JSR 245 JavaServer™ Pages JSP 2.1 Maintenance Review 2 Part II

Please send feedback to kin-man.chung@sun.com

This is a maintenance review for JSR-245, JavaServer™ Pages 2.1 Specification (JSP 2.1). There are two parts in the MR, Part I covers the JSP specification, and part II covers the Expression Language specification.

The main goal for this part of the maintenance review is is to introduce a new syntax for method invocations in EL, and to add an access method to `ValueExpression` as requested by JSR-303.

Propsed Changes

1. EL Operators [] and .

These operators now accepts optional parameters:

```
expr-a[expr-b](parameters)
expr-a.identifier-b(parameters)
```

where `parameters` is 0 or more expressions, separated by commas. This syntax can be used for method invocations.

The expression `expr-a` is evaluated to represent a bean object. The expression `expr-b` is evaluated and coerced to a string. This string or the string `identifier-b` is the name of the method in `expr-a`. The `parameters` are the actual parameters for the method invocation.

If the expression is a `ValueEpxression`, then calling its `getValue` method causes the method to be invoked. If the expression is a `MethodExpression`, then calling its `invoke` method causes the method to be invoked, and the parameters `params` for the `invoke` method will be ignored, since those specified in EL will be used.

To be more precise, the behavior for invoking the method specified in the above EL expression is resolved by the EL resolver in the current EL context. A method `invoke` is added to `ELResolver` to resolve the value for such EL syntax. The `invoke` method in `BeanELResolver` provides a default behavior for invoking methods in a bean object. By using a custom EL resolver, the user can make calls to "methods" that do not exist in a class. This allows for creation of macros, and can lead to interesting applications.

2. New method javax.el.ELResolver.invoke

Attemps to resolve and invoke the given `method` on the given `base` object.

If this resolver handles the given (base, method) pair, the `propertyResolved` property of the `ELContext` object must be set to `true` by the resolver, before returning. If this property is not `true` after this method is called, the caller should ignore the return value.

A default implementation is provided that returns null so that existing classes that extend ELResolver can continue to function.

```
@param context The context of this evaluation.
@param base The bean on which to invoke the method
@param method The simple name of the method to invoke.
    Will be coerced to a String.
```

```
    @param paramTypes An array of Class objects identifying the
        method's formal parameter types, in declared order.
        Use an empty array if the method has no parameters.
        Can be null, in which case the method's formal
        parameter types are assumed to be unknown.
    @param params The parameters to pass to the method, or
        null if no parameters.
    @return The result of the method invocation (null if
        the method has a void return type).
    @throws MethodNotFoundException if no suitable method can be found.

    @throws ELException if an exception was thrown while performing
        (base, method) resolution.  The thrown exception must be
        included as the cause property of this exception, if
        available.  If the exception thrown is an
        InvocationTargetException, extract its
        cause and pass it to the
        ELException constructor.

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

3. New method java.el.BeanELResolver.invoke

Provides a default implementation for resolving the method invocations in a bean.

If the base object is not `null`, invoke the method, with the given parameters on this bean. The return value from the method is returned.

If the base is not `null`, the `propertyResolved` property of the `ELContext` object must be set to `true` by this resolver, before returning. If this property is not `true` after this method is called, the caller should ignore the return value.

The provided method object will first be coerced to a `String`. The methods in the bean is then examined and an attempt will be made to select one for invocation. If no suitable can be found, a `MethodNotFoundException` is thrown.

If the given paramTypes is not `null`, select the method with the given name and parameter types.

Else select the method with the given name that has the same number of parameters. If there are more than one such method, the method selection process is undefined.

Else select the method with the given name that takes a variable number of arguments.

Note the resolution for overloaded methods will likely be clarified in a future version of the spec.

The provided parameters are coerced to the corresponding parameter types of the method, and the method is then invoked.

```
    @param context The context of this evaluation.
    @param base The bean on which to invoke the method
    @param method The simple name of the method to invoke.
        Will be coerced to a String.  If method is
        "<init>"or "<clinit>" a MethodNotFoundException is
        thrown.
    @param paramTypes An array of Class objects identifying the
        method's formal parameter types, in declared order.
```

```
          Use an empty array if the method has no parameters.
          Can be null, in which case the method's formal
          parameter types are assumed to be unknown.
      @param params The parameters to pass to the method, or
          null if no parameters.
      @return The result of the method invocation (null if
          the method has a void return type).
      @throws MethodNotFoundException if no suitable method can be found.
      @throws ELException if an exception was thrown while performing
          (base, method) resolution.  The thrown exception must be
          included as the cause property of this exception, if
          available.  If the exception thrown is an
          InvocationTargetException, extract its
          cause and pass it to the
          ELException constructor.

    public Object invoke(ELContext context,
                         Object base,
                         Object method,
                         Class[] paramTypes,
                         Object[] params) {
        ...
    }
```

4. New class javax.el.ValueReference

This encapsulates a base model object and one of its properties.

```
public class ValueReference implements Serializable {

    private Object base;
    private Object property;

    public ValueReference(Object base, Object property) {
        this.base = base;
        this.property = property;
    }

    public Object getBase() {
        return base;
    }

    public Object getProperty() {
        return property;
    }
}
```

5. New method javax.el.ValueExpression.getValueReference

Returns a `ValueReference` for this expression instance.

```
      @param context the context of this evaluation
      @return the ValueReference for this
      ValueExpression, or null if this
      ValueExpression is not a reference to
      a base (null or non-null) and a property.
      If the base is null, and the property is a EL variable, return
      the ValueReference for the
      ValueExpression associated with this EL variable.

    public ValueReference getValueReference(ELContext context) {
        return null;
    }
```