ForwardingWithDifferentParameter

```
Acknowledgement:

This short tutorial is an extract from long discussion about forwarding to an action with different parameters.
Here are a list of people that contributed in the discussion:

 * Nimmons, Buster (originator)
 * Geeta Ramani
 * Arvind Rajpurohit
 * Robert Taylor
 * None None
```

Here is a snippet of the original question from Nimmons:

" *I have an Action which is usualy accessed from an html form page... It requires 4 request parameters. I need to forward to this action from another action but the action I'm forwarding from does not have all 4 of the required request parameters. I know what the value of the parameters should be but I cannot put them in the request scope unless I use request.setAttribute() however the action being forwarded to is not looking for the required parameters from the Attributes but the requestParametes.. has anyone figured out an easy way to modify the available requestParameters before forwarding* "

I encounter this problem when I was coding my soon to be personal homegrown weblog. This is my problem:

I want to build a forum discussion for my weblog. In the forum I allow people to post replies on certain threads. So I created a simple form for them to key in their post reply's title (text field) and post's message (text area). After they finish filling in all the necessary fields, I want to redirect them back to the *topic page* that they reply ?

Topic page is a page where the user can view the full content of a topic message in the forum.

The way i show my topic content page is by including a parameter in my request ....do?dipatch=OpenTopic&**topicID=123** The topic id is use to identify which forum topic that they are opeing so that I can load the appropriate topic message from the database.

When user submit their post reply I direct them to my Action:

```
public class PostAction extends DispatchAction
{

    public ActionForward savePost(
                ActionMapping mapping,
                ActionForm form,
                HttpServletRequest request,
                HttpServletResponse response){

            //Saving the post
            return findMapping ( ??? ); //ooops how do I forward to the topic page, because the topic page need
a                           parameter?

    }



}
```

After reading the discussion in Struts mailing list I realize that the solution is so simple. Geeta is the one who shed some light to this ActionForward thing.

So here what I do :

```
public class PostAction extends DispatchAction
{

    public ActionForward savePost(
                ActionMapping mapping,
                ActionForm form,
                HttpServletRequest request,
                HttpServletResponse response){

            //Saving the post
            ActionForward af = new ActionForward();
            //Following line is only necessary when using modules
            af.setModule("moduleName");
            af.setPath("/viewTopic.do?dispatch=viewTopic&topicID=123");

            return af;
    }

}
```

So now I can direct my user to view their post reply in the topic page.

I have encountered a similar problem. The above solution works with some limitations

1. It somewhat breaks the mapping abstraction, and creates additional burden to encode the URL. 2. It does not seem to work if the action to forward to (in the above case, viewTopic) expects the a parameter that is also specified in the original request.

The other solutions I tried include:

1. Use a redirect ActionForward (it solves issue 2). 2. Overwrite the request attribute - ActionForm mapping logic so that the logic first looks at request context attribute before looking at request parameter. With this approach, one can then set additional attributes using request.setAtrribute (...);

However, it seems that solution 2 is not a typical soltuion. Any feedback?

How about a Wrapper class like below that provides Parameter copying functionality? - Mark Diggory

```
package edu.harvard.hmdc.curate.study;

import java.util.Enumeration;
import java.util.Hashtable;
import java.net.URLEncoder;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForward;

public class MyActionForward extends ActionForward {

    private Hashtable parameters = new Hashtable();

    public MyActionForward(ActionForward forward) {
        super(forward.getName(),forward.getPath(),forward.getRedirect(),forward.getContextRelative());
    }

    public MyActionForward(ActionForward forward, HttpServletRequest request) {
        super(forward.getName(),forward.getPath(),forward.getRedirect(),forward.getContextRelative());
        parameters.putAll(request.getParameterMap());
    }

    public void addParameter(String name, String value) {
                String[] newValues = null;
                String[] oldValues = (String[]) parameters.get(name);
                if (oldValues == null) {
                        newValues = new String[1];
                        newValues[0] = value;
                } else {
                        newValues = new String[oldValues.length + 1];
                        System.arraycopy(oldValues, 0, newValues, 0, oldValues.length);
                        newValues[oldValues.length] = value;
                }
                parameters.put(name, newValues);
        }

    public String getPath() {
        String result = super.getPath();

        if(!parameters.isEmpty())
            result += "?";

        for(Enumeration enum = parameters.keys();enum.hasMoreElements();){
            String next = (String)enum.nextElement();
            String[] vals = (String[])parameters.get(next);

            for(int i = 0; i < vals.length;i++){
                if(vals[i] != null){
                    result += next;
                    result += "=";
                    result += URLEncoder.encode(vals[i]);

                    if(i <= vals.length-2)
                        result += "&";
                }
            }

            if(enum.hasMoreElements())
                result += "&";
        }

        return result;
    }

}
```

Note: This practice constitutes a form of ActionChaining which has been the subject of much debate in Struts, and which some consider a bad design practice.