

CommonsChainAndSpringFramework

Below are the steps of a possible solution for using commons-chain with Struts, having the Catalog configured via Spring.

If you only need a single Catalog, you can write one which assumes that the name of any command is the name of a Spring bean, and looks up the Command using the Spring [ApplicationContext](#).

If you need multiple catalogs, you would want an implementation of the [CatalogFactory](#) class which can wire together beans from Spring. This should be pretty simple, but no code is provided here yet.

Below is a possible implementation of the single-Catalog solution.

```
public class SpringCatalog implements Catalog, ApplicationContextAware {

    private ApplicationContext appContext;

    public SpringCatalog() {
        System.out.println("Instantiating Spring Catalog..");
    }

    public void addCommand(String name, Command command) {
        // does nothing here..
    }

    public Command getCommand(String name) {
        return (Command)appContext.getBean(name);
    }

    public Iterator getNames() {
        Map commands = new HashMap();
        try {
            commands = appContext.getBeansOfType(
                org.apache.commons.chain.Command.class,
                true,
                true);
        } catch(Exception e) {
            System.err.println("Error in retrieving command names..");
        }
        return commands.keySet().iterator();
    }

    public void setApplicationContext(ApplicationContext ctx)
        throws ApplicationContextException, BeansException {
        appContext = ctx;
    }
}
```

- Declare your Catalog class and commands in your spring configuration file (usually it is applicationContext.xml).

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

    <!-- ===== GENERAL DEFINITIONS ===== -->

    <!-- Configurer that replaces ${...} placeholders with values from properties files -->
    <!-- (in this case, mail and JDBC related properties) -->

    <bean id="persistenceManager" class="com.myapp.strutschain.bean.PersistenceManager"/>
    <bean name="catalog" class="com.myapp.strutschain.core.SpringCatalog"/>
    <bean name="login" class="org.apache.commons.chain.impl.ChainBase">
        <constructor-arg>
            <list>
                <ref bean="firstCmd"/>
                <ref bean="secondCmd"/>
            </list>
        </constructor-arg>
    </bean>
    <bean name="firstCmd"
          class="com.myapp.strutschain.command.PreLoginCommand"/>

    <bean name="secondCmd"
          class="com.myapp.strutschain.command.LoginCommand">
        <property name="persistenceManager">
            <ref local="persistenceManager"/>
        </property>
    </bean>
</beans>

```

- Declare the Spring plugin in your struts configuration file

```

<plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
    <set-property property="contextConfigLocation"
                  value="/WEB-INF/applicationContext.xml"/>
</plug-in>

```

- Write an Action class servers as base to retrieve commands

```

public class CommandAction extends ActionSupport {

    protected static String SUCCESS = "success";

    protected Command getCommand(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {
        Catalog catalog = (Catalog) getWebApplicationContext().getBean("catalog");

        // Marco uses the mapping's name, but I would suggest that using the path or the parameter are more
        // appropriate,
        // or in Struts 1.3, you could use the mapping's command property directly.
        // The "name" property is used to identify a form bean associated with the action mapping, and may not
        be
        // appropriately unique for your business logic. -- JG
        String name = mapping.getName();
        System.out.println("Retrieving command for action"+ name);
        Command command = catalog.getCommand(name);
        return command;
    }

    protected Context getContext(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {
        ServletContext application = request.getSession()
            .getServletContext();
        Context context = new ServletWebContext(
            application, request, response);

        return context;
    }

    protected ActionForward findLocation(ActionMapping mapping,
                                         boolean stop) {
        if (stop) return mapping.getInputForward(); // Something failed
        return mapping.findForward(SUCCESS);
    }

    public ActionForward execute(
        ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
        Command command = getCommand(mapping, form, request, response);
        Context context = getContext(mapping, form, request, response);
        boolean stop = command.execute(context);
        ActionForward location = findLocation(mapping, stop);
        return location;
    }
}

```