进入地图展示界面：

如果是新建地图，则不需要发起请求，执行空初始化。

如果是加载已有地图，则向服务器出请求，获得所有前端需要的数据，包括但不仅限于：

1.地图自身信息。

2.所需加载的图层list及其样式表。

3.所需加载图层的图层信息（数据信息，绘图信息）。

将以上信息以合适的数据结构在前端设计变量并且存储。

以上是页面加载阶段，这时候会在loading吧。

\*前端全局变量的构造说明:

function Map; Map对象;

//Map对象包含地图的基本信息，并且包含一个Layer的Array，同时提供Array的增删接口。

function Layer; Layer对象

包含所有绘图AB组需要的绘图数据。

上面的阶段会封装在一个**init()**函数里面；

然后则是调用**Display()** 进行对echarts&mapv的绘制框架的初始化，并且初始化时填入上面的数据。同时**Displa()** 中尽可能把echarts&mapv的初始化和数据绘制的部分剥离（但是由于某种限制，比较难），将能被剥离出来的部分写入**redraw()**函数，比如Echars的Series的构造和赋值，bmap样式的修改等，这样便于后面的操作

然后，所有的动态操作（也需要写到对应的函数里）均对这些变量进行操作，完成样式、图层的修改需求，并且修改后调用**redraw()**

而为了适应以上框架，就需要对mapv的操作也进行封装，可以针对其写一个**draw()**函数，管理其较为简单的图层管理工作。（绘制时是直接的

**var** mapvLayer = **new** mapv.baiduMapLayer(mymap, dataSet, options);操作 而做出了修改或着移除的时候就需要先把这个Layer从baiduMap上调用相关的remove函数再进行重绘，这点上，会自动对参数变化做出反应的echarts要优秀和省心的多）

而执行地图保存的时候，则是将所有信息写入数据库。Update地图表的信息，update原有的地图图层list的信息并且对地图的新增图层做Insert操作（因此saveMap的Controller第一步的操作就是对保存前后地图所拥有的图层list的更改情况作对比，以便调用对应的service进行操作）。

这里后台需要构造一个MapInfo类，用以映射前端的所有信息（或者也可以做一个适当拆分）

（MapInfo应该是这么个东西，里面是Map的基础信息和一个MapLayer的数组，从基本上，保存地图是不会涉及到Layers表的读入的（Layers在add完毕后在目前的基础上是只读的，或者可以由管理者删除））

MapInfo{

Map map;

MapLayer[] layerlist;

}

Public ModelAndView saveMap(

MapInfo map

)

{

1.调用服务 获取当前数据库中的MapLayer的List;

2.写一个函数来进行比对，并且分类调用不同的List来执行MapLayer表的更新、删除、插入作业;

3.调用update地图信息的操作。

}

而另存为的操作类似，不过因为是新建的地图，所以不需要做MapLayer的List比对工作。\*新建的地图的保存调用另存为的操作