

1. 学习 datagrid 数据列表组件

常用属性：

属性名	参数	备注
url	string	一个 URL 从远程站点请求数据。
columns	array	DataGrid 列配置对象，详见列属性说明中更多的细节。
pagination	boolean	如果为 true，则在 DataGrid 控件底部显示分页工具栏。
toolbar	array,select or	顶部工具栏的 DataGrid 面板。可能的值： 1) 一个数组，每个工具属性都和 linkbutton 一样。 2) 选择器指定的工具栏。
singleSelect	boolean	如果为 true，则只允许选择一行。

● 列属性

属性名	参数	备注
field	string	列字段名称。
title	string	列标题文本。
width	number	列的宽度。如果没有定义，宽度将自动扩充以适应其内容。
align	string	指明如何对齐列数据。可以使用的值有： 'left','right','center'。
checkbox	boolean	如果为 true，则显示复选框。该复选框列固定宽度。
formatter	function	单元格 formatter(格式化器)函数，带 3 个参数： value: 字段值。 row: 行记录数据。 index: 行索引。

常用方法：

方法名	参数	备注
enable	none	允许拖动。
disable	none	禁止拖动。

演示案例：

```
<!DOCTYPE html>

<html>

  <head>

    <title>datagrid 数据表格</title>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <link rel="stylesheet" type="text/css"
href="../easyui/themes/default/easyui.css">

    <link rel="stylesheet" type="text/css" href="../easyui/themes/icon.css">

    <script type="text/javascript" src="../easyui/jquery.min.js"></script>

    <script type="text/javascript" src="../easyui/jquery.easyui.min.js"></script>

    <script type="text/javascript"
src="../easyui/locale/easyui-lang-zh_CN.js"></script>

    <style type="text/css">

      body{

        font-family: "Microsoft YaHei";

      }

    </style>

    <script type="text/javascript">

      $(document).ready(function(){

        $("#dg").datagrid({

          //url: 代表远程的 url 地址，这个 url 地址必须返回符合数据表格的 json 结
```

```

    构:{ "total":10, "rows":[{},{ }...] }

    url:"datagrid_data1.json",

    //columns: 代表填充数据表单的列数据（获取 rows 的数据）

    //field:代表字段名称

    //title:代表字段标题

    //width:字段宽度

    //formatter: 格式化列内容（常用）

    columns:[[

        {field:"productid",title:"商品编号",width:100},

        {field:"productname",title:"商品名称",width:150},

        {field:"unitcost",title:"商品单价",width:100},

        {field:"status",title:"商品库存状态

",width:100,formatter:function(value,row,index){

            //value: 当前列字段值, row: 当前行数据, index:当前行的索引

            return value=="P"? "<font color=green>有货</font>":"<font

color=red>缺货</font>";

        }},

        {field:"listprice",title:"商品零售价",width:100}

    ]],

    //工具条, 显示到 datagrid 的左上方

    /*

    toolbar:[

        {

            iconCls:"icon-save",

            text:"新增",

            //handler: 按钮点击触发事件

            handler:function(){

                alert("点击了新增");

            }

        }

    ]

}

```

```
        },
    },
    {
        iconCls:"icon-edit",
        text:"编辑"
    },
    {
        iconCls:"icon-remove",
        text:"删除"
    }
]
*/

toolbar:"#toolbar",
//显示分页工具条
pagination:true,
});

$("#btn1").click(function(){
    //全选
    $("#dg").datagrid("selectAll");
});

$("#btn2").click(function(){
    //获取到选择的行
    var rows = $("#dg").datagrid("getSelections");
    $(rows).each(function(i){
        //获取行的字段值
        alert(this.productid); // this:代表当前行对象
    });
});
});
```

```
});  
  
</script>  
  
</head>  
  
<body>  
  
  <div id="toolbar">  
  
    <a href="#" class="easyui-Linkbutton" data-options="iconCls:'icon-save'">添  
加</a>  
  
    <a href="#" class="easyui-Linkbutton" data-options="iconCls:'icon-edit'">修  
改</a>  
  
  </div>  
  
  <table id="dg"></table>  
  
  <br>  
  <hr>  
  
  <p>toolbar 添加 新增、编辑工具栏按钮</p>  
  
  <p>pagination 显示分页导航条</p>  
  
  <input id="btn2" type="button" value="getSelections 返回所有被选中的行并输出">  
  
</body>  
</html>
```

2. 案例：完商品列表分页展示

2.1. 建立商品表

-- 商品表











```
CREATE TABLE t_product(
```

```
id INT PRIMARY KEY AUTO_INCREMENT,  
  
NAME VARCHAR(50),  
  
price DOUBLE,  
  
current_price DOUBLE,  
  
description VARCHAR(2000)  
);
```

2.2. 导入商品表测试数据

```
insert into `t_product`(`id`,`name`,`price`,`current_price`,`description`) values (1,'华为手机',2000,1699,'华为手机'),(2,'乐视手机',1200,1100,'乐视手机'),(3,'联想笔记本',4999,3999,'联想笔记本'),(4,'华硕笔记本',3999,2999,'华硕笔记本'),(5,'iphone4S',2999,2599,'iphone4S'),(6,'iphone5S',3999,2599,'iphone5S'),(7,'iphone6S',4999,3599,'iphone6S'),(8,'创维电视机',3500,3250,'创维电视机'),(9,'小米 5',4000,3500,'小米 5'),(10,'红米 2S',1200,1000,'红米 2S'),(11,'vivo 手机',2999,2700,'vivo 手机');
```

2.3. 导入 Hibernate 的 jar 包和工具类

-  antlr-2.7.7.jar
-  dom4j-1.6.1.jar
-  geronimo-jta_1.1_spec-1.1.1.jar
-  hibernate-commons-annotations-5.0.1.Final.jar
-  hibernate-core-5.0.7.Final.jar
-  hibernate-jpa-2.1-api-1.0.0.Final.jar
-  jandex-2.0.0.Final.jar
-  javassist-3.18.1-GA.jar
-  jboss-logging-3.3.0.Final.jar
-  mysql-connector-java-5.1.7-bin.jar

```
//工具类
```

```
public class HibernateUtil {
```

```
private static Configuration config;

private static SessionFactory sessionFactory;

//初始化 1 次就好了

static{

    config = new Configuration();

    config.configure();

    sessionFactory = config.buildSessionFactory();

}

/**
 * 获取 Session
 */
public static Session getSession(){
    return sessionFactory.openSession();
}

}
```

2.4. 编写 hibernate 核心配置

```
<?xml version="1.0"?>

<!DOCTYPE hibernate-configuration PUBLIC

    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>
```

```
<!-- 第一部分：数据库连接（必要） -->

<property name="hibernate.connection.driver_class">

    com.mysql.jdbc.Driver

</property>

<property name="hibernate.connection.url">

    jdbc:mysql://localhost:3306/ecshop

</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">

    org.hibernate.dialect.MySQL5Dialect

</property>


<!-- 第二部分：扩展参数 （可选） -->

<!-- 是否在控制台输出生成的 sql 语句 （默认为 false） -->

<property name="hibernate.show_sql">true</property>

<!-- 是否在控制台格式化 sql 语句 （默认为 false） -->

<property name="hibernate.format_sql">true</property>

<!-- 是否需要 hibernate 维护表结构

    create: 每次都会自动建立表结构

    update: 如果没有表，就会建立表结构；如果有表，更新表结构

-->

<property name="hibernate.hbm2ddl.auto">update</property>


<!-- 第三部分：映射文件路径 （可选） -->

<mapping class="cn.sm1234.domain.Product"/>

</session-factory>

</hibernate-configuration>
```


2.5. 编写实体和映射

```
@Entity

@Table(name="t_product")

public class Product {

    @Id

    @GeneratedValue(strategy=GenerationType.IDENTITY)

    @Column(name="id")

    private Integer id;

    @Column(name="name")

    private String name;

    @Column(name="price")

    private Double price;

    @Column(name="current_price")

    private Double currentPrice;

    @Column(name="description")

    private String description;

    public Integer getId() {

        return id;

    }

    public void setId(Integer id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

}
```

```
}

    public void setName(String name) {

        this.name = name;

    }

    public Double getPrice() {

        return price;

    }

    public void setPrice(Double price) {

        this.price = price;

    }

    public Double getCurrentPrice() {

        return currentPrice;

    }

    public void setCurrentPrice(Double currentPrice) {

        this.currentPrice = currentPrice;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

}
```

2.6. 编写 Dao

```
public class ProductDaoImpl implements ProductDao {
```

```
@Override
```

```
public List<Product> findByPage(Integer page, Integer rows) {  
    Session session = HibernateUtil.getSession();  
  
    try {  
        Query query = session.createQuery("from Product");  
        query.setFirstResult((page-1)*rows);  
        query.setMaxResults(rows);  
        return query.list();  
    } catch (Exception e) {  
        e.printStackTrace();  
        throw new RuntimeException(e);  
    } finally {  
        session.close();  
    }  
}
```

```
@Override
```

```
public Long findByTotalCount() {  
    Session session = HibernateUtil.getSession();  
  
    try {  
        Query query = session.createQuery("select count(1) from Product");  
        return (Long) query.uniqueResult();  
    } catch (Exception e) {  
        e.printStackTrace();  
        throw new RuntimeException(e);  
    } finally {  
        session.close();  
    }  
}
```














```
}
```

2.7. 编写 Service

```
public class ProductServiceImpl implements ProductService {  
  
    private ProductDao productDao = new ProductDaoImpl();  
  
    @Override  
    public List<Product> findByPage(Integer page, Integer rows) {  
        return productDao.findByPage(page, rows);  
    }  
  
    @Override  
    public Long findByTotalCount() {  
        return productDao.findByTotalCount();  
    }  
}
```

2.8. 导入 Struts2 的 jar 包

名称

-  asm-3.3.jar
-  asm-commons-3.3.jar
-  asm-tree-3.3.jar
-  commons-fileupload-1.3.1.jar
-  commons-io-2.2.jar
-  commons-lang3-3.2.jar
-  freemarker-2.3.22.jar
-  javassist-3.11.0.GA.jar
-  log4j-api-2.2.jar
-  log4j-core-2.2.jar
-  ognl-3.0.6.1.jar
-  struts2-core-2.3.24.3.jar
-  xwork-core-2.3.24.3.jar

2.9. 配置 web.xml 启动 struts2

```
<!-- 配置核心控制器（过滤器） -->

<filter>

    <filter-name>struts2</filter-name>

    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

    <init-param>

        <param-name>struts.action.extension</param-name>

        <param-value>action,,</param-value>

    </init-param>

</filter>

<filter-mapping>

    <filter-name>struts2</filter-name>

    <url-pattern>/*</url-pattern>
```

</filter-mapping>

2.10. 编写 Action 查询商品数据

```
public class ProductAction extends ActionSupport{

    //接收当前页码 和 页面大小

    private Integer page;

    private Integer rows;

    public void setPage(Integer page) {

        this.page = page;

    }

    public void setRows(Integer rows) {

        this.rows = rows;

    }

    private ProductService productService = new ProductServiceImpl();

    /**
     * 查询商品
     */

    public String listByPage(){

        if(page==null){

            page = 1;

        }

        if(rows==null){

            rows = 10;

        }

        List<Product> list = productService.findByPage(page, rows);

        Long totalCount = productService.findTotalCount();

    }

}
```

```
Map<String, Object> result = new HashMap<String, Object>();

result.put("rows", list);

result.put("total", totalCount);

ActionContext.getContext().getValueStack().push(result);

return SUCCESS;
}

}
```

2.11. 配置 struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE struts PUBLIC

    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

    <package name="ecshop" extends="json-default" namespace="/">

        <action name="product_*" class="cn.sm1234.web.ProductAction" method="{1}">

            <result type="json"></result>

        </action>

    </package>

</struts>
```

2.12. 编写 product_manage.jsp 分页显示商品

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```
<html>

<head>

  <title>商品管理</title>

  <meta http-equiv="pragma" content="no-cache">

  <meta http-equiv="cache-control" content="no-cache">

  <meta http-equiv="expires" content="0">

  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">

  <meta http-equiv="description" content="This is my page">

  <script type="text/javascript" src="../easyui/jquery.min.js"></script>

  <link rel="stylesheet" type="text/css" href="../easyui/themes/icon.css">

  <link rel="stylesheet" type="text/css"
href="../easyui/themes/default/easyui.css">

  <script type="text/javascript" src="../easyui/jquery.easyui.min.js"></script>

  <script type="text/javascript"
src="../easyui/locale/easyui-lang-zh_CN.js"></script>

</head>

<body>

  <table id="list"></table>

  <script type="text/javascript">

    $(function(){

      $("#list").datagrid({

        url:"product_listByPage.action",

        columns:[[

          {

            field:"id",
```



```
                title:"编号",
                width:30
            },
            {
                field:"name",
                title:"商品名称",
                width:200
            },
            {
                field:"price",
                title:"商品原价",
                width:200
            },
            {
                field:"currentPrice",
                title:"商品优惠价",
                width:200
            },
            {
                field:"description",
                title:"商品描述",
                width:200
            }
        ]],
        pagination:true
    });

});

</script>
```

```
</body>
</html>
```

3. 学习 window 窗口组件

3.1.window 窗口组件

常用属性：

属性名	参数	备注
title	string	窗口的标题文本。
closed	boolean	定义是否可以关闭窗口。

常用方法：

方法名	参数	备注
open	forceOpen	在'forceOpen'参数设置为 true 的时候，打开面板时将跳过'onBeforeOpen'回调函数。
close	forceClose	在'forceClose'参数设置为 true 的时候，关闭面板时将跳过'onBeforeClose'回调函数。

演示案例：

```
<!DOCTYPE html>

<html>

<head>

<title>window 窗口</title>

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

<link rel="stylesheet" type="text/css"

      href="../easyui/themes/default/easyui.css">

<link rel="stylesheet" type="text/css" href="../easyui/themes/icon.css">

<script type="text/javascript" src="../easyui/jquery.min.js"></script>
```

```
<script type="text/javascript" src="../easyui/jquery.easyui.min.js"></script>

<script type="text/javascript"

    src="../easyui/locale/easyui-lang-zh_CN.js"></script>

<style type="text/css">

body {

    font-family: "Microsoft YaHei";

}

p {

    color: white;

    background-color: #0092E7;

}

</style>

<script type="text/javascript">

    $(document).ready(function() {

        $("#win").window({

            width:200,

            height:200,

            title:"标题",

            //模式化窗口（窗口遮罩效果）

            modal:false,

            //关闭状态

            closed:true

        });

        $("#btn1").click(function(){

            //打开窗口

            $("#win").window("open");

        });

    });

}
```

```
$( "#btn2" ).click(function(){  
    //关闭窗口  
    $( "#win" ).window("close");  
});  
});  
</script>  
</head>  
  
<body>  
    <p>HTML 标签创建</p>  
    <!-- <div class="easyui-window" title="My Window"  
        style="width:600px;height:400px"  
        data-options="iconCls:'icon-save',modal:true">  
        Window Content  
    </div> -->  
    用户名: <input type="text" name="name"/>  
  
    <br>  
    <p>JavaScript 创建</p>  
    <div id="win"></div>  
  
    <br>  
    <hr>  
    <input id="btn1" type="button" value="open 打开窗口">  
    <input id="btn2" type="button" value="close 关闭窗口">  
</body>  
</html>
```

3.2. dialog 对话框窗口组件

常用属性：

属性名	参数	备注
toolbar	array,selector	设置对话框窗口顶部工具栏，可用值有： 1) 一个数组，每一个工具栏中的工具属性都和 linkbutton 相同。 2) 一个选择器指定工具栏。
buttons	array,selector	对话框窗口底部按钮，可用值有： 1) 一个数组，每一个按钮的属性都和 linkbutton 相同。 2) 一个选择器指定按钮栏。

常用方法：

和 window 一样！

演示案例：

```
<!DOCTYPE html>

<html>

  <head>

    <title>dialog 对话框窗口</title>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <link rel="stylesheet" type="text/css"
href="../../easyui/themes/default/easyui.css">

    <link rel="stylesheet" type="text/css" href="../../easyui/themes/icon.css">

    <script type="text/javascript" src="../../easyui/jquery.min.js"></script>

    <script type="text/javascript" src="../../easyui/jquery.easyui.min.js"></script>

    <script type="text/javascript"
src="../../easyui/locale/easyui-lang-zh_CN.js"></script>
```

```
<style type="text/css">

body{

    font-family: "Microsoft YaHei";

}

p {

    color: white;

    background-color: #0092E7;

}

</style>

<script type="text/javascript">

$(document).ready(function(){

    $("#dd").dialog({

        width:200,

        height:200,

        title:"对话框窗口",

        //工具条

        toolbar:[

            {

                iconCls:"icon-save",

                text:"添加"

            },{

                iconCls:"icon-edit",

                text:"修改"

            }

        ],

        //按钮栏

        buttons:[

            {
```

```

        iconCls:"icon-save",
        text:"保存"
    },{
        iconCls:"icon-clear",
        text:"重置"
    }
]
});
});
</script>

</head>

<body>

<p>HTML 标签创建</p>

<!-- <div class="easyui-dialog" title="My Dialog"
style="width:400px;height:200px;"
data-options="iconCls:'icon-save',resizable:true,modal:true">
    Dialog Content.
</div> -->

<br>

<p>JavaScript 创建</p>

<div id="dd">Dialog Content.</div>

<br>

</body>

</html>

```

3.3. messenger 消息框组件

常用方法：

方法名	参数	备注
<code>\$.messenger.alert</code>	<code>title, msg, icon, fn</code>	显示警告窗口。参数： title : 在头部面板显示的标题文本。 msg : 显示的消息文本。 icon : 显示的图标图像。
<code>\$.messenger.confirm</code>	<code>title, msg, fn</code>	可用值有：显示一个包含“确定”和“取消”按钮的确认消息窗口。 参数： title : 在头部面板显示的标题文本。 msg : 显示的消息文本。 fn(b) : 当用户点击“确定”按钮的时候将传递一个 true 值给回调函数，否则传递一个 false 值。
<code>\$.messenger.prompt</code>	<code>title, msg, fn</code>	显示一个用户可以输入文本的并且带“确定”和“取消”按钮的消息窗体。参数： title : 在头部面板显示的标题文本。 msg : 显示的消息文本。 fn(val) : 在用户输入一个值参数的时候执行的回调函数。
<code>\$.messenger.show</code>	<code>options</code>	在屏幕右下角显示一条消息窗口。该选项参数是一个可配置的

		<p>对象：</p> <p>showType: 定义将如何显示该消息。可用值有： null,slide,fade,show。默认： slide。</p> <p>showSpeed: 定义窗口显示的过度时间。默认：600 毫秒。</p> <p>width: 定义消息窗口的宽度。默认：250px。</p> <p>height: 定义消息窗口的高度。默认：100px。</p> <p>title: 在头部面板显示的标题文本。</p> <p>msg: 显示的消息文本。</p> <p>style: 定义消息窗体的自定义样式。</p> <p>timeout: 如果定义为 0，消息窗体将不会自动关闭，除非用户关闭他。如果定义成非 0 的数，消息窗体将在超时后自动关闭。默认：4 秒。</p>
<code>\$.messenger.progress</code>	options or method	<p>显示一个进度消息窗体。</p> <p>属性定义为：</p> <p>title: 在头部面板显示的标题文本。默认：空。</p> <p>msg: 显示的消息文本。默认：空。</p> <p>text: 在进度条上显示的文本。默认：undefined。</p>

		<p>interval: 每次进度更新的间隔时间。默认：300 毫秒。</p> <p>方法定义为：</p> <p>bar: 获取进度条对象。</p> <p>close: 关闭进度窗口。</p>
--	--	---

演示案例：

```
<!DOCTYPE html>

<html>

  <head>

    <title>messenger 消息窗口</title>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <link rel="stylesheet" type="text/css"
href="../easyui/themes/default/easyui.css">

    <link rel="stylesheet" type="text/css" href="../easyui/themes/icon.css">

    <script type="text/javascript" src="../easyui/jquery.min.js"></script>

    <script type="text/javascript" src="../easyui/jquery.easyui.min.js"></script>

    <script type="text/javascript"
src="../easyui/locale/easyui-lang-zh_CN.js"></script>

    <style type="text/css">

      body{

        font-family: "Microsoft YaHei";

      }

    </style>

    <script type="text/javascript">

      $(document).ready(function(){

        $("#btn1").click(function(){
```

```
$.messenger.alert("提示","操作成功","info");

});

$("#btn2").click(function(){

$.messenger.confirm("提示","确认删除数据吗?",function(value){

    if(value){

        //确定

        alert("删除了数据");

    }

});

});

$("#btn3").click(function(){

$.messenger.prompt("提示","请输入姓名",function(value){ //value:代表
输入的内容

    if(value){

        alert("你的姓名: "+value);

    }

});

});

$("#btn4").click(function(){

$.messenger.show({

    title:"提示",

    msg:"老王上线啦",

    //消失的时间

    timeout:2000

});

});
```

```
$( "#btn5" ).click(function(){  
    $.messager.progress({  
        title:"提示",  
        msg:"数据连接中",  
    });  
});  
});  
  
</script>  
  
</head>  
  
<body>  
  
    <br>  
  
    <hr>  
  
    <input id="btn1" type="button" value="alert 警告窗口">  
  
    <input id="btn2" type="button" value="confirm 确认窗口">  
  
    <input id="btn3" type="button" value="prompt 输入窗口">  
  
    <input id="btn4" type="button" value="show 消息窗口">  
  
    <input id="btn5" type="button" value="progress 消息窗口">  
  
</body>  
</html>
```

4. 案例：完成商品编辑窗口制作

```
<div id="toolbar">  
    <a id="addBtn" href="#" class="easyui-Linkbutton"
```

```
data-options="iconCls:'icon-save'">添加</a>

<a id="editBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-edit'">修改</a>

<a id="deleteBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-remove'">删除</a>

</div>
```

```
        title: "商品描述",
        width: 200
    },
    ],
    pagination: true,
    toolbar: "#toolbar"
});
```

```
<!-- 编辑窗口 -->

<div id="editWin" class="easyui-window" data-options="title:'商品编辑窗口'
,width:350,height:300,closed:true">

    <form id="editForm" action="" method="post">

        <table>

            <tr>

                <td>商品名称</td>

                <td><input type="text" name="name"/></td>

            </tr>

            <tr>

                <td>商品原价</td>

                <td><input type="text" name="price"/></td>

            </tr>

            <tr>

                <td>商品优惠价</td>

                <td><input type="text" name="currentPrice"/></td>

            </tr>
```

```

        <tr>

            <td>商品描述</td>

            <td>

                <textarea rows="5" cols="20" name="description"></textarea>

            </td>

        </tr>

        <tr>

            <td colspan="2">

                <a id="save" href="#" class="easyui-Linkbutton"
data-options="iconCls:'icon-save'">保存</a>

            </td>

        </tr>

    </table>

</form>

</div>
```

//点击添加按钮，弹出编辑窗口

```

        $("#addBtn").click(function(){

            $("#editWin").window("open");

        });
```

5. 学习 Form 表单组件

5.1. validatebox 验证框组件

常用属性：

属性名	参数	备注
required	boolean	定义为必填字段。
missingMessage	string	当文本框未填写时出现的提示信息。

validType	string,array	定义字段验证类型，比如：email, url 等等。可用值有： 1).一个有效类型字符串运用一个验证规则。 2).一个有效类型数组运用多个验证规则
invalidMessage	string	当文本框的内容被验证为无效时出现的提示。

演示案例：

```
$(document).ready(function(){
    //自定义 minLength 的验证类型
    $.extend($.fn.validatebox.defaults.rules, {
        //minLength: 类型名称，在引用类型的时候使用该名称
        minLength: {
            //validator: 验证的逻辑
            // value: 用户输入的内容
            // param: 参数，是一个数组类型，在使用的时候会把参数值传入，例如
minLength[4]
            validator: function(value, param){
                return value.length >= param[0];
            },
            //message: 验证失败时的提示信息
            // {0}: 获取参数列表的第几个参数
            message: '请输入至少 {0}个字符！'
        }
    });

    $("input[name=name]").validatebox({
```

```
//必填提示

missingMessage:"姓名必须填写",

validType:"length[4,12]",

invalidMessage:"姓名必须在 4-12 个字符之间"

});

$( "input[name=email]").validatebox({

    required:true,

    validType:"minLength[4]"

});

});

<form id="ff">

    <table align="center" cellpadding="0" cellspacing="0" style="width:80%">

        <tr>

            <td align="right" width="180">姓名: </td>

            <td><input type="text" name="name" class="easyui-validatebox"

data-options="required:true"/></td>

        </tr>

        <tr>

            <td align="right" width="180">电子邮箱: </td>

            <td><input type="text" name="email"/></td>

        </tr>

        <tr>

            <td align="right" width="180">主题: </td>

            <td><input type="text" name="theme" value="默认"/></td>

        </tr>

        <tr>

            <td align="right" width="180">性别: </td>

            <td>
```



```
<select name="gender">
    <option></option>
    <option value="1">男</option>
    <option value="0">女</option>
</select>
</td>
</tr>
</table>
</form>
```

5.2. form 表单组件

常用方法：

方法名	参数	备注
load	data	读取记录填充到表单中。数据参数可以是一个字符串或一个对象类型，如果是字符串则作为远程 URL，否则作为本地记录。
submit	options	执行提交操作，该选项的参数是一个对象，它包含以下属性： url: 请求的 URL 地址。 onSubmit: 提交之前的回调函数。 success: 提交成功后的回调函数。
validate	none	做表单字段验证，当所有字段都有效的时候返回 true。该方法使用 validatebox(验证框)插件。
clear	none	清除表单数据。
reset	none	重置表单数据。

演示案例：

```
<!DOCTYPE html>
```

```
<html>

<head>

<title>form 表单</title>

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

<link rel="stylesheet" type="text/css"
href="../easyui/themes/default/easyui.css">

<link rel="stylesheet" type="text/css" href="../easyui/themes/icon.css">

<script type="text/javascript" src="../easyui/jquery.min.js"></script>

<script type="text/javascript" src="../easyui/jquery.easyui.min.js"></script>

<script type="text/javascript"
src="../easyui/locale/easyui-lang-zh_CN.js"></script>

<style type="text/css">

body{

    font-family: "Microsoft YaHei";

}

p {

    color: white;

    background-color: #0092E7;

}

</style>

<script type="text/javascript">

$(document).ready(function(){

    $("#btn1").click(function(){

        //load 方法填充表单数据

        //本地填充

        /*

        $("#fff").form("load",{
```

```
        name:"eric",
        email:"eric@qq.com",
        theme:"eric 的主题",
    });
    */

    //远程获取数据填充

    //注意: url 地址必须返回一个 json 数据:{name:xxx,email:xxx}

    $("#fff").form("load","../GetDataServlet");
    });

    $("#btn2").click(function(){
        //清空

        $("#fff").form("clear");
    });

    $("#btn3").click(function(){
        //重置

        $("#fff").form("reset");
    });

    $("#btn4").click(function(){
        //提交表单

        $("#fff").form("submit",{

            //url:表单提交的地址

            url:"../FormDataServlet",

            //onSubmit:提交前,通常提交前判断验证是否全部通过啦,如果全部通过了,
            才能提交表单

            onSubmit:function(){
```

```
//判断验证是否全部通过

        return $("#ff").form("validate");
    },

    //success:服务器执行完毕

    success:function(data){//data:服务器返回的数据

        //把 data 转出为 json

        data = eval("(" + data + ")");

        if(data.success){

            $.messager.alert("提示","保存成功","info");

        }else{

            $.messager.alert("提示","保存失败,原因:"

+data.msg,"error");

        }

    });

});

});

</script>

</head>

<body>

    <form id="ff">

        <table align="center" cellpadding="0" cellspacing="0" style="width:80%">

            <tr>

                <td align="right" width="180">姓名: </td>

                <td><input type="text" name="name" class="easyui-validatebox"

data-options="required:true"/></td>

            </tr>
```

```
<tr>

    <td align="right" width="180">电子邮箱: </td>

    <td><input type="text" name="email"/></td>

</tr>

<tr>

    <td align="right" width="180">主题: </td>

    <td><input type="text" name="theme" value="默认"/></td>

</tr>

<tr>

    <td align="right" width="180">性别: </td>

    <td>

        <select name="gender">

            <option></option>

            <option value="1">男</option>

            <option value="0">女</option>

        </select>

    </td>

</tr>

</table>

</form>

<br>

<hr>

<input id="btn1" type="button" value="Load 加载数据并自动填充数据到表单">

<input id="btn2" type="button" value="Clear 清空表单数据">

<input id="btn3" type="button" value="reset 重置表单数据">

<input id="btn4" type="button" value="提交表单数据">

</body>

</html>
```



```

</style>

<script type="text/javascript">
  $(document).ready(function(){
  });
</script>

</head>

<body>
  <p>HTML 标签创建</p>
  <input type="text" class="easyui-numberbox" value="100"
data-options="min:0,precision:2"></input>

  <br>
  <hr>

</body>
</html>
```

5.4. combobox 组件

常用属性：

属性名	参数	备注
url	string	通过 URL 加载远程列表数据。
valueField	string	基础数据值名称绑定到该下拉列表框。
textField	string	基础数据字段名称绑定到该下拉列表框。

常用方法：

事件名	参数	备注
setValue	value	设置下拉列表框的值。

演示案例：

```

<!DOCTYPE html>

<html>

  <head>

    <title>combobox 下拉列表框</title>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <link rel="stylesheet" type="text/css"
href="../easyui/themes/default/easyui.css">

    <link rel="stylesheet" type="text/css" href="../easyui/themes/icon.css">

    <script type="text/javascript" src="../easyui/jquery.min.js"></script>

    <script type="text/javascript" src="../easyui/jquery.easyui.min.js"></script>

    <script type="text/javascript"
src="../easyui/locale/easyui-lang-zh_CN.js"></script>

    <style type="text/css">

      body{

        font-family: "Microsoft YaHei";

      }

    </style>

    <script type="text/javascript">

      $(document).ready(function(){

        $("#btn1").click(function(){

          //选中下拉框的值

          $("#cc").combobox("setValue", "Java");

        });
    
```



```
    });  
  
    </script>  
  
</head>  
  
<body>  
  
    <select id="cc" class="easyui-combobox" name="dept" style="width:200px;" >  
  
        <option>Java</option>  
  
        <option>PHP</option>  
  
        <option>IOS</option>  
  
        <option>UI</option>  
  
        <option>C</option>  
  
    </select>  
  
    <br>  
  
    <hr>  
  
    <input id="btn1" type="button" value="setValues 设置 Java 和 IOS 选中">  
  
    <input id="btn2" type="button" value="LoadData 重新加载选项数据">  
  
</body>  
</html>
```

5.5. datebox 和 datetimerbox

演示案例：

```
<!DOCTYPE html>  
  
<html>  
  
    <head>  
  
        <title>datebox、datetimerbox 日期时间输入框</title>  
  
        <meta http-equiv="content-type" content="text/html; charset=UTF-8">  
  
        <link rel="stylesheet" type="text/css"
```

```
href="../../easyui/themes/default/easyui.css">
    <link rel="stylesheet" type="text/css"
href="../../easyui/themes/icon.css">
    <script type="text/javascript"
src="../../easyui/jquery.min.js"></script>
    <script type="text/javascript"
src="../../easyui/jquery.easyui.min.js"></script>
    <script type="text/javascript"
src="../../easyui/locale/easyui-lang-zh_CN.js"></script>
    <style type="text/css">
    body{
        font-family: "Microsoft YaHei";
    }
    p {
        color: white;
        background-color: #0092E7;
    }
    </style>
    <script type="text/javascript">
    $(document).ready(function(){
    });
    </script>
</head>

<body>
```

```
<p>日期组件</p>

<input id="dd" type="text" class="easyui-datebox" required
="required"> </input>

<br>

<p>日期时间组件</p>

<input class="easyui-datetimebox" name="birthday"
    data-options="required:true,showSeconds:false" value="3/4/2010
2:3" style="width:150px">

<br>

<hr>

</body>

</html>
```

6. 案例：完成商品添加

6.1. 商品编辑表单验证

```
<tr>

    <td>商品名称</td>

    <td><input type="text" name="name" class="easyui-validatebox"
data-options="required:true"/></td>

</tr>

<tr>

    <td>商品原价</td>

    <td><input type="text" name="price" class="easyui-numberbox"
data-options="required:true"/></td>

</tr>

<tr>
```

```
<td>商品优惠价</td>

<td><input type="text" name="currentPrice"

class="easyui-numberbox" data-options="required:true"/></td>

</tr>
```

6.2. 保存商品

6.2.1. 编写 Dao 方法

```
@Override

public void save(Product product) {

    Session session = HibernateUtil.getSession();

    try {

        session.beginTransaction();

        session.saveOrUpdate(product);

        session.getTransaction().commit();

    } catch (Exception e) {

        e.printStackTrace();

        session.getTransaction().rollback();

        throw new RuntimeException(e);

    }finally{

        session.close();

    }

}
```

6.2.2. 编写 Service 方法

```
@Override

public void save(Product product) {

    productDao.save(product);

}
```

```
}
```

6.2.3. 编写 Action 方法

```
private Product product = new Product();

/**
 * 保存商品
 */
public String save(){
    Map<String,Object> result = new HashMap<String,Object>();

    try {
        productService.save(product);
        result.put("success", true);
    } catch (Exception e) {
        e.printStackTrace();
        result.put("success", false);
        result.put("msg", e.getMessage());
    }

    ActionContext.getContext().getValueStack().push(result);

    return SUCCESS;
}

@Override
public Product getModel() {
    return product;
}
```

6.2.4. product_manage.jsp 页面

```
//保存商品
```

```
$("#save").click(function(){  
    // 使用 form 的 submit 方法  
    $("#editForm").form("submit", {  
        url : "../product_save.action",  
        onSubmit : function() {  
            // 判断表单的验证是否通过  
            return $("#editForm").form("validate");  
        },  
        success : function(data) { // data:服务器返回的数据, data.success:  
成功标记  
            // data.msg 失败原因  
            // 转换为 json 格式  
            data = eval("(" + data + ")");  
            if (data.success) {  
                // 刷新 datagrid 的数据  
                $("#list").datagrid("reload");  
                // 关闭编辑窗口  
                $("#editWin").window("close");  
                // 成功  
                $.messager.show({  
                    title : "提示",  
                    msg : "保存成功"  
                })  
            } else {  
                // 失败  
                $.messager.alert("提示", "保存失败: " + data.msg, "error")  
            }  
        }  
    })  
}
```

```
});  
  
});
```

7. 案例：完成商品修改-信息回显

7.1. 编写 Dao

```
@Override  
  
public Product findById(Integer id) {  
    Session session = HibernateUtil.getSession();  
  
    try {  
        return session.get(Product.class, id);  
    } catch (Exception e) {  
        e.printStackTrace();  
        throw new RuntimeException(e);  
    } finally {  
        session.close();  
    }  
}
```

7.2. 编写 Service

```
@Override  
  
public Product findById(Integer id) {  
    return productDao.findById(id);  
}
```

7.3. 编写 Action

```
/**
```

```
* 修改前-回显商品  
*/  
  
public String get(){  
    Product prod = productService.findById(product.getId());  
    ActionContext.getContext().getValueStack().push(prod);  
    return SUCCESS;  
}
```

7.4. product_manage.jsp 页面

```
// 点击修改，弹出编辑窗口  
  
$("#editBtn").click(function() {  
    // 清空表单数据  
    $("#editForm").form("clear");  
  
    // 判断用户只能选择一行  
    var rows = $("#list").datagrid("getSelections");  
    if (rows.length != 1) {  
        $.messager.alert("提示", "修改操作只能选择一行", "warning");  
        return;  
    }  
    // 获取当前选择的行的 id  
    var id = rows[0].id;  
    // 请求后台的 url，获取数据，填充表单  
    // uuid:使用 uuid 名称是为了避免与 Action 的模型的 id 发生冲突  
    $("#editForm").form("load", "../product_get.action?id=" + id);  
  
    // 打开窗口  
    $("#editWin").window("open");  
});
```


8. 案例：完成商品修改-保存更新

```
<form id="editForm" action="" method="post">  
    <input type="hidden" name="id"/>  
    <table>
```

9. 案例：完成商品删除

9.1. 编写 Dao

```
@Override  
  
    public void delete(Integer id) {  
        Session session = HibernateUtil.getSession();  
  
        try {  
            session.beginTransaction();  
  
            Product prod = session.get(Product.class, id);  
  
            session.delete(prod);  
  
            session.getTransaction().commit();  
        } catch (Exception e) {  
            e.printStackTrace();  
  
            session.getTransaction().rollback();  
  
            throw new RuntimeException(e);  
        } finally {  
            session.close();  
        }  
    }  
}
```

9.2. 编写 Service

```
@Override
```

```
public void delete(String ids) {  
    if(ids!=null && !ids.equals("")){  
        String[] idArray = ids.split(",");  
        for (String id : idArray) {  
            productDao.delete(Integer.parseInt(id));  
        }  
    }  
}
```

9.3. 编写 Action

```
private String ids;  
  
public void setIds(String ids) {  
    this.ids = ids;  
}  
  
/**  
 * 批量删除商品  
 */  
  
public String delete(){  
    Map<String,Object> result = new HashMap<String,Object>();  
  
    try {  
        productService.delete(ids);  
        result.put("success", true);  
    } catch (Exception e) {  
        e.printStackTrace();  
        result.put("success", false);  
        result.put("msg", e.getMessage());  
    }  
  
    ActionContext.getContext().getValueStack().push(result);  
}
```

```
        return SUCCESS;
    }
}
```

9.4. product_manage.jsp 页面

```
// 删除

$("#deleteBtn").click(function() {

    // 1. 获取选择的行

    var rows = $("#list").datagrid("getSelections");

    // 删除操作至少选择一行

    if (rows.length < 1) {

        $.messager.alert("提示", "删除操作至少选择一行", "warning");

        return;
    }

    // 确认用户是否要删除

    $.messager.confirm("提示", "确认删除数据吗?", function(value) {

        if (value) {

            // 制定 id 参数格式: ids=1,2,10...

            var ids = new Array();

            // 2. 获取选择行的 id

            for (var i = 0; i < rows.length; i++) {

                // push(): 把元素放入到数组中

                ids.push(rows[i].id);

            }

            // array-string: join("字符串") :

            // 取出数组里面的每个元素, 使用指定的字符串进行拼接, 最后返回拼接
            // 后的字符串

            ids = ids.join(",");

            // alert(ids);
        }
    });
}
```

```
// 3.把所有的 id 发送给后台
/*
 * 参数一: url 地址 参数二: 传递的参数 ids=1 或 {id:1} 参数三 回
调函数 参数四: 服务器返回类型
 * html,json
 */
$.post("../product_delete.action", {
    "ids" : ids
}, function(data) { // data:服务器返回数据
    if (data.success) {
        // 刷新 datagrid 的数据
        $("#list").datagrid("reload");
        // 成功
        $.messager.show({
            title : "提示",
            msg : "删除成功"
        })
    } else {
        // 失败
        $.messager.alert("提示", "保存失败: " + data.msg,
"error")
    }
}, "json");
});
});
```