MongoDB与SpringMVC学习笔记

## 1.MongoDB

MongoDB[1] 是目前在IT行业非常流行的一种非关系型数据库(NoSql),其灵活的数据存储方式备受当前IT从业人员的青睐。MongoDB很好的实现了面向对象的思想(OO思想),在MongoDB中 每一条记录都是一个Document对象。MongoDB最大的优势在于所有的数据持久操作都无需开发人员手动编写SQL语句,直接调用方法就可以轻松的实现CRUD操作。

MongoDB是一种NoSql数据库。NoSql[2]，全称是 Not Only Sql,指的是非关系型的数据库。下一代数据库主要解决几个要点：非关系型的、分布式的、开源的、水平可扩展的。原始的目的是为了大规模web应用，这场运动开始于2009年初，通常特性应用如：模式自由、支持简易复制、简单的API、最终的一致性（非ACID）、大容量数据等。NoSQL被我们用得最多的当数key-value存储，当然还有其他的文档型的、列存储、图型数据库、xml数据库等

### MongoDB的特点

高性能、易部署、易使用，存储数据非常方便。

主要功能特性有：

面向集合存储，易存储对象类型的数据。

模式自由。

支持动态查询。

支持完全索引，包含内部对象。

支持查询。

支持复制和故障恢复。

使用高效的二进制数据存储，包括大型对象（如视频等）。

自动处理碎片，以支持云计算层次的扩展性

支持Python，PHP，Ruby，Java，C，C#，Javascript，Perl及C++语言的驱动程序，社区中也提供了对Erlang及.NET等平台的驱动程序。

文件存储格式为BSON（一种JSON的扩展）。

可通过网络访问

### MongoDB功能

面向集合的存储：适合存储对象及JSON形式的数据。

动态查询：Mongo支持丰富的查询表达式。查询指令使用JSON形式的标记，可轻易查询文档中内嵌的对象及数组。

完整的索引支持：包括文档内嵌对象及数组。Mongo的查询优化器会分析查询表达式，并生成一个高效的查询计划。

查询监视：Mongo包含一个监视工具用于分析数据库操作的性能。

复制及自动故障转移：Mongo数据库支持服务器之间的数据复制，支持主-从模式及服务器之间的相互复制。复制的主要目标是提供冗余及自动故障转移。

高效的传统存储方式：支持二进制数据及大型对象（如照片或图片）

自动分片以支持云级别的伸缩性：自动分片功能支持水平的数据库集群，可动态添加额外的机器。

### 适用场合

网站数据：Mongo非常适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。

缓存：由于性能很高，Mongo也适合作为信息基础设施的缓存层。在系统重启之后，由Mongo搭建的持久化缓存层可以避免下层的数据源 过载。

大尺寸，低价值的数据：使用传统的关系型数据库存储一些数据时可能会比较昂贵，在此之前，很多时候程序员往往会选择传统的文件进行存储。

高伸缩性的场景：Mongo非常适合由数十或数百台服务器组成的数据库。Mongo的路线图中已经包含对MapReduce引擎的内置支持。

用于对象及JSON数据的存储：Mongo的BSON数据格式非常适合文档化格式的存储及查询。

### MongoDB与关系型数据库比较

|  |  |  |
| --- | --- | --- |
| 对比项 | MongoDB | mysql,oracle |
| 表 | 集合collection | 二维表table |
| 表中行 | 文档document | 一条记录record |
| 表字段 | 键key | 字段field |
| 字段值 | 值value | 值value |
| 主外键 | 无 | PK,FK |
| 灵活度、扩展性 | 极高 | 差 |

1.关系型数据库每一行的字段是固定的

2. MongoDB的每一个document的key可以不一样

3.关系型数据库查询使用SQL

4. MongoDB查询使用内置find函数（基于Bson的特殊查询工具）

### MongoDB之bson的介绍

MongoDB[3]使用了BSON这种结构来存储数据和网络数据交换。把这种格式转化成一文档这个概念(Document)，因为BSON是schema-free的，所以在MongoDB中所对应的文档也有这个特征，这里的一个Document也可以理解成关系数据库中的一条记录(Record)，只是这里的Document的变化更丰富一些，如Document可以嵌套。

MongoDB以BSON做为其存储结构的一种重要原因是其可遍历性。

BSON是一种类json的一种二进制形式的存储格式，简称Binary JSON，它和JSON一样，支持内嵌的文档对象和数组对象，但是BSON有JSON没有的一些数据类型，如Date和BinData类型。

BSON可以做为网络数据交换的一种存储形式，这个有点类似于Google的Protocol Buffer，但是BSON是一种schema-less的存储形式，它的优点是灵活性高，但它的缺点是空间利用率不是很理想，

BSON有三个特点：轻量性、可遍历性、高效性。

BSON数据类型如下表：

|  |  |
| --- | --- |
| null | 用来表示空或不存在的字段 |
| 布尔类型 | 两个数值true和false |
| 32位或64位整数 | Shell中不支持，需要用到其他高级语言的驱动来完成，JS不可使用 |
| 64为浮点数 | Shell中使用的数字全是这种类型{x:3.412} |
| UTF-8 | 字符串的类型 |
| ObjectID（对象ID） | 内置默认的ID对象{\_id:ObjectId()} |
| 日期类型 | {x:new Date()} |
| 正则表达式 | {x:/uspcat/i} |
| Javascript代码块 | {x:function(){...}} |
| undefined | 未定义类型（和null不是一个类型） |
| 数组类型 | {array:[20,56]} |
| 内嵌文档 | {docu:{name:"uspcat",age:20}} |
| 二进制 | 任意字节的字符串（Shell中不支持） |

### MongDB在shell客户端增删改查的实现[4]

1.创建数据库： use [database] 如 use foobar

2. 查询所有数据库 show dbs

3. 给数据库添加集合并添加记录 db.[collection].insert({...})

例子：创建集合persons，插入document {name:"lili",age:20}

db.persons.insert({name:"lili",age:20})

4. 查询数据库中所有集合 show collections

5. 查询指定集合中的文档

查询所有：db.[collection].find()

查询第一条数据：db.[collection].findOne()

6. 更新文档数据 db.[collection].update({查询条件},{更新内容})

例子：var p=db.persons.findOne()

db.persons.update(p,{name:"uspcat2"})//这种修改同时会删除其他属性

使用set修改器

db.persons.update({name:"uspcat2"},{$set:{age:1,name:"uspcat"}})

7. 删除集合中相应的文档

db.[collection].remove({条件})

例子：db.persons.remove({name:"uspcat"})

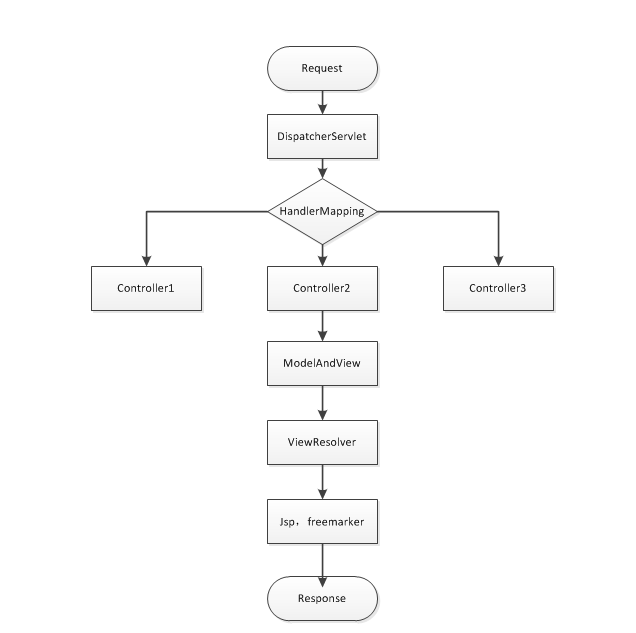
8. 删除数据库中集合 db.[collection].drop()

9. 删除数据库 db.dropDatabase()

## 2.Spring MVC

Spring MVC[5]是一种基于Java的实现了Web MVC设计模式的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行职责解耦，基于请求驱动指的就是使用请求-响应模型 Spring Web MVC可简化我们日常Web开发的。

Spring MVC的架构如下图：



核心架构的具体流程步骤如下：

1、  首先用户发送请求——>DispatcherServlet（中央控制器），DispatcherServlet在收到请求后自己不进行处理，根据HandlerMapping的映射策略将请求转发给对应的控制器Controller；

2、  DispatcherServlet——>HandlerMapping（映射处理器）， HandlerMapping将会把请求映射为HandlerExecutionChain对象（包含一个Handler处理器（页面控制器）对象、多个HandlerInterceptor拦截器）对象，通过这种策略模式，很容易添加新的映射策略；

3、 HandlerMapping——>Controller（处理器），通过HandlerMapping的映射转发策略，DispatcherServlet将请求转发给对应的控制器Controller。

4、 Controller——>ModelAndView，ModelAndView是服务层返回数据和试图的封装类。控制器调用业务逻辑层，进行相应的业务逻辑处理，处理后的结果通过ModelAndView返回。

5、  ModelAndView——> ViewResolver（视图解析器）， ViewResolver将把逻辑视图名解析为具体的View，通过这种策略模式，很容易更换其他视图技术；

6、  View——>渲染，View会根据传进来的Model模型数据进行渲染，此处的Model实际是一个Map数据结构，因此很容易支持其他视图技术；

7、返回控制权给DispatcherServlet，由DispatcherServlet返回响应给用户，到此一个流程结束。

## 3.Spring MVC+MongoDB实现的小示例

设计初衷：研究生在校期间，有非常多的课程大作业需要组队完成，遇到问题或检查产出时，通常要确定一个开会时间。由于大多数人都很忙，各自的空余时间难以手动动统计。设计这个小程序旨在统计小组成员的空余时间，根据统计结果，给出建议的开会时间。

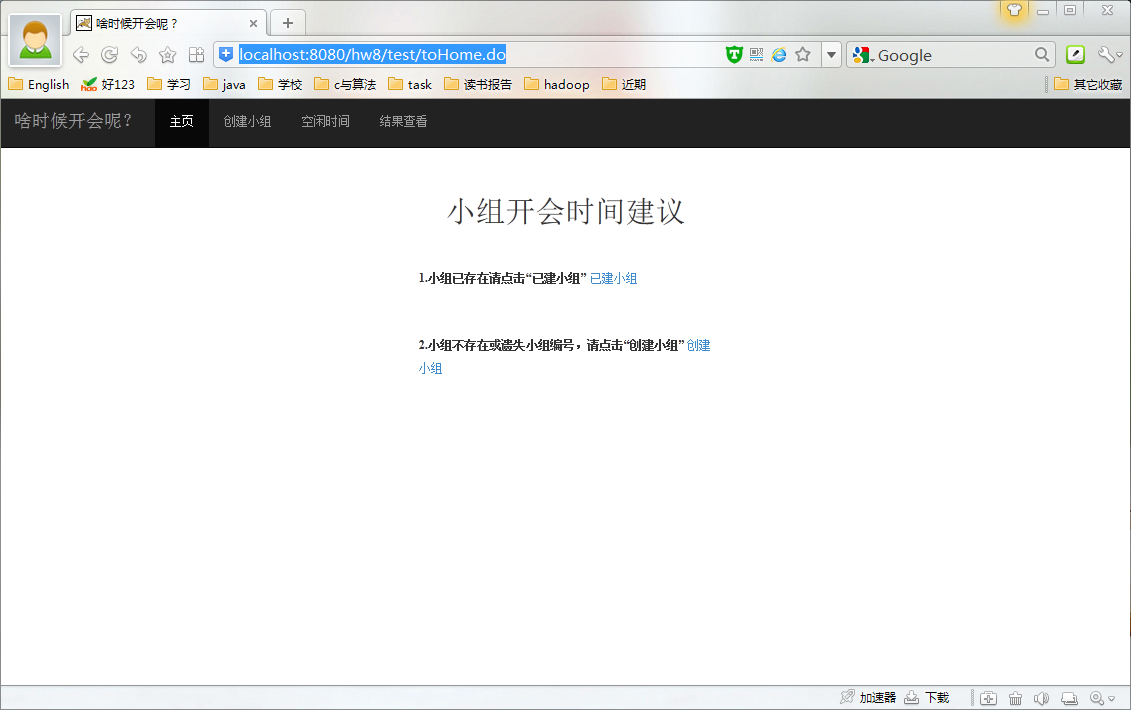
使用技术：前端使用BootStrap[6]完成，框架使用的是Spring MVC，后台数据库采用的是NoSql数据库MongoDB。

功能描述：

小示例总共包括四个页面：主页，创建小组页面，空余时间统计页面，结果展示页面。

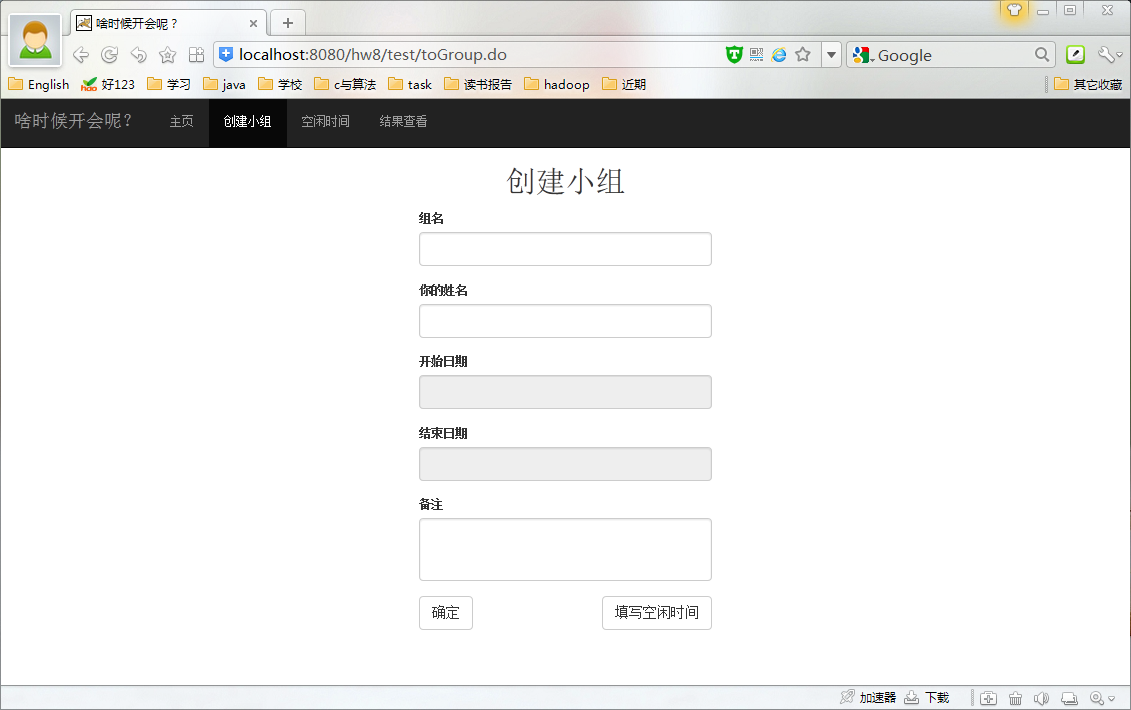
使用步骤：

1.通过链接<http://localhost:8080/hw8/test/toHome.do>进入主页；

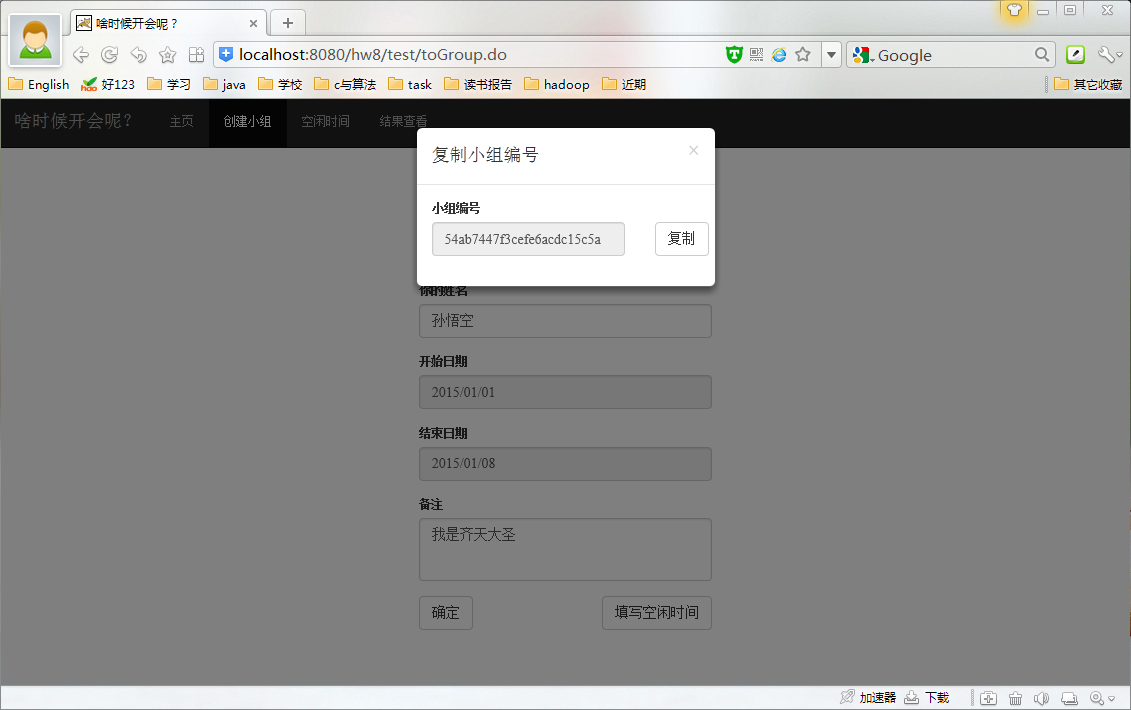


若原先为创建好小组：

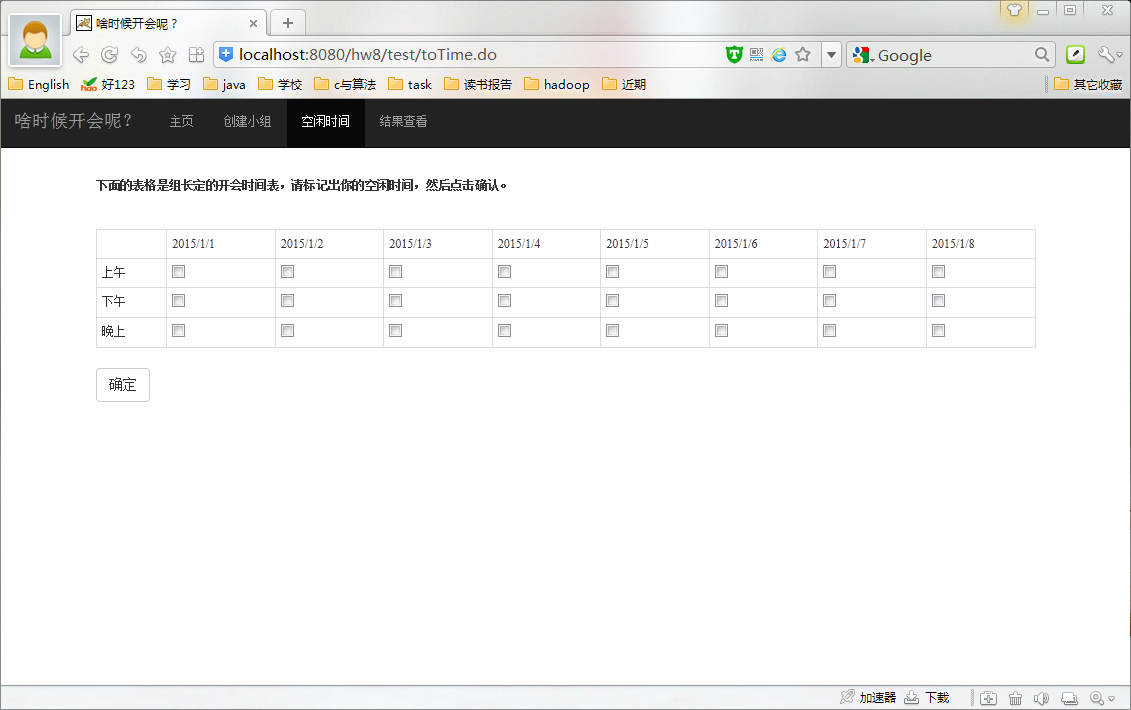
2. 由于小组还未创建，点击"创建小组"进入创建小组页面；



填写好小组名称，你的姓名、开会的大概时间段以及备注信息，点击确认按钮，小组信息写入数据库，生成一个小组编号，复制保存这个小组编号，其他小组成员即可通过这个小组编号进行空闲时间统计。

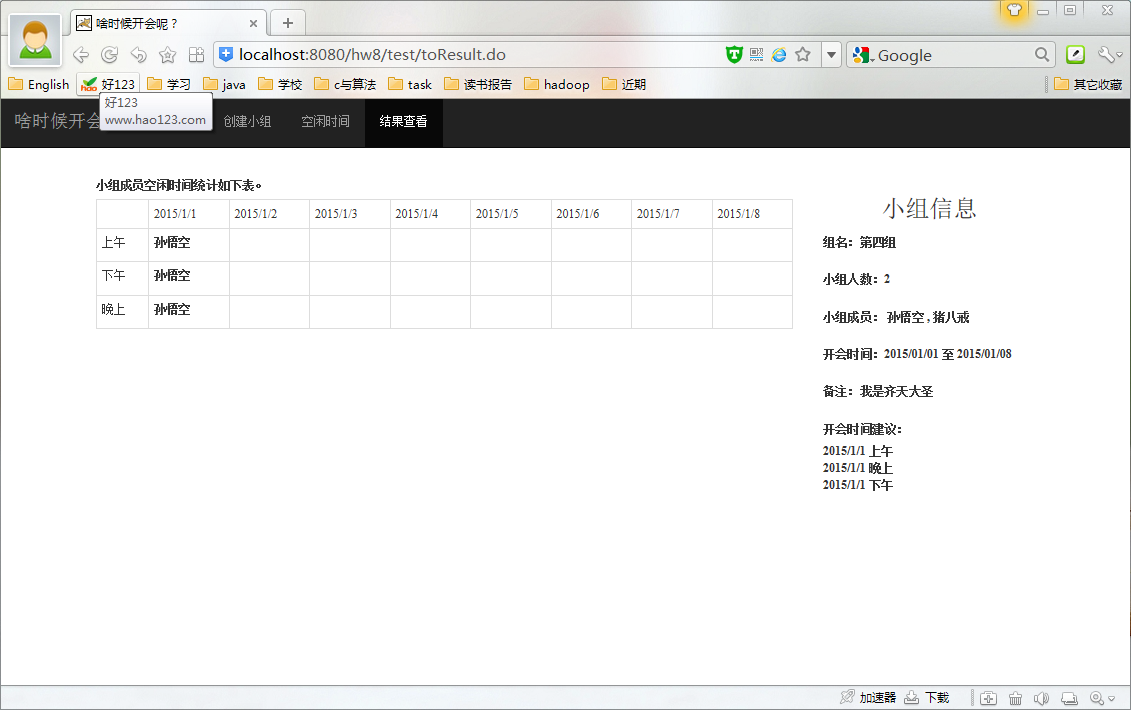


3.保存好小组编号"54ab7447f3cefe6acdc15c5a"后,页面会跳转到空闲时间；



空闲时间统计页面是根据你刚才创建小组是填写的信息，自动在页面上生成一个表格，选择你空闲的时间段，然后点击确认，提交你的空余时间。

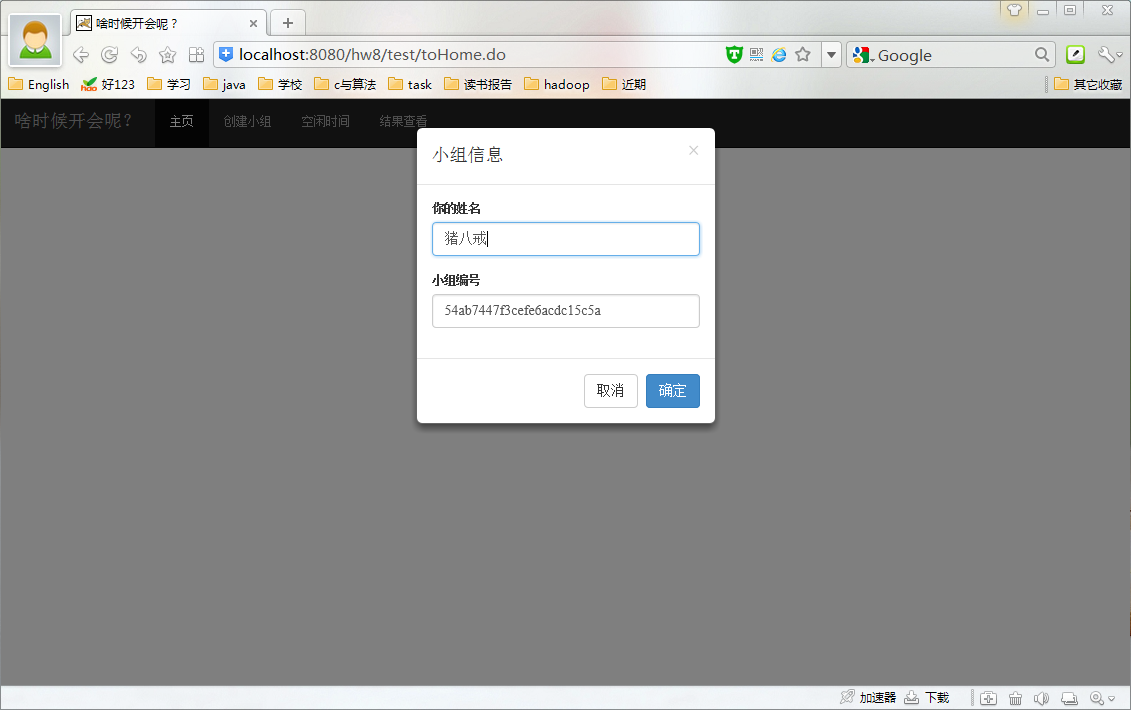
4.空余时间提交后，会调到结果展示页面；



结果展示页面分两栏，左边一栏是一个表格，各个td中的内容为空闲小组成员的名字；右边一栏是小组信息的简单展示，包括：小组名称、本参与空余时间统计的人数、小组成员的姓名、开会时间信息、备注信息以及根据空余时间统计结果给出的开会时间的建议。

若原先以创建好小组：

2.由于已经存在小组，点击"已建小组"，会跳出一个对话框，填写你的姓名以及小组编号。



点击确认后会跳转到空余时间统计页面。之后的操作同"若原先为创建好小组"情况下的3、4步。

## 参考资料

[1]MongoDB的维基百科，http://zh.wikipedia.org/wiki/MongoDB

[2]NoSQL介绍，http://database.51cto.com/art/201103/247882.htm

[2]BSON介绍，http://blog.csdn.net/amuseme\_lu/article/details/6573912

[4]MongoDB API，http://api.mongodb.org/

[5]Spring MVC介绍，http://jinnianshilongnian.iteye.com/blog/1594806

[6] BootStrap3学习，http://v3.bootcss.com/getting-started