**说明文档**

这是对于此项目的说明，主要分为两部分：一，完成功能及位置介绍，二，使用说明。

**一， 完成功能及位置介绍：**

1. 使用MongoDB做数据库管理

（实现位置：主要controllers\default.go 文件中GetRandom,FillMatrix函数中）

1. 集成Redis缓存机制

（实现位置：主要controllers\default.go 文件中FillMatrix函数中）

1. 对外暴露RESTful API，但客户端不能选择web portal/client/command line

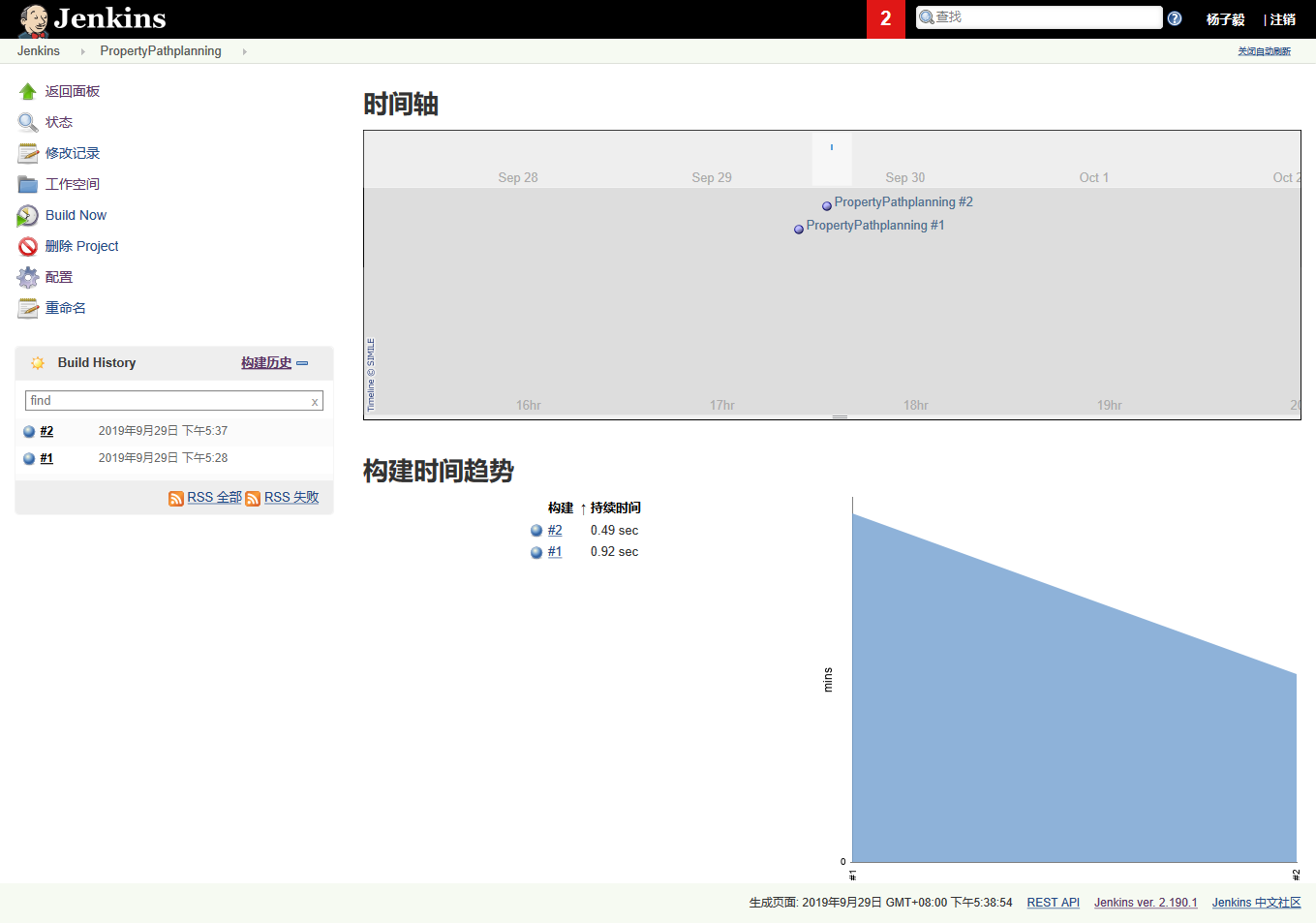
（实现位置：整个项目都在beego的框架下搭建）

1. 抽样百分比可输入，根据百分比随机抽样座位号

（实现位置：主要controllers\default.go 文件中GetRandom函数中以及views/index.html文件）

1. 最短路径规划算法

（实现位置：主要controllers\default.go 文件中TSP，ShortestPathDisplay , FillMatrix函数中）

1. 使用Jenkins持续集成CI/CD （不知道这样算不算完成） 

**二，使用说明**

此项目采用的是beego的框架搭建，而beego又是一个经典的MVC框架，项目文件夹下main.go是入口文件，controllers是逻辑控制模块，routers是路由模块，views是视图模块。

因为定义了结构体如下

type Seat struct {  
 No int *//座位号* Name string *//财产的名称* Amount int32 *//财产的数额*}*//座位及相关信息*type Range struct {  
 FirstNo int*//第一个座位编号* SecondNo int*//第二个座位编号* Distance int*//两点距离*}*//两个座位间距离*

，所以程序的预置使用条件是在monogoDB中有”Property”库中有”Seats”,”Distance”集合将座位信息如上储存，在Distance集合中要求要有两个座位的距离都存储才算将两地距离存储完成，限定任意两地可达并已知距离并储存才可运行，并限定编号0为起始出发点，如下例

Range{  
 FirstNo: 1,  
 SecondNo: 0,  
 Distance: 12,  
}

Range{  
 FirstNo: 0,  
 SecondNo: 1,  
 Distance: 12,  
}

出于测试考虑，controllers\default.go 文件中有函数SaveMongoDB()用于写入10个座位数据及其随机距离来供测试使用，该函数部分如下

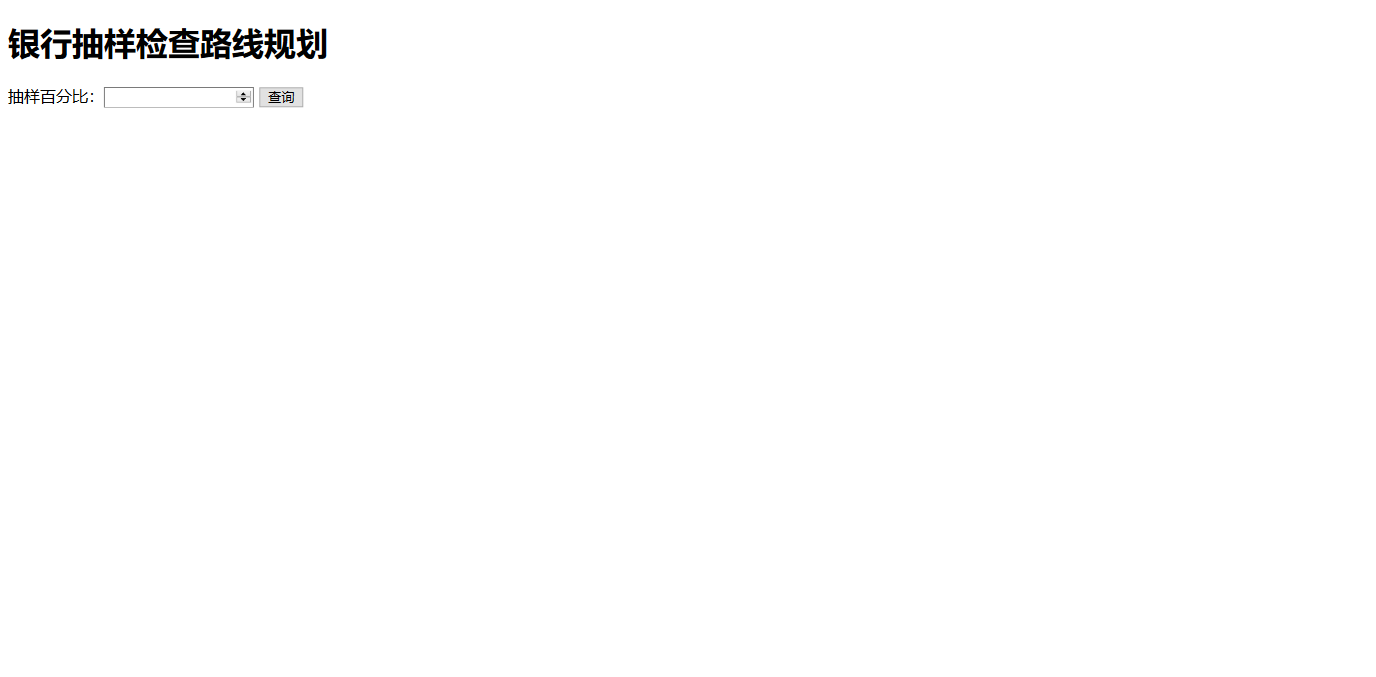
session, err := mgo.Dial("mongodb://localhost")  
if err != nil { panic(err)}  
defer session.Close()  
session.SetMode(mgo.Monotonic, true)  
c := session.DB("Property").C("Seats")  
 err = c.Insert(&Seat{  
 No: 0,  
 Name: "Origin",  
 Amount: 0,  
 }

如有测试数据，将代码中的这一函数调用注释掉即可

c=session.DB("Property").C("Distance")  
rand.New(rand.NewSource(time.Now().UnixNano()))  
for i:=0;i<10 ;i++ {  
 for j := 0; j < 10; j++ {  
 if i!=j{  
 err=c.Insert(&Range{  
 FirstNo: i,  
 SecondNo: j,  
 Distance:rand.Intn(99)+1 ,  
 })  
 }  
 }  
}

err = c.Insert(&Seat{  
 No: 0,  
 Name: "Origin",  
 Amount: 0,  
 }

下面是运行，在gobuild之后点开浏览器localhost网址会有以下界面



输入1-100数值，再点查询

会得到以下界面，结果以途经座位编号来显示，路径包含出发与回到出发点

