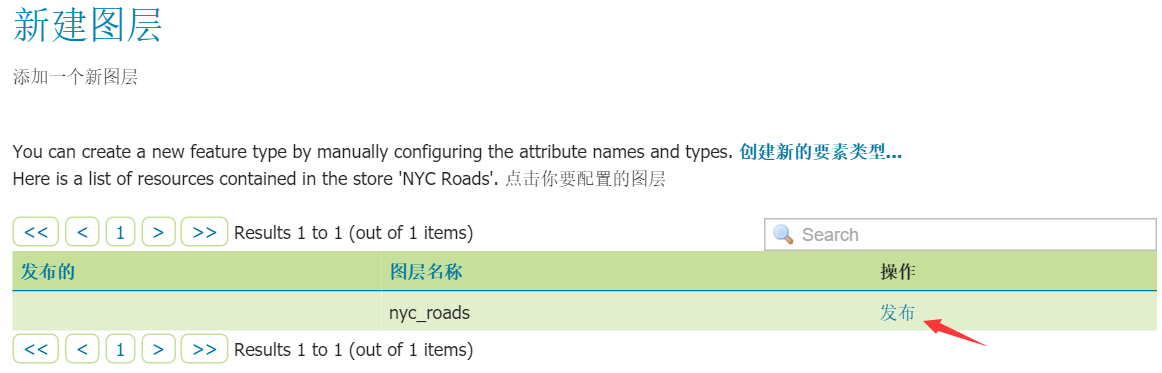
创建数据存储：





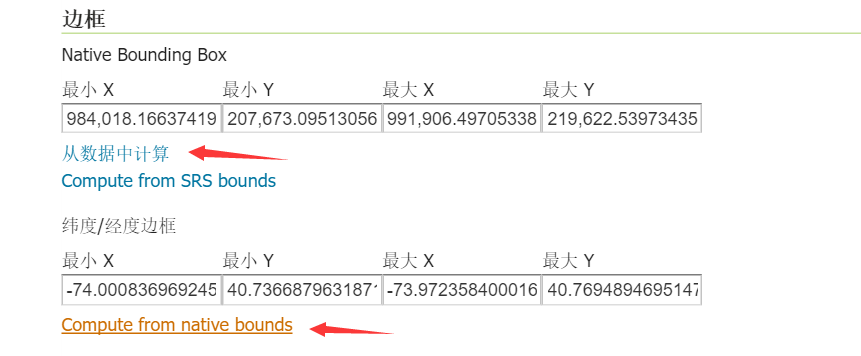


创建图层，数据源配置好后，保存，就出现下面这个界面：



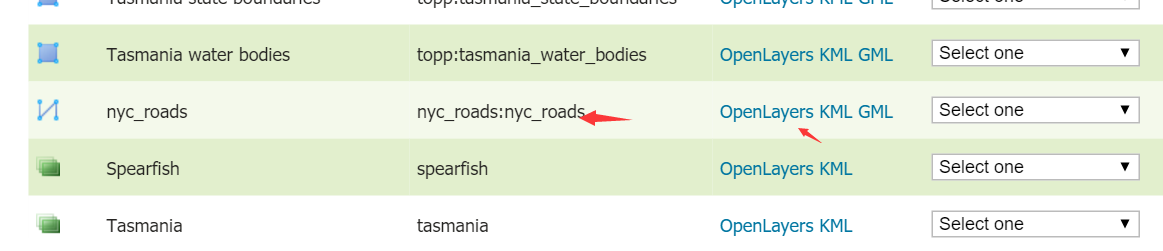
点击发布创建新图层：

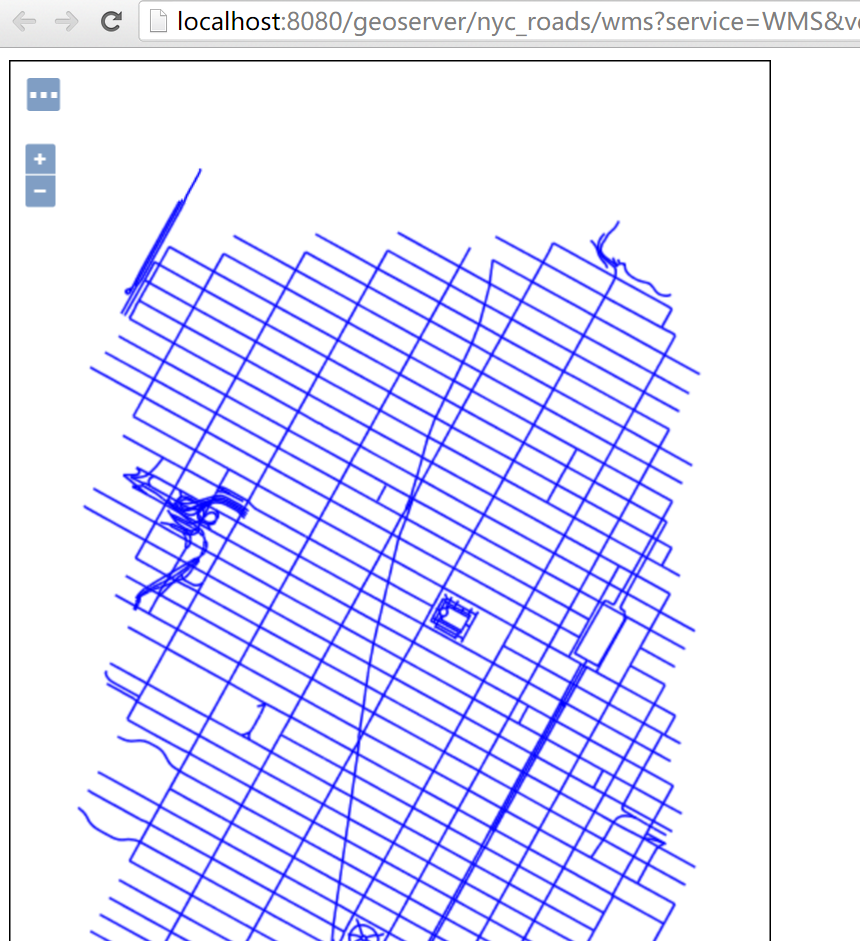




最后点击页面最下方的保存按钮，就配置好了。

最后预览一下，点击左边的Layer Preview，在右边找到刚才创建的图层nyc\_roads:nyc\_roads，点击右边的OpenLayers，就可以打开新页面，显示预览结果。

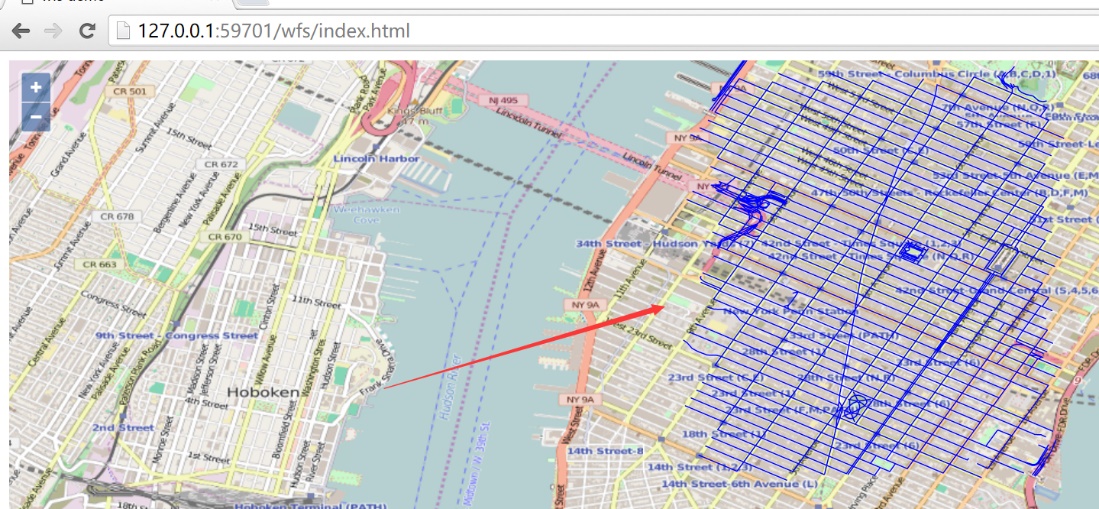




出现最后一个页面，就说明数据源配置好了。

### 2. 通过wfs查询要素

一切都准备好了，现在终于可以通过ol5加载配置好的数据了。上一节中最后的预览结果，大家已经看到了，此处我们自己通过ol5来实现这个预览页面，效果图如下：



对应的代码如下：

<html>

<head>

<meta charset="utf-8">

<title>wfs demo</title>

<link rel="stylesheet" href="v5.3.0-dist/ol.css" type="text/css" />

<script src="v5.3.0-dist/ol.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<div id="map" style="width:100%;height:100%;"></div>

<script>

var vector = new ol.layer.Vector({

source: new ol.source.Vector({

format: new ol.format.GeoJSON(),

url: 'http://localhost:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=nyc\_roads:nyc\_roads&outputFormat=application/json&srsname=EPSG:4326'

}),

style: function (feature, resolution) {

return new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'blue',

width: 1

})

});

}

});

var map = new ol.Map({

layers: [new ol.layer.Tile({

source: new ol.source.OSM()

}), vector],

target: 'map',

view: new ol.View({

center: [-73.99710639567148, 40.742270050255556],

maxZoom: 19,

zoom: 14,

projection: 'EPSG:4326'

})

});

</script>

</body>

</html>

和一般的矢量地图加载没什么两样，对于wfs而言，需要弄明白的是ol.source.Vector的url参数：

<http://localhost:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=nyc_roads:nyc_roads&outputFormat=application/json&srsname=EPSG:4326>。 如果对wfs协议不太清楚的，建议优先看一下geoserver的官网文档[WFS](http://docs.geoserver.org/stable/en/user/services/wfs/index.html)。

WFS详解

<https://docs.geoserver.org/stable/en/user/services/wfs/reference.html>

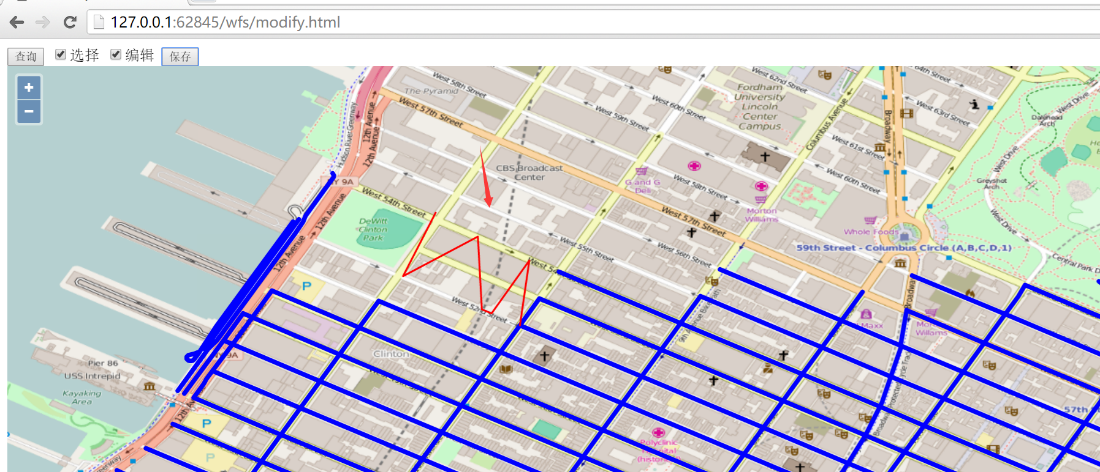
此处我们要获取所有的要素，所以设置request=GetFeature，typeNames的值设置为nyc\_roads:nyc\_roads，是因为我们之前配置的图层命名如此，可以在geoserver管理页面的Layer Preview里面看对应图层的Name。 对于outputFormat和srsname（Spatial Reference System，简称 SRS，空间参照系）就不做过多解释，大家看值就容易明白了。因为view设置的projection: 'EPSG:4326'，所以此处设置srsname=EPSG:4326。

BTW: 上面是全部查询，我们知道wfs也支持filter，所以我们可以在url里面设置filter，从而实现更细粒度的查询，比如这样：

http://localhost:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=nyc\_roads:nyc\_roads&outputFormat=application/json&srsname=EPSG:4326&cql\_filter=in ('nyc\_roads.1162')。 大家可以自行用这个url进行测试一下。关于filter更详细的信息参见[Supported filter languages](http://docs.geoserver.org/latest/en/user/filter/syntax.html)。 因为这不是教程的重点，所以此处不进行细说。

### 通过wfs修改要素

在查询的基础上，本小节我们更进一步对界面上呈现的要素进行修改，然后通过wfs协议保存到服务器端。界面效果如下:



在界面上方，先点击查询按钮，通过wfs协议把所有的要素查询到前端界面上显示，然后选择复选框编辑，就可以选择界面上的线段，进行编辑，比如把直线编辑成W形状，然后点击按钮保存，就可以把编辑后的线段保存下来。

在点击保存按钮之前，请打开开发者面板，然后再点击保存，之后就可以看到发起了一个wfs的http请求到geoserver服务器，请求的url为：

http://localhost:8080/geoserver/wfs?service=wfs，发送的内容大致为:

<Transaction xmlns="http://www.opengis.net/wfs" service="WFS" version="1.1.0" xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance> xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">

<Update typeName="feature:nyc\_roads" mlns:feature="http://geoserver.org/nyc\_roads">

<Property>

<Name>the\_geom</Name>

<Value>

<MultiLineString xmlns="http://www.opengis.net/gml" srsName="EPSG:4326">

<lineStringMember>

<LineString srsName="EPSG:4326">

<posList>-73.98528635501862 40.768332481384284 -73.98608637 40.76719342 -73.98449242115021 40.767849683761604 -73.98447096347809 40.76647639274598 -73.98299038410187 40.767334699630744 -73.98336657 40.76604531</posList>

</LineString>

</lineStringMember>

</MultiLineString>

</Value>

</Property>

<Property>

<Name>modified</Name>

<Value>5/28/2001</Value>

</Property>

<Property>

<Name>name</Name>

<Value>W 56 ST</Value>

</Property>

<Property>

<Name>vsam</Name>

<Value>15060</Value>

</Property>

<Property>

<Name>sourcedate</Name>

<Value>3/31/1996</Value>

</Property>

<Property>

<Name>sourcetype</Name>

<Value>Photogrammetric</Value>

</Property>

<Property>

<Name>source\_id</Name>

<Value>96083</Value>

</Property>

<Property>

<Name>borough</Name>

<Value>Manhattan</Value>

</Property>

<Property>

<Name>feat\_code</Name>

<Value>2900</Value>

</Property>

<Property>

<Name>feat\_desc</Name>

<Value>Paved Road</Value>

</Property>

<Property>

<Name>exported</Name>

<Value>05/19/2004</Value>

</Property>

<Property>

<Name>feat\_type</Name>

<Value>0</Value>

</Property>

<Filter xmlns="http://www.opengis.net/ogc">

<FeatureId fid="nyc\_roads.882" />

</Filter>

</Update>

</Transaction>

如果保存成功，则response的内容大致如下：

<?xml version="1.0" encoding="UTF-8" ?>

<wfs:TransactionResponse xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:sf="http://www.openplans.org/spearfish"

xmlns:wfs="http://www.opengis.net/wfs"

xmlns:gml="http://www.opengis.net/gml"

xmlns:nyc\_roads="http://geoserver.org/nyc\_roads"

xmlns:ogc="http://www.opengis.net/ogc"

xmlns:ows="http://www.opengis.net/ows"

xmlns:tiger="http://www.census.gov"

xmlns:topp="http://www.openplans.org/topp"

xmlns:xlink="http://www.w3.org/1999/xlink" x

mlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

version="1.1.0"

xsi:schemaLocation="http://www.opengis.net/wfs http://localhost:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">

<wfs:TransactionSummary>

<wfs:totalInserted>0</wfs:totalInserted>

<wfs:totalUpdated>1</wfs:totalUpdated>

<wfs:totalDeleted>0</wfs:totalDeleted>

</wfs:TransactionSummary>

<wfs:TransactionResults />

<wfs:InsertResults>

<wfs:Feature><ogc:FeatureId fid="none" /></wfs:Feature>

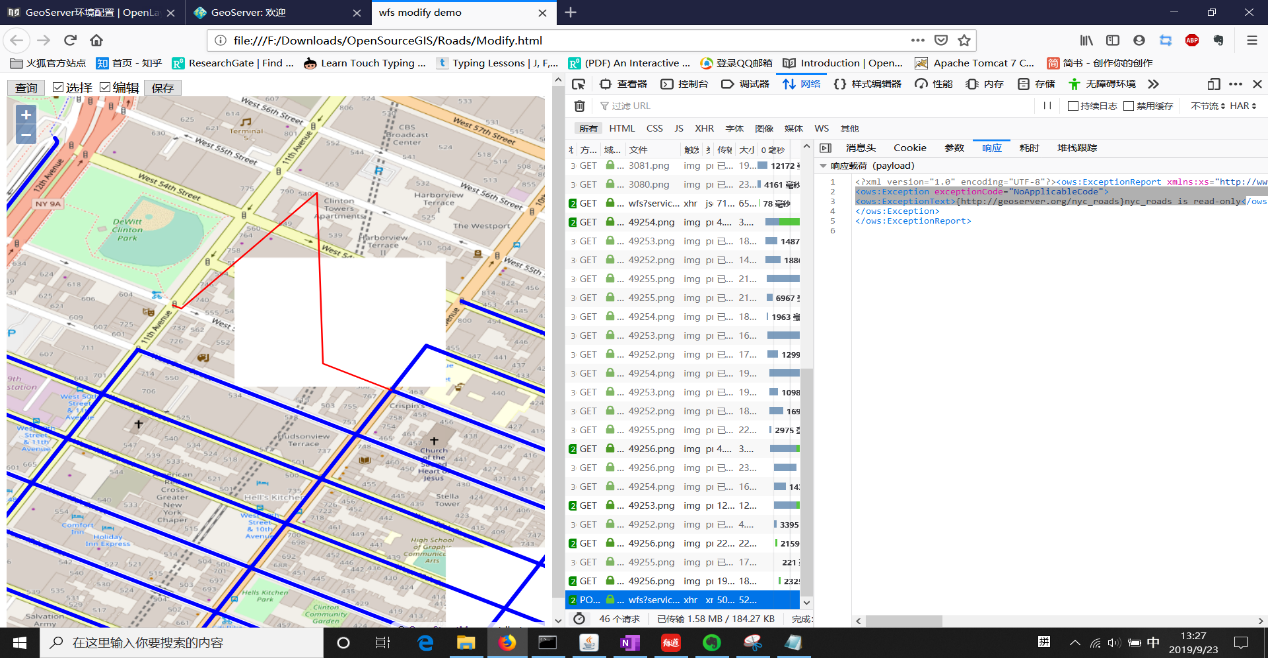
</wfs:InsertResults>

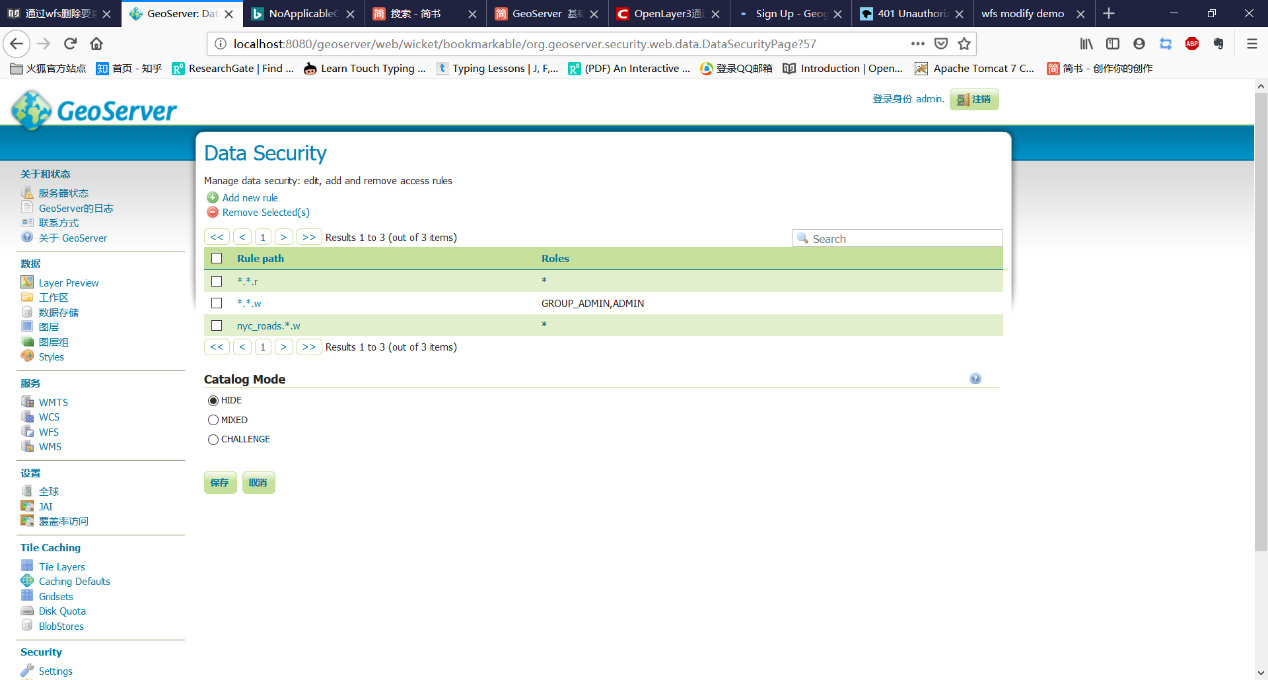
</wfs:TransactionResponse>

我的报错了：

<ows:Exception exceptionCode="NoApplicableCode">

<ows:ExceptionText>{http://geoserver.org/nyc\_roads}nyc\_roads is read-only</ows>





重新刷新页面后，再次点击查询按钮，可以验证之前修改的线段是否修改成功。由于不能提供geoserver服务器，所以只能让大家自行用下面的代码在本地验证：

<html>

<head>

<meta charset="utf-8">

<title>wfs modify demo</title>

<link rel="stylesheet" href="v5.3.0-dist/ol.css" type="text/css" />

<script src="v5.3.0-dist/ol.js" type="text/javascript" charset="utf-8"></script>

<script src="v5.3.0-dist/zepto.min.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<input type="button" value="查询" onclick="queryWfs();" />

<input id="select" type="checkbox" value="select" />选择

<input id="modify" type="checkbox" value="modify" />编辑

<input id="save" type="button" value="保存" onclick="onSave();" />

<div id="map" style="width:100%;height:100%;"></div>

<script>

var wfsVectorLayer = null;

var modifiedFeatures = null;

// 选择器

var selectInteraction = new ol.interaction.Select({

style: new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'red',

width: 2

})

})

});

// 修改器

var modifyInteraction = new ol.interaction.Modify({

style: new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'red',

width: 5

})

}),

features: selectInteraction.getFeatures()

});

modifyInteraction.on('modifyend', function(e) {

// 把修改完成的feature暂存起来

modifiedFeatures = e.features;

});

var map = new ol.Map({

layers: [new ol.layer.Tile({

source: new ol.source.OSM()

})],

target: 'map',

view: new ol.View({

center: [-73.99710639567148, 40.742270050255556],

maxZoom: 19,

zoom: 13,

projection: 'EPSG:4326'

})

});

// 通过wfs查询所有的要素

function queryWfs() {

// 支持重新查询

if (wfsVectorLayer) {

map.removeLayer(wfsVectorLayer);

}

// 创建新的图层来加载wfs的要素

wfsVectorLayer = new ol.layer.Vector({

source: new ol.source.Vector({

format: new ol.format.GeoJSON({

geometryName: 'the\_geom' // 因为数据源里面字段the\_geom存储的是geometry，所以需要指定

}),

url: 'http://localhost:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=nyc\_roads:nyc\_roads&outputFormat=application/json&srsname=EPSG:4326'

}),

style: function(feature, resolution) {

return new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'blue',

width: 5

})

});

}

});

map.addLayer(wfsVectorLayer);

}

$('#select').change(function() {

if (this.checked) {

// 勾选选择复选框时，添加选择器到地图

map.removeInteraction(selectInteraction);

map.addInteraction(selectInteraction);

} else {

// 不勾选选择复选框的情况下，移出选择器和修改器

map.removeInteraction(selectInteraction);

document.getElementById('modify').checked = false;

map.removeInteraction(modifyInteraction);

modifiedFeatures = null;

}

});

$('#modify').change(function() {

if (this.checked) {

// 勾选修改复选框时，添加选择器和修改器到地图

document.getElementById('select').checked = true;

map.removeInteraction(modifyInteraction);

map.addInteraction(modifyInteraction);

map.removeInteraction(selectInteraction);

map.addInteraction(selectInteraction);

} else {

// 不勾选修改复选框时，移出修改器

map.removeInteraction(modifyInteraction);

modifiedFeatures = null;

}

});

// 保存已经编辑的要素

function onSave() {

if (modifiedFeatures && modifiedFeatures.getLength() > 0) {

// 转换坐标

var modifiedFeature = modifiedFeatures.item(0).clone();

// 注意ID是必须，通过ID才能找到对应修改的feature

modifiedFeature.setId(modifiedFeatures.item(0).getId());

// 调换经纬度坐标，以符合wfs协议中经纬度的位置

modifiedFeature.getGeometry().applyTransform(function(flatCoordinates, flatCoordinates2, stride) {

for (var j = 0; j < flatCoordinates.length; j += stride) {

var y = flatCoordinates[j];

var x = flatCoordinates[j + 1];

flatCoordinates[j] = x;

flatCoordinates[j + 1] = y;

}

});

modifyWfs([modifiedFeature]);

}

}

// 把修改提交到服务器端

function modifyWfs(features) {

var WFSTSerializer = new ol.format.WFS();

var featObject = WFSTSerializer.writeTransaction(null,

features, null, {

featureType: 'nyc\_roads',

featureNS: 'http://geoserver.org/nyc\_roads', // 注意这个值必须为创建工作区时的命名空间URI

srsName: 'EPSG:4326'

});

// 转换为xml内容发送到服务器端

var serializer = new XMLSerializer();

var featString = serializer.serializeToString(featObject);

var request = new XMLHttpRequest();

request.open('POST', 'http://localhost:8080/geoserver/wfs?service=wfs');

// 指定内容为xml类型

request.setRequestHeader('Content-Type', 'text/xml');

request.send(featString);

}

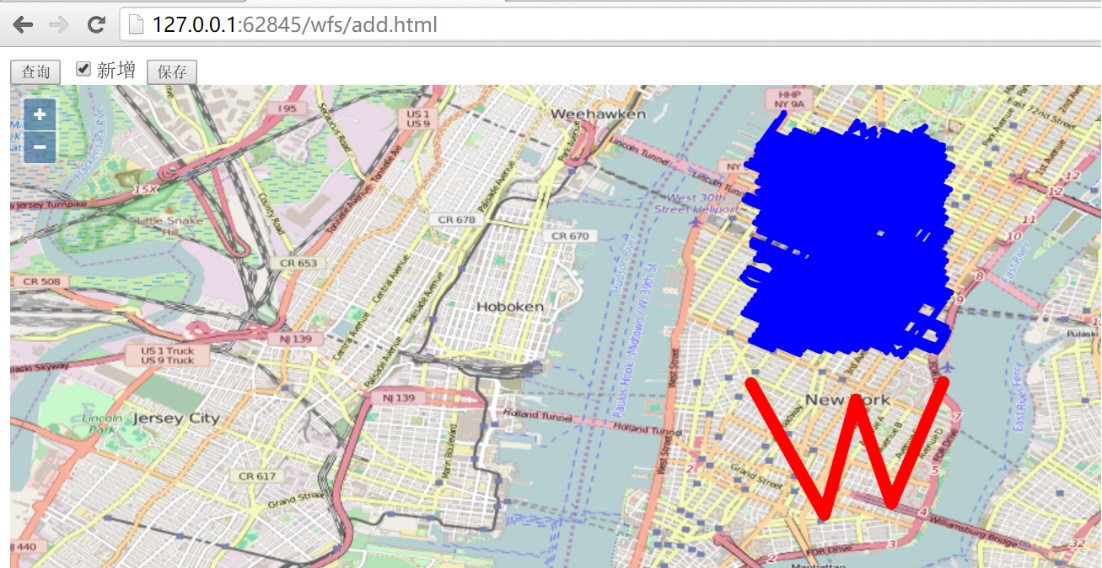
</script>

</body>

</html>

### 4. 通过wfs添加要素

现在我们该介绍一下如何在前端绘制一个新的要素，并且保存到服务器端。还是先看一下界面：



勾选新增复选框之后，就可以在界面上绘制新的线段，如图，绘制了一个W形状的线，绘制完成后，点击按钮保存就可以把界面上新增的线保存到服务器端，在开发者工具界面可以看到http请求: <http://localhost:8080/geoserver/wfs?service=wfs，请求发送的内容为>：

<Transaction xmlns="http://www.opengis.net/wfs" service="WFS" version="1.1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">

<Insert>

<nyc\_roads xmlns="http://geoserver.org/nyc\_roads" fid="nyc\_roads.new.1">

<the\_geom>

<MultiLineString xmlns="http://www.opengis.net/gml" srsName="EPSG:4326">

<lineStringMember>

<LineString srsName="EPSG:4326">

<posList>-73.99970054626465 40.732669830322266 -73.98974418640137 40.71481704711914 -73.98545265197754 40.730438232421875 -73.98064613342285 40.71636199951172 -73.97360801696777 40.73284149169922</posList>

</LineString>

</lineStringMember>

</MultiLineString>

</the\_geom>

<name>nyc\_roads.new.1</name>

<modified>nyc\_roads.new.1</modified>

<vsam>0</vsam>

<sourcedate></sourcedate>

<sourcetype></sourcetype>

<source\_id>1</source\_id>

<borough></borough>

<feat\_code>0</feat\_code>

<feat\_desc>11</feat\_desc>

<feat\_type>0</feat\_type>

<exported>true</exported>

</nyc\_roads>

</Insert>

</Transaction>

服务器端response的内容为：

<?xml version="1.0" encoding="UTF-8" ?>

<wfs:TransactionResponse xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sf="http://www.openplans.org/spearfish"

xmlns:wfs="http://www.opengis.net/wfs"

xmlns:gml="http://www.opengis.net/gml"

xmlns:nyc\_roads="http://geoserver.org/nyc\_roads"

xmlns:ogc="http://www.opengis.net/ogc"

xmlns:ows="http://www.opengis.net/ows"

xmlns:tiger="http://www.census.gov"

xmlns:topp="http://www.openplans.org/topp"

xmlns:xlink="http://www.w3.org/1999/xlink"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

version="1.1.0"

xsi:schemaLocation="http://www.opengis.net/wfs http://localhost:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">

<wfs:TransactionSummary>

<wfs:totalInserted>1</wfs:totalInserted>

<wfs:totalUpdated>0</wfs:totalUpdated>

<wfs:totalDeleted>0</wfs:totalDeleted>

</wfs:TransactionSummary>

<wfs:TransactionResults />

<wfs:InsertResults>

<wfs:Feature>

<ogc:FeatureId fid="new0" />

</wfs:Feature>

</wfs:InsertResults>

</wfs:TransactionResponse>

通过再次刷新界面，点击查询按钮查看所有的feature，可以确认是否添加成功，请自行验证。下面给出对应的代码：

<html>

<head>

<title>wfs add demo</title>

<meta charset="utf-8">

<link rel="stylesheet" href="v5.3.0-dist/ol.css" type="text/css" />

<script src="v5.3.0-dist/ol.js" type="text/javascript" charset="utf-8"></script>

<script src="v5.3.0-dist/zepto.min.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<input type="button" value="查询" onclick="queryWfs();" />

<input id="add" type="checkbox" value="add" />新增

<input id="saveNew" type="button" value="保存" onclick="onSaveNew();" />

<div id="map" style="width:100%;height:100%;"></div>

<script>

var newId = 1;

var wfsVectorLayer = null;

var drawedFeature = null;

// 创建用于新绘制feature的layer

var drawLayer = new ol.layer.Vector({

source: new ol.source.Vector(),

style: new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'blue',

width: 5

})

})

});

// 添加绘制新图形的interaction，用于添加新的线条

var drawInteraction = new ol.interaction.Draw({

type: 'LineString', // 设定为线条

style: new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'red',

width: 10

})

}),

source: drawLayer.getSource()

});

drawInteraction.on('drawend', function (e) {

// 绘制结束时暂存绘制的feature

drawedFeature = e.feature;

});

var map = new ol.Map({

layers: [new ol.layer.Tile({

source: new ol.source.OSM()

}), drawLayer],

target: 'map',

view: new ol.View({

center: [-73.99710639567148, 40.742270050255556],

maxZoom: 19,

zoom: 13,

projection: 'EPSG:4326'

})

});

function queryWfs() {

if (wfsVectorLayer) {

map.removeLayer(wfsVectorLayer);

}

wfsVectorLayer = new ol.layer.Vector({

source: new ol.source.Vector({

format: new ol.format.GeoJSON({

geometryName: 'the\_geom'

}),

url: 'http://localhost:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=nyc\_roads:nyc\_roads&outputFormat=application/json&srsname=EPSG:4326'

}),

style: function (feature, resolution) {

return new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'blue',

width: 5

})

});

}

});

map.addLayer(wfsVectorLayer);

}

$('#add').change(function () {

if (this.checked) {

// 勾选新增复选框时，添加绘制的Interaction

map.removeInteraction(drawInteraction);

map.addInteraction(drawInteraction);

} else {

// 取消勾选新增复选框时，移出绘制的Interaction，删除已经绘制的feature

map.removeInteraction(drawInteraction);

if (drawedFeature) {

drawLayer.getSource().removeFeature(drawedFeature);

}

drawedFeature = null;

}

});

// 保存新绘制的feature

function onSaveNew() {

// 转换坐标

var geometry = drawedFeature.getGeometry().clone();

geometry.applyTransform(function (flatCoordinates, flatCoordinates2, stride) {

for (var j = 0; j < flatCoordinates.length; j += stride) {

var y = flatCoordinates[j];

var x = flatCoordinates[j + 1];

flatCoordinates[j] = x;

flatCoordinates[j + 1] = y;

}

});

// 设置feature对应的属性，这些属性是根据数据源的字段来设置的

var newFeature = new ol.Feature();

newFeature.setId('nyc\_roads.new.' + newId);

newFeature.setGeometryName('the\_geom');

newFeature.set('the\_geom', null);

newFeature.set('name', newFeature.getId());

newFeature.set('modified', newFeature.getId());

newFeature.set('vsam', 0);

newFeature.set('sourcedate', '');

newFeature.set('sourcetype', '');

newFeature.set('source\_id', newId);

newFeature.set('borough', '');

newFeature.set('feat\_code', 0);

newFeature.set('feat\_desc', '11');

newFeature.set('feat\_type', 0);

newFeature.set('exported', 'true');

newFeature.setGeometry(new ol.geom.MultiLineString([geometry.getCoordinates()]));

addWfs([newFeature]);

// 更新id

newId = newId + 1;

// 3秒后，自动刷新页面上的feature

setTimeout(function () {

drawLayer.getSource().clear();

queryWfs();

}, 3000);

}

// 添加到服务器端

function addWfs(features) {

var WFSTSerializer = new ol.format.WFS();

var featObject = WFSTSerializer.writeTransaction(features,

null, null, {

featureType: 'nyc\_roads',

featureNS: 'http://geoserver.org/nyc\_roads',

srsName: 'EPSG:4326'

});

var serializer = new XMLSerializer();

var featString = serializer.serializeToString(featObject);

var request = new XMLHttpRequest();

request.open('POST', 'http://localhost:8080/geoserver/wfs?service=wfs');

request.setRequestHeader('Content-Type', 'text/xml');

request.send(featString);

}

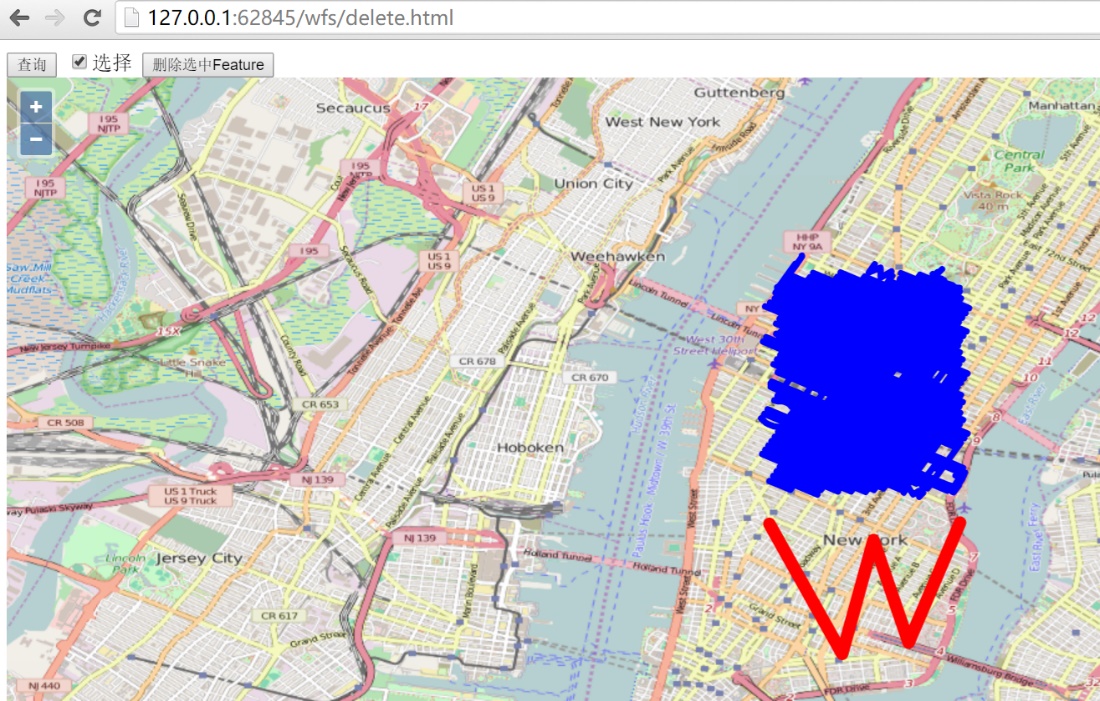
</script>

</body>

</html>

### 5. 通过wfs删除要素

删除feature是wfs协议中的最后一个例子了，和之前的修改，添加差不多，大同小异。还是先看界面：



选择查询按钮，把所有的feature加载到前端，然后勾选选择复选框，就可以在地图上选择要删除的feature，图示选择之前添加的W形状的线条，然后点击删除选中Feature按钮，就可以把feature删除掉。

在开发者工具窗口里面可以看到删除feature的http请求： <http://localhost:8080/geoserver/wfs?service=wfs，其发送的内容为>：

<Transaction xmlns="http://www.opengis.net/wfs" service="WFS" version="1.1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">

<Delete typeName="feature:nyc\_roads" xmlns:feature="http://geoserver.org/nyc\_roads">

<Filter xmlns="http://www.opengis.net/ogc">

<FeatureId fid="nyc\_roads.1302" />

</Filter>

</Delete>

</Transaction>

服务器端response的内容为：

<?xml version="1.0" encoding="UTF-8" ?>

<wfs:TransactionResponse xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sf="http://www.openplans.org/spearfish" xmlns:wfs="http://www.opengis.net/wfs" xmlns:gml="http://www.opengis.net/gml" xmlns:nyc\_roads="http://geoserver.org/nyc\_roads" xmlns:ogc="http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows" xmlns:tiger="http://www.census.gov" xmlns:topp="http://www.openplans.org/topp" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs http://localhost:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">

<wfs:TransactionSummary>

<wfs:totalInserted>0</wfs:totalInserted>

<wfs:totalUpdated>0</wfs:totalUpdated>

<wfs:totalDeleted>1</wfs:totalDeleted>

</wfs:TransactionSummary>

<wfs:TransactionResults />

<wfs:InsertResults>

<wfs:Feature>

<ogc:FeatureId fid="none" />

</wfs:Feature>

</wfs:InsertResults>

</wfs:TransactionResponse>

通过再次刷新查询，可以确认刚才的feature是否成功删除。请自行验证，对应实例的代码如下：

<html>

<head>

<title>wfs crud demo</title>

<meta charset="utf-8">

<link rel="stylesheet" href="v5.3.0-dist/ol.css" type="text/css" />

<script src="v5.3.0-dist/ol.js" type="text/javascript" charset="utf-8"></script>

<script src="v5.3.0-dist/zepto.min.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<input type="button" value="查询" onclick="queryWfs();" />

<input id="select" type="checkbox" value="select" />选择

<input id="delete" type="button" value="删除选中Feature" onclick="onDeleteFeature();" />

<div id="map" style="width:100%;height:100%;"></div>

<script>

var wfsVectorLayer = null;

// 选择器

var selectInteraction = new ol.interaction.Select({

style: new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'red',

width: 10

})

})

});

var map = new ol.Map({

layers: [new ol.layer.Tile({

source: new ol.source.OSM()

})],

target: 'map',

view: new ol.View({

center: [-73.99710639567148, 40.742270050255556],

maxZoom: 19,

zoom: 13,

projection: 'EPSG:4326'

})

});

function queryWfs() {

if (wfsVectorLayer) {

map.removeLayer(wfsVectorLayer);

}

wfsVectorLayer = new ol.layer.Vector({

source: new ol.source.Vector({

format: new ol.format.GeoJSON({

geometryName: 'the\_geom'

}),

url: 'http://localhost:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=nyc\_roads:nyc\_roads&outputFormat=application/json&srsname=EPSG:4326'

}),

style: function (feature, resolution) {

return new ol.style.Style({

stroke: new ol.style.Stroke({

color: 'blue',

width: 5

})

});

}

});

map.addLayer(wfsVectorLayer);

}

$('#select').change(function () {

if (this.checked) {

map.removeInteraction(selectInteraction);

map.addInteraction(selectInteraction);

} else {

map.removeInteraction(selectInteraction);

}

});

function onDeleteFeature() {

// 删选择器选中的feature

if (selectInteraction.getFeatures().getLength() > 0) {

deleteWfs([selectInteraction.getFeatures().item(0)]);

// 3秒后自动更新features

setTimeout(function () {

selectInteraction.getFeatures().clear();

queryWfs();

}, 3000);

}

}

// 在服务器端删除feature

function deleteWfs(features) {

var WFSTSerializer = new ol.format.WFS();

var featObject = WFSTSerializer.writeTransaction(null,

null, features, {

featureType: 'nyc\_roads',

featureNS: 'http://geoserver.org/nyc\_roads',

srsName: 'EPSG:4326'

});

var serializer = new XMLSerializer();

var featString = serializer.serializeToString(featObject);

var request = new XMLHttpRequest();

request.open('POST', 'http://localhost:8080/geoserver/wfs?service=wfs');

request.setRequestHeader('Content-Type', 'text/xml');

request.send(featString);

}

</script>

</body>

</html>