我再也不想写@RequestParam(”current”)了

==========

起因

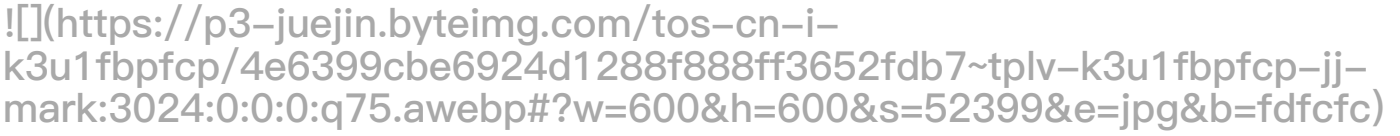
--

众所周知，分页表格属于是各种web系统的常客，系统中每个功能几乎都至少对应了一个分页表格。

所以当我在写各种Controller时，敲下的第一个接口往往是`/xxx/page`,同时总是需要接上各种分页参数

```
@GetMapping(”/xxx/page”)
public Page<User> page(@RequestParam(”current”) Integer current,
                       @RequestParam(”size”) Integer size,
                       @RequestParam(value = ”column”, required = false)
String column,
                       @RequestParam(value = ”is_asc”, required = false)
Boolean isAsc,
                       @RequestParam(value = ”keyword”, required = false)
String keyword) {
    // xxx
    return page;
}
```

真是公公又式式啊，不过作为一个二手代码搬运工，这样公式化的接口之前都是直接cv的，方便快捷不用过脑。不过最近闲下来了，回顾之前的代码，看着这一长串的参数接收，越看越难受。



思路

--

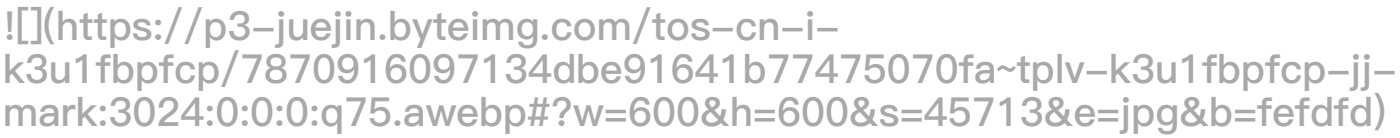
首先想解决的问题是在Controller层要接收的参数太多了，且部分参数基本固定，在查询时我还需要手动把他们转换成一个Page对象作为查询入参（MybatisPlus）

思考了一下常见的分页表格，一般由2部分组成。

* 固定部分：分页参数（页号、页码、排序列、正序倒序）
* 不固定部分：筛选条件（关键词搜索，其他各种筛选）

筛选条件的部分各个功能都不一样，封装太难统一了，这次不如就把固定部分封装一下好了，一下干掉4个参数，这样传参的时候也能清爽很多。并且通常业务不存在需要2个列同时进行排序，那就直接写死，1个参数放排序列名，1个参数放 正序/倒序

不要问我为什么不直接用post请求放body里，哥们表格从来就是get请求，祖宗之法不可变~



开搞
--

定义统一的参数对象

```
@Data
public class PageParams {
    private Integer current;
    private Integer size;
    private String column;
    private Boolean isAsc;

    public <T> IPage<T> toPage() {
        Page<T> page = Page.of(this.current, this.size);
        if (this.column != null) {
            boolean isAsc = this.isAsc == null || this.isAsc;
```

```
            page.addOrder(new OrderItem(this.column, isAsc));
        }
        return page;
    }
}
```

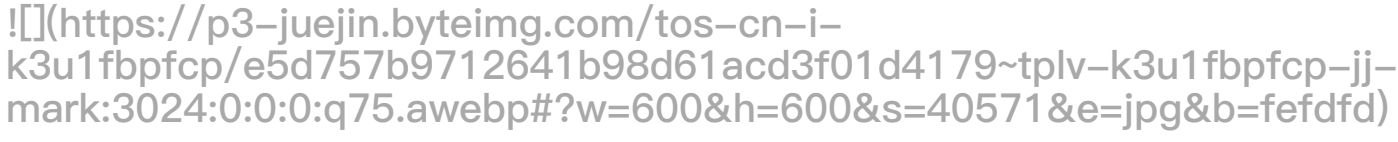
这里额外除了定义分页参数，也加了一个通用的方法，用于把这几个参数转换成MybatisPlus的分页参数对象，这样查询的时候就能直接传参了

自定义注解

```
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface PageParam {

    String currentName() default "current";

    String sizeName() default "size";

    String columnName() default "column";

    String isAscName() default "is_asc";
}
```

要让Spring更准确的注入参数，也减少引发bug的概率，自定义注解是必不可少的

虽然通常这几个参数的名称都是一样的，但是为了防止需求方突然犯病，还是留下了自定义参数名称的地方



参数注入接口

```
public class PageParamArgumentResolver implements
HandlerMethodArgumentResolver {

    @Override
    public boolean supportsParameter(MethodParameter parameter) {
        if (!parameter.hasParameterAnnotation(PageParam.class)) {
            return false;
        }
        Class<?> type = parameter.getParameterType();
        return type.isAssignableFrom(PageParams.class) ||
type.isAssignableFrom(Page.class);

    }

    @Override
    public Object resolveArgument(MethodParameter parameter,
ModelAndViewContainer mavContainer,
                                  NativeWebRequest webRequest,
WebDataBinderFactory binderFactory) throws Exception {
        PageParam anno =
parameter.getParameterAnnotation(PageParam.class);
        if (anno == null) {
            return null;
        }
        Class<?> type = parameter.getParameterType();

        Integer current =
getIntParamNotNullByWebRequest(anno.currentName(), webRequest);
        Integer size = getIntParamNotNullByWebRequest(anno.sizeName(),
webRequest);
        String column = webRequest.getParameter(anno.columnName());
        String isAsc = webRequest.getParameter(anno.isAscName());

        PageParams pageParam = new PageParams();
        pageParam.setCurrent(current);
        pageParam.setSize(size);
        if (StringUtils.isNotBlank(column)) {
            pageParam.setColumn(column);
            // 默认是升序排列
pageParam.setIsAsc(!Boolean.FALSE.toString().equalsIgnoreCase(isAsc));

        }
        if (type.isAssignableFrom(Page.class)) {
            return pageParam.toPage();
        }
```

```
        return pageParam;
    }

    private Integer getIntParamNotNullByWebRequest(String name,
NativeWebRequest webRequest) {
        String value = webRequest.getParameter(name);
        if (StringUtils.isBlank(value)) {
            throw new IllegalArgumentException(String.format("参数: %s 为
空", name));
        }
        return Integer.parseInt(value);
    }
}
```

一顿搜索之后发现Spring很贴心的为我们准备了`HandlerMethodArgumentResolver`接口实现参数的注入.

这里我打算支持2种参数，我们自定义的分页参数和MybatisPlus的分页参数都支持，进一步减少一行参数转换

纯业务需求默认必传页号和页码，不想加也行

添加WebConfig

```
@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void
addArgumentResolvers(List<HandlerMethodArgumentResolver>
resolvers) {
        resolvers.add(new PageParamArgumentResolver());
    }
}
```

大功告成！！！

k3u1fbpfcp/5ba17e869686400ca014b742c7b2921f~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=600&h=600&s=27341&e=jpg&b=fde193)

```
    @GetMapping("/params1")
    public String params1(@PageParam PageParams pageParams) {
        return "hello";
    }

    @GetMapping("/params2")
    public String params2(@PageParam Page<User> page) {
        return "hello";
    }
```

使用的时候直接用注解+分页参数，完美解决

原文链接: https://juejin.cn/post/7360497239871094824