http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte2:ptl:annotationcommandmapargumentresolver

AnnotationCommandmapArgumentResolver

Outline

When you need to get inputs from the screen (JSP) in Controller, the Object Map is preferred over the conventional Object Command(Form Class). The egovFramework Ver. 3.0 now helps you use the Object Map via @CommandMap and AnnotationCommandmapArgumentResolver, contrary to the older versions where CommandMapArgumentResolver came into play.

Being the implementation class for org.springframework.web.method.support.HandlerMethodArgumentResolver, AnnotationCommandMapArgumentResolver allows the Object Map contain the parameter values of the Object HTTP Request for use in Controller.

Description

HandlerMethodArgumentResolver

Though the up-to-date egovFramework offers a variety of types (check out the linkfor more information) that the argument of method @Controller for Spring MVC , you'll need to use the custom argument from time to time.
In this connection, Spring MVC now offers the Interface HandlerMethodArgumentResolver for your to customize the type of argument for any Controller.

Contrary to the older version of Spring implementing WebArgumentResolver prior to registration to AnnotationMethodHandlerAdapter, Spring Ver. 3.1 or newer requests a user to implement HandlerMethodArgumentResolver, register thereof to RequestMappingHandlerAdapter and the eventual application of ArgumentResolver.

When you need to use the custom argument entitled "MySpecialArg" for a method of Controller:

```
@Controller
public class HelloController{
	public String hello(MySpecialArg mySpecialArg,...){
		...
		return "...";
	}
}
```

You'll need to generate a class implementing the Interface HandlerMethodArgumentResolver as follows:

```
boolean supportsParameter(MethodParameter parameter);

Object resolveArgument(MethodParameter parameter,ModelAndViewContainer mavContainer, NativeWebRequest webRequest,WebDataBinderFactory binderFactory) throws Exception;
public class MySpecialArgumentResolver implements HandlerMethodArgumentResolver{

		@Override
		public boolean supportsParameter(MethodParameter parameter) {
			if(MySpecialArg.class.isAssignableFrom(parameter.getParameterType())) {
				return true;
			}
			else {
				return false;
			}
		}
```

```
	@Override
	public Object resolveArgument(MethodParameter parameter,
		ModelAndViewContainer mavContainer, NativeWebRequest webRequest,
		WebDataBinderFactory binderFactory) throws Exception {
		return new MySpecialArg();
	}
}
```

You'll then need to register the generated ArgumentResolver to the attribute customArgumentResolvers of RequestMappingHandlerAdapter. You may register multiple ArgumentResolver classes as these properties fall under "list type property".

```
	<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
		<property name="customArgumentResolvers">
			<list>
				<bean class="... MySpecialArgumentResolver" />
			</list>
		</property>
	</bean>
```

AnnotationCommandMapArgumentResolver

You may either have the parameter values in the Object HTTP Request in a bean of specific format, or take to having the Object Map contain the parameter values.
In egovFramework Ver. 3.0 or older, you'll need to work @CommandMap and AnnotationCommandMapArgumentResolver if you want to use the Object Map in Controller.

```
public String helloPost(@CommandMap Map commandMap, ModelMap model) {
...
}
```

For you to have commandMap contain the parameter values, you'll need to work AnnotationCommandMapArgumentResolver, as described above. As far as the Object Map contains @CommandMap in any argument of Contoller, you may have the Object Map contain the parameter values in the Object HTTP Request.

```
package egovframework.rte.ptl.mvc.bind;
...
public class AnnotationCommandMapArgumentResolver implements HandlerMethodArgumentResolver{

	@Override
	public boolean supportsParameter(MethodParameter parameter) {
		if(Map.class.isAssignableFrom(parameter.getParameterType())
			&& parameter.hasParameterAnnotation(CommandMap.class)) {
			return true;
		}
		else {
			return false;
		}
	}

	@Override
	public Object resolveArgument(MethodParameter parameter,
		ModelAndViewContainer mavContainer, NativeWebRequest webRequest,
		WebDataBinderFactory binderFactory) throws Exception {

		Map<String, Object> commandMap = new HashMap<String, Object>();
		HttpServletRequest request = (HttpServletRequest) webRequest.getNativeRequest();
		Enumeration<?> enumeration = request.getParameterNames();
```

```
		while(enumeration.hasMoreElements()){
			String key = (String) enumeration.nextElement();
			String[] values = request.getParameterValues(key);
			if(values!=null){
				commandMap.put(key, (values.length > 1) ? values:values[0] );
			}
		}
		return commandMap;
	}
}
```

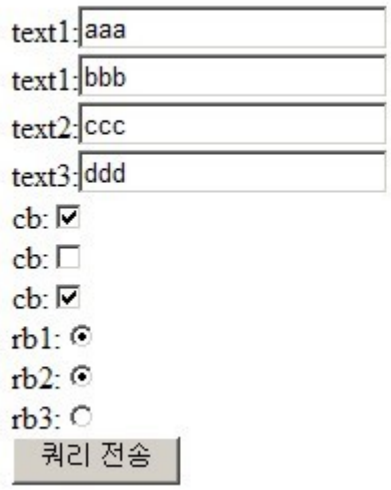
You'll need to register EgovRequestMappingHandlerAdapter before you intend to use AnnotationCommandMapArgumentResolver. Contrary to the older version where RequestMappingHandlerAdapter was all you need to register, you'll need to use EgovRequestMappingHandlerAdapter before having AnnotationCommandMapArgumentResolver registered for use of the Object Map in Controller.

Special attention is need if you would want to use RequestMappingHAndlerAdapter or <mvc:annotation-driven> in which case you're not allowed to use AnnotationCommandMapArgumentResolver.

```
<bean class="egovframework.rte.ptl.mvc.bind.annotation.EgovRequestMappingHandlerAdapter">
	<property name="customArgumentResolvers">
		<list>
			<bean
class="egovframework.rte.ptl.mvc.bind.AnnotationCommandMapArgumentResolver" />
		</list>
	</property>
</bean>

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>
```

When you have the test form and submit:



You'll have the data of the objective @CommandMap Map in the form of key or value. Some values are contained in a single parameter in the form of array.

```
key:text1	value:{aaa,bbb}
key:text2	value:ccc
key:text3	value:ddd
key:cb	value:{on,on}
key:rb2	value:on
key:rb1	value:on
```

References

- new handler method controller processing