Published on *Javalobby* (http://java.dzone.com)

Unified EL Learns Method Invocations

By *beckchr*
Created *2009/08/10 - 12:55am*

The Unified Expression Language has been specified in JSR245 as part of JSP 2.1. The final release came in 2006. Now, the EL gets a silent update for JEE6 which adds an eagerly-awaited feature: method invocations. This means that expressions like ${foo.bar(baz)} will be valid EL expressions. This should significantly improve the flexibility in JSP programming and lower the need to embed native Java in JSPs to zero.

However, the EL can also easily been used outside JSP, and that's what we're going to do in this article.

How do method invocations work?

The new API adds a new method to the abstract javax.el.ELResolver class:

```
public Object invoke(ELContext context, Object base, Object method, Class<?>[] paramTypes, Object[] params)
 return null;
}
```

The base argument references the object on which a method whose name is given by the method argument should be invoked. The paramTypes argument will be always null. Finally, the params argument holds the evaluated values to be used as method parameters. As we can see, the default implementation does nothing.

The javax.el.BeanELResolver overrides this method and implements reflection-base method invocations by performing the following steps:

1. Try to choose a public non-vararg method with the given name and a number of arguments matching params.length
2. If no method has been found in (1), try to choose a public vararg method with the given name and a number of arguments at most params.length + 1
3. Coerce params to the formal parameters of the method chosen in (1) or (2).
4. Invoke the method, call context.setPropertyResolved(true) and return the result

Everything is done by a standard resolver and even vararg methods can be used..

Using the EL in your application

The new API is implemented by the latest releases of *JUEL*, an Apache licensed implementation of the EL. Download JUEL 2.2.x [1] and add juel-2.2.x.jar to you classpath.

The following sample code illustrates the use of method invocations:

```
// the ExpressionFactory implementation is de.odysseus.el.ExpressionFactoryImpl|
System.setProperty("javax.el.methodInvocations", "true");
ExpressionFactory factory = new de.odysseus.el.ExpressionFactoryImpl(System.getProperties());

// package de.odysseus.el.util provides a ready-to-use subclass of ELContext
de.odysseus.el.util.SimpleContext context = new de.odysseus.el.util.SimpleContext();

// set value for top-level property "foo" to String value "bar"
factory.createValueExpression(context, "${foo}", String.class).setValue(context, "bar");

// create an expression
ValueExpression e = factory.createValueExpression(context, "${foo.toUpperCase()}", String.class);
// evaluate...
System.out.println(e.getValue(context)); // --> BAR
```

Source URL: http://java.dzone.com/articles/unified-el-learns-method

Links:
[1] http://sourceforge.net/projects/juel/files/