隧道动态检测系统后台开发文档

**[环境准备]**

**MySql动态库配置:**

MySql.Data.dll导入项目文件夹并添加引用

若运行提示版本错误，可选择在NuGet包管理中重新安装对应版本

**Json解析库安装：**

VS2019/VS2017 工具 >> NuGet包管理器 >> 程序包管理器控制台

>> 执行 Install-Package Newtonsoft.Json

**数据库接口导入：**

将MetroTunnelDB.cs文件导入项目

**[接口说明]**

1. **轨交线路接口**

**接口文件：**LineHandler.ashx，LineHandler.ashx.cs

**说明：**提供轨交线路的数据

**返回数据:**

**success：**

{

data:{

lines:[

{lineId:1,name:’1号线’},{ lineId:2,name:’2号线’}...

]

}

}

**fail:** null

1. **检测时间接口**

**接口文件：**DetectTimeHandler.ashx, DetectTimeHandler.ashx.cs

**说明：**提供所选轨交线路所有有采集数据的日期列表

**输入参数：**

{

lineId:1 //线路编号

}

**返回数据：**

**success：**

{

data:{

dates:[

‘2018-01-02’, ‘2018-05-17’, ‘2019-09-22’...

]

}

}

**fail:** null

1. **设备接口**

**接口文件：**DeviceHandler.ashx, DeviceHandler.ashx.cs

**说明：**提供所选轨交线路和日期对应的有采集数据的设备列表

**输入参数：**

{

lineId:1, //线路编号

date:’ 2019-09-22’ //检测日期

}

**返回数据：**

**success：**

{

data:{

devices:[

{deviceId:1,name:’设备1’},{id:2,name:’设备2’}...

]

}

}

**fail:** null

1. **检测记录接口**

**接口文件：**DetectRecordHandler.ashx, DetectRecordHandler.ashx.cs

**说明：**根据线路号和日期提供检测记录号及检测记录信息

**输入参数：**

{

lineId:1,//线路编号

date:’2019-09-22’ //检测日期

}

**返回数据：**

**success:**

{

data:{

records:[

{catchId:1,deviceId:1,totallength:15000,recordsCount:150000},

{catchId:2,deviceId:1,totallength:26000,recordsCount:350000},

]

}

}

**fail:** null

1. **检测数据接口**

**接口文件：**DetectDataHandler.ashx, DetectDataHandler.ashx.cs

**说明：**根据线路号和日期提供采集的数据

**输入参数：**

{

catchId:1, //采集数据编号

queryStart:0, //待查询数据起始行

queryNum:500 //待查询数据条数

}

**返回数据：**

**success:**

{

data:{

records:[

//各个距离检测点的数据，包括长轴，水平轴，短轴，短轴仰角，是否收敛，是否裂缝

{id:1,distance:100,longAxis:25,horizontalAxis:23,shortAxis:12,shortAxisAngle:30,hasConstr:0,hasCrack:0}, {id:2,distance:200,longAxis:26,horizontalAxis:24,shortAxis:13,shortAxisAngle:32,hasConstr:0,hasCrack:1},

{id:3,distance:300,longAxis:25,horizontalAxis:23,shortAxis:12,shortAxisAngle:30,hasConstr:1,hasCrack:0},

…

]

}

}

**fail:** null

1. **二维三维显示数据接口**

**接口文件：**DisplayDataHandler.ashx, DisplayDataHandler.ashx.cs

**说明：**通过给定的采集数据编号提供整条隧道的采集数据，二维坐标系x横轴z纵轴，三维坐标系x横轴z纵轴y距离，正常点蓝色显示，病害点红色显示。(或者通过给定的距离或者距离编号提供相应的采集数据)。由于数据数量较多，在后台时就就对数据格式进行封装，前台接收数据后不再进行二次加工处理，便于前台加载和显示，提高客户界面友好性。

**输入参数：**

{

catchId:1,

queryStart:0, //待查询数据起始行

queryNum:500 //待查询数据条数

}

**返回数据：**

**success:**

{

data:{

records:[//具体各个距离检测点的数据

{dpNo:1,value:[x,y,z],itemStyle:{color:’blue’}},

{ dpNo:1,value:[x,y,z],itemStyle:{color:’blue’}},

{ dpNo:2,value:[x,y,z],itemStyle:{color:’red}},

…

**注：dpNo是里程位置，value是三维坐标，x是轨道方向径坐标（里程位置），y是水平方向横坐标，z是垂直方向纵坐标**

]

}

}

**fail:** null

1. **收敛点数据接口**

**接口文件：**ConstrictLocHandler.ashx, ConstrictLocHandler.ashx.cs

**说明：**给距离控件提供收敛数据

**输入参数：**

{

catchId:1

}

**返回数据：**

**success:**

{

data:{

recordsCount:1500,//收敛点数据的条数

records:[//具体各个收敛点的距离数据

{ dpNo:3,distance:100},

{ dpNo:12,distance:230},

{ dpNo:46,distance:503},

…

]

}

}

**fail:** null

1. **裂缝点数据接口**

**接口文件：**CrackLocHandler.ashx, CrackLocHandler.ashx.cs

**说明：**给距离控件提供裂缝数据

**输入参数：**

{

catchId:1

}

**返回数据：**

**success:**

{

data:{

recordsCount:1500,//裂缝点数据的条数

records:[//具体各个裂缝点的距离数据

{ dpNo:3,distance:100},

{ dpNo:12,distance:230},

{ dpNo:46,distance:503},

…

]

}

}

**fail:** null