共享单车租赁系统（Share Bike Lease System，简称SBLS）是一个校内共享单车的管理系统。在基于MVC的三层开发架构体系上，采用Java作为主要开发语言，结合SpringMVC+Spring+Mybatis三大开发框架进行该B/S系统的开发。一般来说，B/S系统的开发都基于MVC（Moel-View-Control）三层架构体系，使得前台和后台解耦，从而使得各自的业务逻辑互不影响。前台页面交给View展示，后台业务逻辑交给Model处理，Control负责控制两者的同步。这种开发架构体系能够很好的分离逻辑、数据与页面，有助于管理复杂的应用程序。本系统基于MVC体系开发，增加了系统的稳定性，同时也为管理校内共享单车提供了有效助力。

在页面显示上，本系统采用了“厂”字型的页面布局。页面整体被划分为三部分，分别是：位于上方的用户信息栏、位于左边的操作导航栏，以及位于中间的页面显示部分。其中用户信息栏用于显示用户的登录登出操作和系统logo等相关信息；操作导航栏则是给用户提示相对应的操作；中间的页面显示部分则是用于显示用户与系统交互后返回的数据。

登录操作

用户在登录SBLS系统时，首先前台的JS会对用户的输入信息做一个格式的验证。若验证不通过，则会在具体的输入栏后面提示具体的错误原因。在通过验证后，用户填写的信息，经过SpringMVC转发到LoginController里面，LoginControler封装好用户输入的信息后，通过LoginDao与数据库交互验证。若验证失败，则LoginController会封装好相对应的错误提示，然后返回到页面；若验证通过，则会将用户的信息放到session中，使用户在下一次有效期内登录可以不用重新输入用户名与密码，然后在封装好相对应的信息并跳转到用户首页。

定位

用户在定位页面上，选择SBLS系统给出的位置，此时系统会将用户所选的定位信息发送到MapController，MapController将用户所选择的信息封装并发送到MapDao中与数据库交互，从而取得该定位500米半径范围内的单车站点以及该站点内的单车信息情况。

租车

用户在单车信息显示页面上，选中某辆单车并进行租赁操作后，系统首先会通过JS对用户进行验证，具体包括：用户是否有上一次租赁操作且未归还、用户是否缴纳了押金。通过验证后系统将用户所选择的单车编号等相关信息传送到LeaseController，LeaseController在取得单车编号，并封装用户ID、租赁时间等相应信息，然后发送给LeaseDao从而与数据库交互从而更改单车状态、站点状态、以及租赁记录。然后在将返回信息用JSONObject封装后返回给前台页面。

还车

用户在订单列表中，选择还车操作后，系统首先将通过JS检测用户所选单车归还站点是否还有空余位置可供单车停靠。如果没有，则会提示用户选择其他站点；如果通过验证，系统会将单车ID、用户ID、站点ID等相关信息发送到LeaseController，LeaseController首先会计算出这次租赁操作所产生的费用。然后通过LeaseDao与数据库交互，判断用户当前的余额是否足够支付该次租赁操作所产生的费用。如果不够，LeaseController会封装相对应的错误提示信息并返回到前台页面，提示用户前往充值。若如果余额足够，则会修改数据库中关于单车、站点、租赁记录的相关信息。

金额充值

用户在操作栏上选择金额充值后，输入所需充值的金额数目。系统首先通过JS对金额数目作验证。若验证不通过（如金额数目为0或者负数，金额数目大于1000），则系统会给出相对应的错误提示；若验证通过，系统将金额数目发送到UserController，UserController会通过UserDao与数据库交互，从而修改用户的金额数目。然后再返回相对应信息到前台页面。

重置密码

用户在登录页面上选择重置密码操作后，系统跳转到验证页面。用户输入身份证和邮箱后，系统首先通过JS对用户所输入的身份证号进行验证。若验证不通过，则说明操作者不是本人，无法进行重置密码操作；若验证通过，系统将用户所输入的邮箱发送到UserController，UserController会调用AuditResultEmailThread线程来发送邮件到用户输入的邮箱上，用户在接收到邮箱后点击邮箱中的地址从而跳转到重置密码的页面。此时用户在页面上输入新的密码。系统在经过验证后将密码发送到UserController，UserController将密码封装后通过UserDao与数据库交互，从而去修改用户存储在数据库中的密码。

单车数据统计

管理员在后台管理页面中，选择单车数据管理操作后，此时便相当于向BikeController发送了一个请求，BikeController在接收该次请求后，根据相对应的URL路径转发，找到对应的getAllBike()方法，此时该方法会通过BikeDao去数据库查找所有的单车数据。与此同时，会将取得数据通过JfreeChat绘制相对应的饼状图来显示单车的数据情况。

单车管理

管理员在后台管理页面中，选择单车管理操作后，此项系统接收请求，根据URL路径在BikeController中找到了bikeManage()方法，然后通过BikeDao与数据库交互后返回了单车数据，再通过BikeController转发到前台显示页面。此时管理员可以在页面中对单车进行删除、维护、投放等操作。而这一系列操作都是通过BikeController来根据URL路径来找到相对应的处理方法，进而完成操作。

删除单车：

管理员在单车列表中，选择某辆单车后并点击修改按钮后，此事系统将该单车ID等相关信息发送到BikeController的deleteBike()方法，此时该方法会先判断该辆单车是否处于使用状态。若是，则会封装相对应的错误提示并返回到前台；若不是，则通过BikeDao与数据库交互从而完成该辆单车的删除。

修理单车：

管理员在单车列表中，选择某辆损坏的单车后并点击修理按钮，此时系统将该单车的ID等相关信息发送到BikeController的fixBike()方法，该方法封装了单车信息后，通过BikeDao与数据库交互后，从而去修改单车状态。与此同时，系统中的定时任务，会每隔10分钟去数据库查询状态为“待修理”的单车，从而去更改单车状态来达到修理单车的目的。

投放单车：

管理员在单车列表中，选择状态为“闲置”的单车后并点击投放按钮，此时系统会检测投放单车的站点是否有空余位置可供单车停靠。若没有，系统会给出相对应的错误提示；反之系统则将该单车的ID等相关信息发送到BikeController的launch()方法，该方法封装了单车信息后，通过BikeDao与数据库交互，从而修改单车状态为空闲，达到投放单车的目的。

站点管理

管理员在后台管理页面中，选择站点管理操作后，此项系统接收请求，根据URL路径在SiteController中找到了getAllSite ()方法，然后通过SiteDao与数据库交互后返回了站点数据，再通过SiteController转发到前台显示页面。此时管理员可以在页面中对站点进行修改、删除等操作。而这一系列操作都是通过SiteController来根据URL路径来找到相对应的处理方法，进而完成操作。

修改站点信息：

管理员在站点列表中，选择某个站点后并点击修改按钮，此时系统会先检测该站点是否处于使用状态（即：是否有单车停靠）。若是，则系统会给出相对应提示；反之系统则将该站点的ID以及管理员输入的数据等相关信息发送到SiteController的updateSite()方法，该方法封装了站点信息后，通过SiteDao与数据库交互后，从而修改该站点信息。

删除站点：

管理员在站点列表中，选择某个站点后并点击删除按钮，此时系统会先检测该站点是否处于使用状态（即：是否有单车停靠）。若是，则系统会给出相对应提示；反之系统则将该站点的ID等相关信息发送到SiteController的deleteSite()方法，该方法封装了站点信息后，通过SiteDao与数据库交互后，从而删除该站点。