首先，发送请求/train/trainModel到服务器

服务器处理请求，获取相应训练数据（从数据库获取评价指标得分）

将数据写入文本，存放在硬盘

调用MyModelTrain的train方法，将其数据路径传入，训练网络，

MyModelTrain. train (){

创建DataSet 在DataSet中有一个集合来保从文本中获取的数据集，集合中存放的是一个一个的Record

Record是DataSet的一个内部类，其中包含两个属性：attrs来保存网络输入层的值 lable来保存预期值

创建BPNetwork（构造方法中）初始化网络层与节点数（设置网络的层数，以及每一层网络的节点数）

调用 BPNetwork.trainModel()训练网络

在这个方法里面，首先调用BPNetwork.initLayer()初始化网络。

创建NeuralLayer对象，随机设置网络权重，偏置

接着创建Listener对象，调用start()方法，启动监听网络训练（本类的一个内部类，开启不同的线程来执行训练，并判断其达到目标值时，终止训练）

调用BPNetwork .test()方法计算正确率

获取每一行训练数据， 计算训练数据的值

计算与预期值之间的差，然后累加到sum

sum除样本数，计算平均差

计算正确率并返回

While(判断正确率是否符合想要的结果)

不符合:调用BPNetwork.train(Dataset)继续训练数据

拿到每一行数据

调用BPNetwork.train(Record)

设置输入层的输出值为当前记录值

BPNetwork.getLayer(0).setOutputs(record);

更新后面各层的输出值**tmp += weights[i][j] \* lastOutputs[i];**

**outputs[j] = 1 / (1 + Math.*pow*(Math.*E*, -(tmp + theta[j])));**

BPNetwork.updateOutput();

更新输出层的错误率**errors[j] = outputs[j] \* (1 - outputs[j]) \* (target[j] - outputs[j]);**

BPNetwork.updateOutputLayerErrors(record);

更新每一层的错误率**tmp += nextErrors[k] \* weights[j][k];**

**errors[j] = outputs[j] \* (1 - outputs[j]) \* tmp**

BPNetwork.updateErrors();

更新权重：**weights[i][j] += (*LEARN\_RATE* \* errors[j] \* lastOutputs[i]);**

BPNetwork.updateWeights();

更新偏置**biases[j] += *LEARN\_RATE* \* errors[j];**

BPNetwork.updateBiases();

符合:返回权重以及偏置

}