Tapestry5SpringIntegrationAlternative2

Integrating Spring & T5.0.5

Please note that this method was inspired from the solution presented by [SergeEby] [TapestrySpringIntegrationAlternative1] in t5.0.2. – [JunTsai]

See also:

- TapestrySpringIntegration

add spring module class

```
package corner.orm.tapestry.spring;

import org.apache.tapestry.ioc.ObjectLocator;
import org.apache.tapestry.ioc.ObjectProvider;
import org.apache.tapestry.ioc.OrderedConfiguration;

/**
 *  spring 1.x/2.x + Tapestry5.0.5
 * @author jun.tsai
 * @version $Revision: 2830 $
 * @since 3.0
 */
public class SpringModule {

        //contribution master object provider
        public static void contributeMasterObjectProvider(
                        OrderedConfiguration<ObjectProvider> configuration,
                        ObjectLocator locator) {
                configuration.add("spring", locator
                                .autobuild(SpringObjectProvider.class));
        }
}
```

add spring annotation class

```
@Target( { PARAMETER, FIELD })
@Retention(RUNTIME)
@Documented
public @interface Spring {
        public String value();
}
```

Create the SpringObjectProvider class

The class listing is presented below:

```
package corner.orm.tapestry.spring;

import org.apache.tapestry.ioc.AnnotationProvider;
import org.apache.tapestry.ioc.ObjectLocator;
import org.apache.tapestry.ioc.ObjectProvider;
import org.apache.tapestry.ioc.annotations.InjectService;
import org.apache.tapestry.ioc.services.TypeCoercer;



/**
 * @author jcai
 * @version $Revision: 2828 $
 * @since 3.0
 */
public class SpringObjectProvider implements ObjectProvider {
        private TypeCoercer _typeCoercer;

        public SpringObjectProvider(@InjectService("TypeCoercer")
        TypeCoercer typeCoercer) {
                _typeCoercer = typeCoercer;
        }


        /**
         * @see org.apache.tapestry.ioc.ObjectProvider#provide(java.lang.Class,
         *      org.apache.tapestry.ioc.AnnotationProvider,
         *      org.apache.tapestry.ioc.ObjectLocator)
         */
        public <T> T provide(Class<T> objectType,
                        AnnotationProvider annotationProvider, ObjectLocator locator) {
                Spring annotation = annotationProvider.getAnnotation(Spring.class);

                if (annotation == null)
                        return null;

                String value = annotation.value();
                Object obj = getSpringContext().getBean(value);

                T coerced = _typeCoercer.coerce(obj, objectType);

                return coerced;

        }
        private ApplicationContext getSpringContext(){
              // get application context
        }

}
```

add module define in your MANIFEST.MF file

```
Tapestry-Module-Classes: corner.orm.tapestry.spring.SpringModule
```

Inject Spring beans

We are done! it's now time to inject some Spring beans. For illustration purpose, I am updating an example from Tapestry 5 integration test source code:

First, here is sample applicationContext-hibernate.xml config file. Only the relevant parts are presented (I am using Spring/Hibernate stack)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema
/beans/spring-beans-2.0.xsd">

    <!-- Hibernate SessionFactory -->
        <bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.
AnnotationSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="configLocation" value="${hibernate.config}"/>
        <property name="hibernateProperties">
            <value>
                hibernate.dialect=${hibernate.dialect}
                hibernate.show_sql=${hibernate.show_sql}
                hibernate.query.substitutions=true 'Y', false 'N'
            </value>
        </property>
    </bean>

    ...

    <!-- Transaction manager for a single Hibernate SessionFactory (alternativeto JTA) -->
    <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory"/>
    </bean>

    <bean id="toDoDatabase" class="org.apache.tapestry.integration.app1.services.ToDoDatabaseHibernateImpl">
        <property name="sessionFactory" ref="sessionFactory"/>
    </bean>

</beans>
```

As you can see, the toDoDatabase is now mapped to the ToDoDatabaseHibernateImpl class.

And then, the ToDoList class:

```
        package org.apache.tapestry.integration.app1.pages;

        import java.util.List;

        import org.apache.tapestry.PrimaryKeyEncoder;
        import org.apache.tapestry.annotations.Component;
        import org.apache.tapestry.annotations.Inject;
        import org.apache.tapestry.corelib.components.Form;
        import org.apache.tapestry.integration.app1.data.ToDoItem;
        import org.apache.tapestry.integration.app1.services.ToDoDatabase;
        import org.apache.tapestry.util.DefaultPrimaryKeyEncoder;

        public class ToDoList
        {
            @Inject
            @Spring("toDoDatabase")
            private ToDoDatabase _database;

            private ToDoItem _item;

            private DefaultPrimaryKeyEncoder<Long, ToDoItem> _encoder;

            @Component
            private Form _form;

            public List<ToDoItem> getItems()
            {
                return _encoder.getValues();
            }
```

```
    public ToDoItem getItem()
    {
        return _item;
    }

    public void setItem(ToDoItem item)
    {
        _item = item;
    }

    public ToDoDatabase getDatabase()
    {
        return _database;
    }

    public PrimaryKeyEncoder getEncoder()
    {
        return _encoder;
    }

    void onPrepare()
    {
        List<ToDoItem> items = _database.findAll();

        _encoder = new DefaultPrimaryKeyEncoder<Long, ToDoItem>();

        for (ToDoItem item : items)
        {
            _encoder.add(item.getId(), item);
        }
    }

    void onSuccess()
    {
        int order = 0;

        for (ToDoItem item : _encoder.getValues())
        {
            item.setOrder(order++);
            _database.update(item);
        }
    }

    void onSelectedFromAddNew()
    {
        if (_form.isValid())
        {
            ToDoItem item = new ToDoItem();
            item.setTitle("<New To Do>");
            item.setOrder(_encoder.getValues().size());

            _database.add(item);
        }
    }

    void onActionFromReset()
    {
        _database.reset();
    }
}
```

Note that the only change to this class the the @Inject annotation

Congrats! you are now using injected Spring beans in T5.0.5