Enum Parameter Recipe

Enum Component Parameter

It's not uncommon to create a component that has a bit of complex behavior that you want to be able to easily control, and an enumerated type (a Java enum) seems like the right approach.

Our example comes from Tapestry's Select component, which has a blankOption parameter that has an enum type.

Let's start with the enum type itself:

BlankOption.java

```
public enum BlankOption
{
    /** Always include the blank option, even if the underlying property is required. */
    ALWAYS,

    /** Never include the blank option, even if the underlying property is optional. */
    NEVER,

    /** The default: include the blank option if the underlying property is optional. */
    AUTO;
}
```

Next, we define the parameter:

Select.java (partial)

```
    /**
     * Controls whether an additional blank option is provided. The blank option precedes all other options and
is never
     * selected. The value for the blank option is always the empty string, the label may be the blank string;
the
     * label is from the blankLabel parameter (and is often also the empty string).
     */
    @Parameter(value = "auto", defaultPrefix = BindingConstants.LITERAL)
    private BlankOption blankOption;
```

Note the use of literal as the default prefix; this allows us to use the name of the option in our template, e.g. `<t:select blankoption="never" .../>`. Without the default prefix setting, "never" would be interpreted as a property expression (and you'd see an error when you loaded the page).

The final piece of the puzzle is to inform Tapestry how to convert from a string, such as "never", to a BlankOption value.

TapestryModule.java (partial)

```
    public static void contributeTypeCoercer(Configuration<CoercionTuple> configuration)
    {
       . . .

       add(configuration, BlankOption.class);

       . . .
    }

    private static <T extends Enum> void add(Configuration<CoercionTuple> configuration, Class<T> enumType)
    {
        configuration.add(CoercionTuple.create(String.class, enumType, StringToEnumCoercion.create(enumType)));
    }
```

Related Articles

- Supporting Informal Parameters
- Default Parameter
- Enum Parameter Recipe
- Component Parameters

The TypeCoercer service is ultimately responsible for converting the string to a BlankOption, but we have to tell it how, by contributing an appropriate CoercionTuple. The CoercionTuple identifies the source and target types (String and BlankOption), and an object to perform the coercion (an instance of StringToEnumCoercion, via the `create()` static method).