分页组件

概述

为了契合项目，分页组件采用了公司自身封装的组件，其他分页组件并不能更好的融合到项目里面，此外此组件比较轻量，并且是物理分页组件，并不会给内存带来大的消耗。

依赖

```
<dependency>
  <groupId>com.gsww.cd.commons</groupId>
  <artifactId>commons-dal-starter-jdbc</artifactId>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>

<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
</dependency>
```

因为此组件是物理分页组件所以需要依赖对应的数据库和数据库连接池，示例选用的数据库是 `MySql` 以及 `HikariCP` 连接池。依赖也在上面一共给出。 注：因为是封装的项目，具体还有些依赖可能没有写明，使用方法会在下面给出。

API

分页组件提供了五种查询分页的方式，其中四种是方法重载，具体如下：

返回值	方法	解释
`<E> Page<E>`	`queryForPage(String var1, Object var2, Class<E> var3, PageParam var4);`	查询分页，返回传入实体类的分页对象，sql语句参数是Object类型
`<E, R> Page<R>`	`queryForPage(String var1, Object var2, Class<E> var3, PageParam var4, Function<E, R> var5);`	查询分页，返回传入实体类的分页经由函数式接口转换后的分页对象，sql语句参数是Object类型
`<E> Page<E>`	`queryForPage(String var1, InputParameters var2, Class<E> var3, PageParam var4);`	查询分页，返回传入实体类的分页对象，sql语句参数是封装的key value形式的对象
`<E, R> Page<R>`	`queryForPage(String var1, InputParameters var2, Class<E> var3, PageParam var4, Function<E, R> var5);`	查询分页，返回传入实体类的分页经由函数式接口转换后的分页对象，sql语句参数是key value形式的对象
`Page`	`queryForMapPage(String var1, Object var2, PageParam var3);`	查询分页，返回分页对象，其中列表数据封装在map里面

示例

示例采用实体类是公共组件项目里提供的示例实体类 `Department`，对应的代码和 `DTO` 类如下：

```
/**
 * Created by SuperS on 2019/9/5.
 *
 * @author SuperS
 */
@Data
@Entity
@Table(name = "eco_department")
@EqualsAndHashCode(callSuper = true)
public class Department extends AbstractDomainValue implements CreateAndUpdateTimeable,
UnDeletable {

    @Column(name = "department_name")
    private String name;

    /**
     * 创建时间
     */
    @Column(name = "create_time")
    private Date createTime;
    /**
     * 更新时间
     */
    @Column(name = "update_time")
    private Date updateTime;
    /**
     * 删除标识
     */
    @Column(name = "deleted_flag")
    private Boolean deletedFlag;

    @Override
    public boolean isDeleted() {
        return deletedFlag;
    }

    @Override
    public void markedAsDeleted() {
        this.deletedFlag = Boolean.TRUE;
    }

    @Override
    public void markedAsAvailable() {
        this.deletedFlag = Boolean.FALSE;
    }

    public DepartmentDTO convert() {
       return DepartmentDTO.builder()
               .id(this.id)
               .name(this.name)
               .build();
    }
}
```

```
/**
 * Created by SuperS on 2019/9/5.
```

```
 *
 * @author SuperS
 */
@Data
@ApiModel("部门实体")
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class DepartmentDTO {
    @ApiModelProperty("主键Id")
    private Long id;
    @ApiModelProperty("部门名称")
    private String name;
}
```

API接口提供了两种分页查询示例，对应着 `GET` 和 `POST` 两种请求，查询接口使用 `POST` 考虑到前端查询可能有不止一种参数，可以把同一模块多个参数封装成一个dto对象用于传输。而 `GET` 请求用于单个或者几个不同模块参数的方式，因为前端封装的 `axios` 的 `get` 方法不支持这种对象参数传递的方式，需要将分页信息也一并封装到dto里面，需要看具体情况使用。

```
/**
 * 分页示例
 * @author litian
 */
@Api(value = "Pagination Api", tags = {"分页示例"})
@FeignClient("${eco.sample.biz1.cloud-app-name}")
@RequestMapping(path = "/core-api/eco/sample/v1/pagination")
public interface PaginationApi {

    /**
     * 分页Post请求示例
     * @param departmentDTO
     * @param currentPage
     * @param pageSize
     * @return 分页<数据实体>
     */
    @ApiOperation(value = "分页Post请求示例", notes = "传递整个条件Dto以及分页信息，Post请求查询分页列表")
    @ApiImplicitParams({
            @ApiImplicitParam( name = "currentPage", value = "查询的页数", required = true, dataType = "Integer"),
            @ApiImplicitParam( name = "pageSize", value = "每页查询的数量", required = true, dataType = "Integer"),
            @ApiImplicitParam( name = "departmentDTO", value = "查询的条件dto", required = false, dataType = "DepartmentDTO")
    })
    @PostMapping("/findPaginationPage/{currentPage}/{pageSize}")
    JsonResponse<Page<DepartmentDTO>> findPaginationPage(@RequestBody DepartmentDTO departmentDTO,
                                                         @PathVariable Integer currentPage,
                                                         @PathVariable Integer pageSize);

    /**
     * 分页get请求示例
     * @param currentPage
     * @param name 传递参数
     * @param pageSize
     * @return 分页<数据实体>
     */
    @ApiOperation(value = "分页get请求示例", notes = "分页Post请求示例，Get请求查询分页列表")
    @ApiImplicitParams({
```

```
            @ApiImplicitParam( name = "currentPage", value = "查询的页数", required = true,
dataType = "Integer"),
            @ApiImplicitParam( name = "pageSize", value = "每页查询的数量", required = true,
dataType = "Integer"),
            @ApiImplicitParam( name = "name", value = "查询参数", required = false, dataType =
"String")
    })
    @GetMapping("/findPaginationPage")
    JsonResponse<Page<DepartmentDTO>> findPaginationPage(@RequestParam(value = "name")String
name,
                                                         @RequestParam(value =
"currentPage") Integer currentPage,
                                                         @RequestParam(value = "pageSize")
Integer pageSize);

}
```

controller层：

```
/**
 * 分页示例
 * @author litian
 */
@RestController
public class PaginationController implements PaginationApi {

    @Resource
    private PaginationService paginationService;

    @Override
    public JsonResponse<Page<DepartmentDTO>> findPaginationPage(DepartmentDTO departmentDTO,
Integer currentPage, Integer pageSize) {
        return JsonResponse.ok(paginationService.findPaginationPage(departmentDTO,
currentPage, pageSize));
    }

    @Override
    public JsonResponse<Page<DepartmentDTO>> findPaginationPage(String name, Integer
currentPage, Integer pageSize) {
        return JsonResponse.ok(paginationService.findPaginationPage(name, currentPage,
pageSize));
    }
}
```

服务接口层：

```
/**
 * 分页示例
 * @author litian
 */
public interface PaginationService {

    /**
     * 分页查询Post示例
     * @param departmentDTO 查询参数对象
     * @param pageNum
     * @param pageSize
     * @return
     */
```

```
    Page<DepartmentDTO> findPaginationPage(DepartmentDTO departmentDTO, Integer pageNum,
Integer pageSize);

    /**
     * 分页查询Get示例
     * @param name 查询参数
     * @param currentPage
     * @param pageSize
     * @return
     */
    Page<DepartmentDTO> findPaginationPage(String name, Integer currentPage, Integer
pageSize);

}
```

服务接口实现层：

```
/**
 * 分页示例
 * @author litian
 */
@Service
@Slf4j
public class PaginationServiceImpl implements PaginationService {

    @Resource
    private JdbcQueryManager jdbcQueryManager;

    @Override
    public Page<DepartmentDTO> findPaginationPage(DepartmentDTO departmentDTO, Integer
currentPage, Integer pageSize) {
        PageParam pageParam = PageParam.create(currentPage, pageSize);
        return jdbcQueryManager.queryForPage("departmentService_findByPage",
                InputParameters.create("departmentName", departmentDTO == null ?  null :
departmentDTO.getName()),
                Department.class,
                pageParam,
                Department::convert);
    }

    @Override
    public Page<DepartmentDTO> findPaginationPage(String name, Integer currentPage, Integer
pageSize) {
        PageParam pageParam = PageParam.create(currentPage, pageSize);
        return jdbcQueryManager.queryForPage("departmentService_findByPage",
                InputParameters.create("departmentName", name),
                Department.class,
                pageParam,
                Department::convert);
    }

}
```

至于sql语句一并给出：

```
<?xml version="1.0" encoding="UTF-8" ?>
<dal:dalScripts domainCode="eco_sample" xmlns:dal="urn:com.gsww.cd.commons.dal:commons-
dal:scripts">
    <dal:select id="departmentService_findByName">
```

```
        select
        id, department_name, deleted_flag, create_time, update_time
        from eco_department department
        where department_name = #{departmentName}
        and deleted_flag = 0
    </dal:select>

    <dal:select id="departmentService_findByPage">
        select
        id, department_name, deleted_flag, create_time, update_time
        from eco_department department
        where deleted_flag = 0
        <dal:if test="departmentName != null and departmentName != ''">
            and department.department_name = #{departmentName}
        </dal:if>
    </dal:select>
</dal:dalScripts>
```

前端组件

前端采用了 `EP-UI` 的 `EpPagenation` 组件，区别于iview的分页组件，通过 `onchange` 事件返回的参数 `curpage` ，`pagesize` 以及其他用户参数查询数据，查询完成后通过 `callback(当前页数据条数，数据总条数，当前页码)` 重新更新分页显示，对 `ininpage` 属性取反重新初始化分页。

```
<template>
  <EpPagenation @onchange="pageevent"  :pageinit="pageinit" :isshowgopage="isshowgopage">
</EpPagenation>
</template>
<script>
  export default {
    methods: {
      initpage() {
        // 重新初始化分页，会从第一页开始发起请求
        this.pageinit = !this.pageinit
      },
      pageevent(curpage, pagesize, $callback) {
        console.log(curpage, pagesize)
        // 通过curpage，  pagesize以及其他参数querydata;
        // $callback(当前页数据条数，数据总条数，当前页码);当前页数据条数，数据总条数为必要参数
        // $callback(0, 0)
        // eslint-disable-next-line startdard/no-callback-literal
        $callback(10, 100)
      }
    }
  }
</script>
```

在测试过程中如果参数直接命名为 `callback` 会没有反应，换成其他参数均能测试通过，所以上面换成了 `$callback` 。

API

`EpPagenation props`

属性	说明	类型	默认值
`pageinit`	初始化分页，每次初始化时候对值取反即可	Boolean	True
`pageconfig`	分页配置	Object	{ `isshowtotal: true, // 是否显示总条数` `isshowpagesize: true, // 是否显示当前页的数据条数` `isshowpagesizeselect: true, // 是否可以改变pageszie` `btncount: 7, // 最多显示的页码个数` `isshowfirstorlastpage: true, // 是否显示首页或尾页` `isshowgopage: true, //是否显示输入框` `pagesize: 10 // 默认每页数据条数` }

EpPagenation event

事件名	说明	返回值
`onchange`	分页改变是触发	Curpage,pagesize,callback,其中callback为回调函数，数据请求完成后调用；callback(当前页数据条数，数据总条数，当前页码)

EpPagenation slot

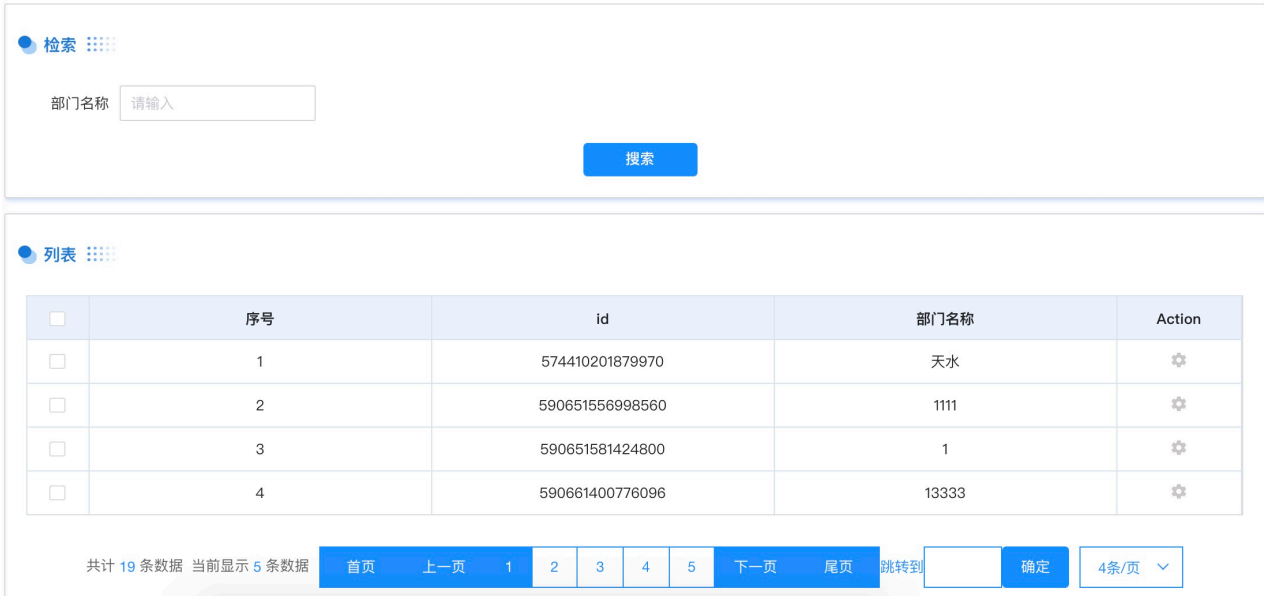
插槽名	说明
–	–

前端完整示例

```
<template>
  <EpPagenation @onchange="pageevent"   :pageinit="pageinit" :pageconfig="pageconfig">
</EpPagenation>
</template>
import { post, get } from './../../../../api/http'
export default {
  name: 'list',
  data() {
    return {
      page: {
        pageNum: 1,
        pageSize: 1,
        totalRows: 0
      },
      params: {
        name: ''
      },
      pageconfig: {
        pagesize: 4
      }
    }
  },
  methods: {
    pageevent(curpage, pagesize, $callback) {
```

```
      if (curpage === 0) {
        return
      }
      // 通过curpage, pagesize以及其他参数querydata;
      // callback(当前页数据条数, 数据总条数,当前页码); 当前页数据条数, 数据总条数为必要参数
      this.findByPage(curpage, pagesize, $callback)
    },
    findByPage(currentPage, pageSize, $callback) {
      let $this = this
      post(
        'http://localhost:30033/core-api/eco/sample/v1/pagination/findPaginationPage/' +
          currentPage + '/' + pageSize, this.params
      ).then(
        function(data) {
          let result = data.data
          if (result) {
            $this.moduleTableData = result.items
            $callback(result.totalPage, result.totalRows, result.currentPage)
          }
        },
        function(e) {
          alert(e)
        }
      )
    }
  },
  mounted() {
    this.init()
  }
}
</script>
```

效果图



遗留问题

分页组件的 `queryForMapPage` 方法还存在一些个问题，需要进行进一步验证。前端组件还没有找到可以通过点击除了分页按钮之外的按钮触发分页事件的方法，也需要和前端人员进行确认。