原始方式获取表单数据封装到实体类

```
        HttpServletRequest request = ServletActionContext.getRequest();
        String name = request.getParameter("username");
        String pswd = request.getParameter("password");
        String age = request.getParameter("age");

        User user = new User();
        user.setUsername(name);
        user.setPassword(pswd);
        user.setAge(Integer.valueOf(age));
        System.out.println(user);
```

属性封装（会用）

1、直接把表单提交的属性分装到 Action 的属性里面

2、实现步骤

(1)、在 action 成员变量位置定义变量

- 变量名称和表单输入项的 name 属性值一样

```
    <form action="${pageContext.request.contextPath }/formData" method="post">
        username:<input type="text" name="username"/>
        <br>
        password:<input type="text" name="password"/>
        <br>
        age:<input type="text" name="age"/>
        <br>
        <input type="submit" value="登录"/>
    </form>
```

(2)、生成变量的 set 方法（最好把 set 和 get 方法都写出来）

```
public class FromDataAction extends ActionSupport {
```

```
2
3       private String username;
4       private String password;
5       private String age;
6       public String getUsername() {
7           return username;
8       }
9       public void setUsername(String username) {
10          this.username = username;
11      }
12      public String getPassword() {
13          return password;
14      }
15      public void setPassword(String password) {
16          this.password = password;
17      }
18  }
```

模型驱动封装（重点）

1、使用模型驱动方式，可以直接把表单数据封装到实体类的对象里面（注：表单输入项的 name 属性值和实体类的属性名称要一致）

2、实现步骤

(1)、action 实现接口 ModelDriven

(2)、实现接口里面的方法 getModel 方法

- 把创建对象返回

(3)、在 action 里面创建实体类对象

3、使用模型驱动封装和属性封装注意的问题：

(1)、在一个 Action 中，获取表单数据可以属性封装，使用模型驱动封装，不能同时使用属性封装和模型驱动封装获取同一个表单数据

如果同时使用，只会执行模型驱动封装。

4、参考代码

```
1   package daiwei.learning.struts2;
```

```
2
3  import com.opensymphony.xwork2.Action;
4  import com.opensymphony.xwork2.ActionSupport;
5  import com.opensymphony.xwork2.ModelDriven;
6
7  import daiwei.learning.pojo.User;
8
9  public class FormData2Action extends ActionSupport implements
   ModelDriven<User> {
10
11     private User user = new User();
12
13     @Override
14     public User getModel() {
15
16         return user;
17     }
18
19     @Override
20     public String execute() throws Exception {
21
   System.out.println(user.getUsername()+"::"+user.getPassword()+"::"+user.getAge
   ());
22         return Action.NONE;
23     }
24
25
26 }
```

表达式封装（会用）

1、实现过程

(1)、使用表达式封装可以把表单数据封装到实体类对象里面

第一步：在 Action 里面声明实体类

第二步：生成实体类变量的 set 和 get 方法

第三步：在表单输入项的 name 属性值里面写表达式形式

2、表达式封装也可以归类到属性封装

3、参考代码

action

```
package daiwei.learning.struts2;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

import daiwei.learning.pojo.User;

public class FormData3Action extends ActionSupport {

    private User user = new User();

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    @Override
    public String execute() throws Exception {
        System.out.println(user);
        return Action.NONE;
    }
}
```

login.jsp

```
    <form action="${pageContext.request.contextPath }/formData2" method="post">
        username:<input type="text" name="user.username"/>
        <br>
        password:<input type="text" name="user.password"/>
        <br>
        age:<input type="text" name="user.age"/>
        <br>
        <input type="submit" value="登录"/>
    </form>
```

比较表达式封装和模型封装

1、相同点：使用表达式封装和模型封装都能把数据封装到实体类对象里面

2、不同点：

(1)、 使用模型驱动只能把数据封装到一个实体类对象里面

-在一个Action 里面不能使用模型驱动把数据封装到不同得实体类对象里面

(2)、使用表达式封装可以把数据封装到不同的实体类对象里面

封装数据到 List 集合

1、实现步骤：

第一步：在 Action 中声明一个 List

第二步：生成 Action 的 set 和 get 方法

第三步：在表单输入项里面写表达式

2、参考代码：

action:

```
package daiwei.learning.struts2;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

import daiwei.learning.pojo.User;

public class FormDataListAction extends ActionSupport {

    private List<User> list = new ArrayList<>();

    public List<User> getList() {
        return list;
    }
```

```
19
20      public void setList(List<User> list) {
21          this.list = list;
22      }
23
24      @Override
25      public String execute() throws Exception {
26          System.out.println(list);
27          return Action.NONE;
28      }
29  }
```

表单中：

```
1       <form action="${pageContext.request.contextPath }/formDataList"
    method="post">
2           username:<input type="text" name="list[0].username"/>
3           <br>
4           password:<input type="text" name="list[0].password"/>
5           <br>
6           age:<input type="text" name="list[0].age"/>
7           <br>
8           <br>
9           <br>
10          username:<input type="text" name="list[1].username"/>
11          <br>
12          password:<input type="text" name="list[1].password"/>
13          <br>
14          age:<input type="text" name="list[1].age"/>
15          <br>
16          <br/>
17          <br/>
18          username:<input type="text" name="list[2].username"/>
19          <br>
20          password:<input type="text" name="list[2].password"/>
21          <br>
22          age:<input type="text" name="list[2].age"/>
23          <br>
24          <input type="submit" value="登录"/>
25      </form>
```

封装数据到 Map 集合

1、实现步骤：

第一步：声明 map 集合

第二步：生成 set 和 get 方法

第三步：在表单输入项的 name 属性值里面写表达式

2、参考代码：

Action中的代码

```
package daiwei.learning.struts2;

import java.util.HashMap;
import java.util.Map;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

import daiwei.learning.pojo.User;

public class FormDataMapAction extends ActionSupport {

    private Map<String,User> map = new HashMap<>();


    public Map<String, User> getMap() {
        return map;
    }


    public void setMap(Map<String, User> map) {
        this.map = map;
    }


    @Override
    public String execute() throws Exception {
        for (String key : map.keySet()) {
            System.out.println(key+":"+map.get(key));
        }
        return Action.NONE;
    }
}
```

表单中的代码：

```
    <form action="${pageContext.request.contextPath }/formDataMap"
method="post">
        <!-- 设置 key值 map['key'] -->
        username:<input type="text" name="map['user1'].username"/>
        <br>
        password:<input type="text" name="map['user1'].password"/>
        <br>
        age:<input type="text" name="map['user1'].age"/>
        <br>
        <br>
        username:<input type="text" name="map['user2'].username"/>
        <br>
        password:<input type="text" name="map['user2'].password"/>
        <br>
        age:<input type="text" name="map['user2'].age"/>
        <br>
        <br>
        username:<input type="text" name="map['user3'].username"/>
        <br>
        password:<input type="text" name="map['user3'].password"/>
        <br>
        age:<input type="text" name="map['user3'].age"/>
        <br>
        <input type="submit" value="登录"/>
    </form>
```

以上的封装到 集合中 都是用的是表达式封装