StrutsCatalogDTOImplementation

Problem
Usually in J2EE based applications we need to shuttle business data between Web-tier and EJB-tier. Normally a Data Transfer Object (DTO) is what people resort to rather than wiring repeated costly fine-grained calls to an entity bean. While working with Struts, "Action-Form" which is one of the DTO implementations, acts as an intermediate repository used to capture client data as well as the business data coming from an EIS system. It is a bad practice to pass Action-Form itself between the tiers.

One of the options is to pass the values from "Form Bean" to the session bean methods in the form of method arguments. This does not have any technical flaw except that on change of number of arguments we have to change the method definition too, and possibly if there is some interface definition, even that would require a change. The suggested solution is to copy the data into a DTO and pass the same to EJB-tier. Ideally this DTO shall be passed to session bean (session façade), which in turn can talk to Entity beans.

Best Practice
According to the Layered pattern, classes in a layered/tiered system should either interact with other classes in the same layer or with classes of the adjacent layers. What that means is "Action-Form" class should either be used by View Layer or Controller Layer. Therefore another DTO should be used to transfer data from Controller Layer to Model Layer. This Transfer Object can be populated manually or a generic utility can be written to handle this transparently.

There are two options available for populating Data Transfer Object

- Create your own "Value or Transfer Object" and hand-code the copying of data from "Action-Form" to this Object and while doing that, take care of the data type conversions. Similarly on its way back you repeat the same exercise.
- **Use Commons' "BeanUtils" class. This uses reflection API to achieve its objectives.**

```
  }}}
It has been observed that developers frequently use the 1st option and there is no issue with that except that
this mundane exercise is repeated every time besides making the code very bulky and ugly. While copying,
suitable data type conversion has to be hand-coded in the DTO setter methods, so that when data is copied from
the Form-Bean to the DTO, it gets converted to appropriate business type. For example, a Form bean can capture
all the client layer attributes as a String but they need to be converted to various Java or native type before
performing any business operation on them.

{{{
```

You can save yourself from hand-coding all this by using copyProperties() method of BeanUtils class This copies the data from one bean to the another provided the variable names of business attributes in "Value or Transfer Object" is same as the one in Form bean. This also relieves you from data-type conversion by transparently converting source-bean attribute type to destination-bean attribute type.

--Puneet Agarwal

Comments

One can use XDoclet to generate "value objects" (DTOs) from his entity bean's classes. But one must be careful, because this approach hurts every time an entity bean definition changes... ;o)

--Carsten Habicht