

分页

分页功能包括两个方面，一个是页面内容，一个是分页导航栏中的内容。

1.1 利用limit获取指定页面的页面内容数据

mysql的limit后面两个数字：

第一个数字：startIndex（用于指定要从结果集中的哪一行开始拿数据。结果集的第一行记录的下标为0。）

第二个数字：pageSize（每页显示的记录条数）

假设已知页码pageNum，还有每页显示的记录条数pageSize，第一个数字可以动态的获取吗？

$startIndex = (pageNum - 1) * pageSize$

获取第PageNum页的页面内容数据：

```
select * from tableName ..... limit (pageNum - 1) * pageSize, pageSize
```

示例：

t_car表

对象

t_car @powernode (localhost) - 表

* 无标题 - 查询

开始事务

文本

筛选

排序

导入

导出

数据生成

创建图表

id	car_num	brand	guide_price	produce_time	car_type
165	1201	大众迈腾	21.00	2020-11-11	燃油车
166	1202	大众速腾	30.00	2020-11-11	燃油车
167	1203	奔驰GLC	5.00	2010-12-03	燃油车
168	1204	奥迪Q7	3.00	2009-10-11	燃油车
169	1205	朗逸	4.00	2001-10-11	新能源
170	1206	奔驰E300L	50.00	2003-02-03	新能源
171	1207	奥迪A6	30.00	2000-01-02	燃油车

```
select * from t_car limit 0, 3;
```

id	car_num	brand	guide_price	produce_time	car_type
165	1201	大众迈腾	21.00	2020-11-11	燃油车
166	1202	大众速腾	30.00	2020-11-11	燃油车
167	1203	奔驰GLC	5.00	2010-12-03	燃油车

limit是用于获取页面内容数据，并不能获取到用于分页导航栏中的数据。

获取页面内容数据不难，难的是获取分页相关的数据（也就是要用于分页导航栏的数据）比较难。可以借助mybatis的PageHelper插件。 分页相关的数据比如：总记录条数，是否有下一页，是否有上一页，分页卡片要显示的个数，等等。

1.2 PageHelper插件

示例：

CarMapper.xml：

只要用了PageHelper，就不用在DQL语句中加limit了，底层会自己加上limit。

```
<select id="selectAll" resultType="Car">
select * from t_car
</select>
```

CarMapper接口：

```
List<Car> selectAll();
```

测试：

```
@Test
public void testPageHelper() throws Exception{
    SqlSessionFactory sqlSessionFactory = new
        SqlSessionFactoryBuilder().build(Resources.getResourceAsStream("mybatis-
config.xml"));
    SqlSession sqlSession = sqlSessionFactory.openSession();
    CarMapper mapper = sqlSession.getMapper(CarMapper.class);

    // 开启分页功能。startPage是一个类方法，第一个参数是PageNum，第二个参数是PageSize。
    PageHelper.startPage(2, 2);

    // 执行查询语句。只要用了PageHelper，就不用在sql语句中加limit了。
    List<Car> cars = mapper.selectAll();

    // 获取分页信息对象。PageInfo类是PageHelper中提供的，用于封装分页所要用的所有数据。
    // PageInfo构造方法的第一个参数是查询结果集合，第二个参数是想在分页导航栏中设置的卡片数。
    // 所以获取分页信息对象，一定要在执行DQL语句之后。
    PageInfo<Car> pageInfo = new PageInfo<>(cars, 5);

    System.out.println(pageInfo);
}
```

PageInfo对象打印出来如下所示：

```
PageInfo{
pageNum=2, pageSize=2, size=2, startRow=3, endRow=4, total=6, pages=3,
list=Page{count=true, pageNum=2, pageSize=2, startRow=2, endRow=4, total=6,
pages=3, reasonable=false, pageSizeZero=false}
[Car{id=86, carNum='1234', brand='丰田霸道', guidePrice=50.5, produceTime='2020-
10-11', carType='燃油车'},
Car{id=87, carNum='1234', brand='丰田霸道', guidePrice=50.5, produceTime='2020-10-
11', carType='燃油车'}],
prePage=1, nextPage=3, isFirstPage=false, isLastPage=false, hasPreviousPage=true,
hasNextPage=true,
navigatePages=5, navigateFirstPage=1, navigateLastPage=3, navigatepageNums=[1, 2,
3]
}
```

其中，**pageNum**：是指当前页码

pagesize：是指每页的页面内容显示的是几条记录

startRow：是指页面内容数据是从结果集的第几行开始拿的，是一个下标

endRow：是指页面数据内容是拿到了结果集的第几行

total：是指结果集中一共有几条记录

pages：是指结果集中的记录按**pagesize**的分法能被分成多少页

isFirstPage：是指当前页码是不是首页

isLastPage：是指当前页码是不是尾页

prePage：是指上一页的页码

nextPage：是指下一页的页码

navigatePages：分页导航栏中的卡片数

关键点：

一定一定要注意：要在执行查询语句之前开启分页功能。

在查询语句之后封装PageInfo对象。意思就是：获取分页信息对象，一定要在执行DQL语句之后。

(PageInfo对象将来会存储到request域当中。在页面上展示。如：request.setAttribute("pageInfo", pageInfo);)