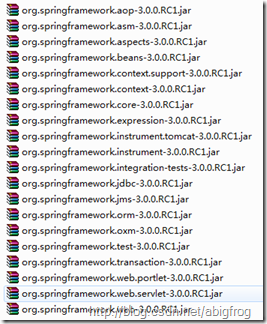
**[Spring3.0新特性](http://matt-u.javaeye.com/blog/508742)**

一、首先，模块的组织更加的细致，从那么多的jar分包就看的出来：   
  
   
  
Spring的构建系统以及依赖管理使用的是Apache Ivy，从源码包看出，也使用了Maven。   
  
Maven确实是个好东西，好处不再多言，以后希望能进一步用好它。   
  
二、新特性如下：   
  
Spring Expression Language （Spring表达式语言）   
  
IoC enhancements/Java based bean metadata （Ioc增强/基于Java的bean元数据）   
  
General-purpose type conversion system and UI field formatting system （通用类型转换系统和UI字段格式化系统）   
  
Object to XML mapping functionality (OXM) moved from Spring Web Services project （对象到XML映射功能从Spring Web Services项目移出）   
  
Comprehensive REST support （广泛的REST支持）   
  
@MVC additions （@MVC增强）   
  
Declarative model validation （声明式模型验证）   
  
Early support for Java EE 6 （提前对Java EE6提供支持）   
  
Embedded database support （嵌入式数据库的支持）   
  
三、针对Java 5的核心API升级   
  
1、BeanFactory接口尽可能返回明确的bean实例，例如：   
  
T getBean(String name, Class requiredType)   
  
Map getBeansOfType(Class type)   
  
Spring3对泛型的支持，又进了一步。个人建议泛型应该多用，有百利而无一害！   
  
2、Spring的TaskExecutor接口现在继承自java.util.concurrent.Executor：   
  
扩展的子接口AsyncTaskExecutor支持标准的具有返回结果Futures的Callables。   
  
任务计划，个人还是更喜欢Quartz。   
  
3、新的基于Java5的API和SPI转换器   
  
无状态的ConversionService 和 Converters   
  
取代标准的JDK PropertyEditors   
  
类型化的ApplicationListener，这是一个实现“观察者设计模式”使用的事件监听器。   
  
基于事件的编程模式，好处多多，在项目中应该考虑使用，基于事件、状态迁移的设计思路，有助于理清软件流程，和减少项目的耦合度。   
  
四、Spring表达式语言   
  
Spring表达式语言是一种从语法上和统一表达式语言（Unified EL）相类似的语言，但提供更多的重要功能。它可以在基于XML配置文件和基于注解的bean配置中使用，并作为基础为跨Spring portfolio平台使用表达式语言提供支持。   
  
接下来，是一个表达式语言如何用于配置一个数据库安装中的属性的示例：   
  
<bean class="mycompany.RewardsTestDatabase">   
    <property name="databaseName"   
        value="#{systemProperties.databaseName}"/>   
    <property name="keyGenerator"   
        value="#{strategyBean.databaseKeyGenerator}"/>   
</bean>   
如果你更愿意使用注解来配置你的组件，那么这种功能同样可用：   
  
@Repository public class RewardsTestDatabase {   
      @Value("#{systemProperties.databaseName}")   
      public void setDatabaseName(String dbName) { … }   
       
      @Value("#{strategyBean.databaseKeyGenerator}")   
      public voidsetKeyGenerator(KeyGenerator kg) { … }   
}   
  
又多一种表达式语言，造轮子的运动还在继续中！   
  
五、基于Java的bean元数据   
  
JavaConfig项目中的一些核心特性已经集成到了Spring中来，这意味着如下这些特性现在已经可用了：   
  
@Configuration   
  
@Bean   
  
@DependsOn   
  
@Primary   
  
@Lazy   
  
@Import   
  
@Value   
  
又来一堆的注解，无语了，感觉还是配置文件方便！：（   
  
这儿有一个例子，关于一个Java类如何使用新的JavaConfig特性提供基础的配置信息：   
  
package org.example.config;   
  
@Configuration   
public class AppConfig {   
    private @Value("#{jdbcProperties.url}") String jdbcUrl;   
    private @Value("#{jdbcProperties.username}") String username;   
    private @Value("#{jdbcProperties.password}") String password;   
  
    @Bean   
    public FooService fooService() {   
        return new FooServiceImpl(fooRepository());   
    }   
  
    @Bean   
    public FooRepository fooRepository() {   
        return new HibernateFooRepository(sessionFactory());   
    }   
  
    @Bean   
    public SessionFactory sessionFactory() {   
        // wire up a session factory   
        AnnotationSessionFactoryBean asFactoryBean =   
            new AnnotationSessionFactoryBean();   
        asFactoryBean.setDataSource(dataSource());   
        // additional config   
        return asFactoryBean.getObject();   
    }   
  
    @Bean   
    public DataSource dataSource() {   
        return new DriverManagerDataSource(jdbcUrl, username, password);   
    }   
}   
  
为了让这段代码开始生效，我们需要添加如下组件扫描入口到最小化的应用程序上下文配置文件中：   
  
<context:component-scan base-package="org.example.config"/>   
<util:properties id="jdbcProperties" location="classpath:org/example/config/jdbc.properties"/>   
  
六、在组件中定义bean的元数据   
  
感觉Spring提供了越来越多的注解、元数据，复杂性已经超出了当初带来的方便本身！   
  
七、通用类型转换系统和UI字段格式化系统   
  
Spring3加入了一个通用的类型转换系统，目前它被SpEL用作类型转换，并且可能被一个Spring容器使用，用于当绑定bean的属性值的时候进行类型转换。   
  
另外，还增加了一个UI字段格式化系统，它提供了更简单的使用并且更强大的功能以替代UI环境下的JavaBean的PropertyEditors，例如在SpringMVC中。   
  
这个特性要好好研究下，通用类型转换系统如果果如所言的话，带来的好处还是很多的。   
  
八、数据层   
  
对象到XML的映射功能已经从Spring Web Services项目移到了Spring框架核心中。它位于org.springframework.oxm包中。   
  
OXM?研究下！时间真不够！   
  
九、Web层   
  
在Web层最激动人心的新特性莫过于新增的对构件REST风格的web服务和web应用的支持！另外，还新增加了一些任何web应用都可以使用的新的注解。   
  
服务端对于REST风格的支持，是通过扩展既有的注解驱动的MVC web框架实现的。   
  
客户端的支持则是RestTemplate类提供的。   
  
无论服务端还是客户端REST功能，都是使用HttpConverter来简化对HTTP请求和应答过程中的对象到表现层的转换过程。   
  
MarshallingHttpMessageConverter使用了上面提到的“对象到XML的映射机制”。   
  
十、@MVC增强   
  
新增了诸如@CookieValue 和 @RequestHeaders这样的注解等。   
  
十一、声明式模型验证   
  
支持JSR 303，使用Hibernate Validator作为实现。   
  
十二、提前对Java EE6提供支持   
  
提供了使用@Async注解对于异步方法调用的支持（或者EJB 3.1里的 @Asynchronous）   
  
另外，新增对JSR 303, JSF 2.0, JPA 2.0等的支持。   
  
十三、嵌入式数据库的支持   
  
对于嵌入式的Java数据库引擎提供了广泛而方便的支持，诸如HSQL, H2, 以及Derby等。   
  
这是不是代表一种潮流呢？数据库向越来越小型化发展，甚至小型化到嵌入式了，我认为这在桌面级应用上还是很有市场的。