Integrating with Spring Framework

Tapestry easily integrates with Spring Framework, allowing beans defined by Spring to be injected into Tapestry IoC services, and into Tapestry components. In addition, with Tapestry 5.2 and later, you can also go the other way, injecting Tapestry services in Spring beans.

For integrating Spring Security into your application, see Security.

Related Articles

- Integrating with Spring Framework
- Security

Contents

Spring Version

This module is compiled and tested against Spring Framework 2.5.6. It should be reasonable to override the dependency to earlier versions of Spring, though the code makes use of some APIs that were added to Spring to support JDK 1.5 annotations.

Usage

The integration is designed to be a very thin layer on top of Spring's normal configuration for a web application.

Detailed instructions are available in the Spring documentation. Please omit the part about creating a ContextLoaderListener: this is now done automatically by Tapestry.

Required dependency

To integrate Spring with Tapestry, you should add the below dependency in your classpath. The following exemple is for Maven users.

```
  <dependency>
    <groupId>org.apache.tapestry</groupId>
    <artifactId>tapestry-spring</artifactId>
    <version>[your-tapestry-version]</version>
  </dependency>
```

Update your web.xml file

The short form is that you must make two small changes to your application's web.xml.

First, a special filter is used in replace of the standard TapestryFilter:

```
  <filter>
    <filter-name>app</filter-name>
    <!-- Special filter that adds in a T5 IoC module derived from the Spring WebApplicationContext. -->
    <filter-class>org.apache.tapestry5.spring.TapestrySpringFilter</filter-class>
  </filter>
```

Secondly, you may add the normal Spring configuration, consisting of an optional <context-param> identifying which Spring bean configuration file(s) to load:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/daoContext.xml /WEB-INF/applicationContext.xml</param-value>
</context-param>
```

The <context-param> lists the Spring bean configuration file. It is optional and defaults to just /WEB-INF/applicationContext.xml if omitted.

Accessing the Spring Application Context

By integrating Spring in Tapestry, you get full access on Spring ApplicationContext as if you were accessing to any Tapestry service. Simply @Inject into your pages and components.

```
@Inject
private ApplicationContext springContext;
```

Injecting beans

Inside your component classes, you may use the @Inject annotation. Typically, just adding @Inject to the field type is sufficient to identify the Spring bean to inject:

```
@Inject
private UserDAO userDAO;
```

Searching for Spring beans is threaded into the MasterObjectProvider service. The Spring context becomes one more place that Tapestry searches when determining the injection for a injected field or method parameter.

Injecting Tapestry services in Spring beans

Added in 5.2

If you have configured Spring to allow annotation-based injection, then you will be able to inject Tapestry services into your Spring Beans.

This feature is only available when Spring ApplicationContext is not configured and loaded externally. Inside your Spring beans, you may use @[Inject|http://tapestry.apache.org/current/apidocs/org/apache/tapestry5/ioc/annotations/Inject.html] and @[Autowired|http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html] annotations.

```
Simply add these two annotations on top the field you want to inject in your Spring bean.

{code:language=java}

    @Inject
    @Autowired
    private MyService myService;

{code}

or use @Inject on top of arguments in @Autowired bean constructor methods

{code:language=java}

    private final MyService myService;
    @Autowired
    public UserDAOImpl(@Inject MyService myService)
    {
      this.myService = myService;
    }
{code}

h3. Configuring Spring with Tapestry Symbols

This is accomplished by a BeanFactoryPostProcessors that resolves the values of 'placeholders' from symbol
values. In the following example the value of the Bean's property 'productionMode' is the value of the
Tapestry's  symbol [tapestry.production-mode|http://tapestry.apache.org/current/apidocs/org/apache/tapestry5
/SymbolConstants.html#PRODUCTION_MODE]

{code:language=xml}
  <bean id="myBean" class="org.example.MyBean">
    <property name="productionMode" value="${tapestry.production-mode}"/>
  </bean>
{code}
```

ApplicationContextCustomizer

A chain-of-command service, ApplicationContextCustomizer allows the application context, created by Tapestry, to be customized as it is created. You may contribute your own ApplicationContextCustomizer instances as needed.

5.0 Compatibility Mode

In some circumstances, it is desirable to configure the Spring ApplicationContext externally. The context <config-param> "tapestry.use-external-spring-context" can be configured to "true". Tapestry will then use an existing ApplicationContext, provided by a Spring ContextLoaderListener. You will still be able to inject Spring beans into Tapestry components and services, and the ApplicationContext service will be visible ... but you will not be able to inject Tapestry IoC services into Spring beans.

This option provides compatibility with the tapestry-spring 5.0, including exposing Spring beans as Tapestry IoC services (something that no longer occurs unless compatibility mode is enabled).

Changes From 5.0

The changes below represent an unfortunate backwards compatibility issue. If necessary, you can still use tapestry-spring version 5.0.18 with the rest of Tapestry.

- You may now use the @Inject or @InjectService annotations inside Spring beans; these will be resolved to Tapestry services or other objects available via the MasterObjectProvider. Please see the detailed guide to Injection.
- The dependency on Spring is no longer scope "provider" and has changed to 2.5.6.
- Spring Beans are no longer exposed as services, unless 5.0 compatibility mode is enabled.
- You no longer create a ContextLoaderListener.

Limitations

Non-singleton beans are not handled properly. Tapestry will request the beans from the application context in a manner unsuitable for their life cycle. For the moment, you should consider the non-singleton beans to be not injectable. Instead, inject the ApplicationContext service and obtain the non-singleton beans as needed.