BusinessTool

null - 1.6

version :1.0

jQuery mobile framework takes the write less, do more mantra to the next level: Instead of writing unique apps for each mobile device or OS, the jQuery mobile framework allows you to design a single highly-branded web site or application that will work on all popular smartphone, tablet, and desktop platforms

COREMODULE

Hibernate-jpa-api
Hibernate definition of the Java Persistence 2.0 (JSR 317) API

EXTERNALMODULE

junit
JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development JUnit is linked as a JAR at compile-time; the framework resides under packages junit.framework for JUnit 3.8 and earlier and under org.junit for JUnit 4 and later.A JUnit Test fixture is a Java object. With older versions of JUnit, fixtures had to inherit from junit.framework.TestCase, but new tests using JUnit 4 should not do this. Test methods must be annotated by the @Test annotation. If the situation requires, it is also possible to define a method to execute before (or after) each (or all) of the test methods with the @Before (or @After) and @BeforeClass (or @AfterClass) annotations.

Jersey-json
Jersey JSON support comes as a set of JAX-RS MessageBodyReader<T> and MessageBodyWriter<T> providers distributed with jersey-json module. These providers enable using three basic approaches when working with JSON format:
1)POJO support
2)JAXB based JSON support
3)Low-level, JSONObject/JSONArray based JSON support
The first method is pretty generic and allows you to map any Java Object to JSON and vice versa. The other two approaches limit you in Java types your resource methods could produce and/or consume. JAXB based approach could be taken if you want to utilize certain JAXB features. The last, low-level, approach gives you the best fine-grained control over the outcoming JSON data format.

jsr311-api
Jersey is the open source, production quality, JAX-RS (JSR 311) Reference Implementation for building RESTful Web services. But, it is also more than the Reference Implementation. Jersey provides an API so that developers may extend Jersey to suit their needs. The governance policy is the same as the one used in the GlassFish project. We also use the same two licenses - CDDL 1.1 and GPL 2 with CPE - so, you can pick which one suites your needs better.

json
JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

log4j
With log4j it is possible to enable logging at runtime without modifying the application binary

Slf4j-api
The Simple Logging Facade for Java or (SLF4J) serves as a simple facade or abstraction for various logging frameworks, such as java.util.logging, log4j and logback. SLF4J allows the end-user to plug in the desired logging framework at deployment time. Note that SLF4J-enabling your library/application implies the addition of only a single mandatory dependency, namely slf4j-api.jar.

Slf4j-log4j12
The Simple Logging Facade for Java or (SLF4J) serves as a simple facade or abstraction for various logging frameworks

Jcl Over Slf4j
Aspectjrt
aspectjweaver
The AspectJ weaver introduces advices to java classes

Servlet-api
The Servlet API contained in the Java package hierarchy javax.servlet, defines the expected interactions of the web container and a servlet

Flexjson
Ojdbc
Org springframework roo annotations
Spring-core
A core library for spring framework.

Spring Framework: Core Properties :-
1)Supplying constructor arguments
2) Using factory methods
3) Dependency injection
4) Supplying other beans as properties or constructor args
5) Bean scopes

Objects whose implementations are likely to change are defined in XML file. Java code does not need to refer to any specific implementation
1)You use the <bean> tag to define object's name and class
2)You use nested <property> or <constructor-arg> elements to give startup values to the object

Spring is useful when you have objects whose implementations change often. These objects are defined in bean definition file, isolating Java code from changes in the implementation You supply initialization values via constructors or setters. Spring is even more useful when the initialization values are other beans. That is, your Spring-managed objects depend on other bean values.

Supplying these values in bean definition file is called “dependency injection”

1) Because Java code doesn’t have to depend explicitly on specific concrete values
2) Instead, bean values are passed in (“injected”) at run time
3) Also called “Inversion of Control” (IoC)

Spring-test

This allows performing requests and generating responses without the need for running in a Servlet container

Spring-aop

Spring -aop provides support to implement custom aspects, complementing their use of OOP with AOP. While OO decomposes applications into a hierarchy of objects, AOP decomposes programs into aspects or concerns. This enables modularization of concerns such as transaction management that would otherwise cut across multiple objects. (Such concerns are often termed crosscutting concerns.) One of the key components of Spring is the AOP framework. While the Spring IoC containers (BeanFactory and ApplicationContext) do not depend on AOP, meaning you don't need to use AOP if you don't want to, AOP complements Spring IoC to provide a very capable middleware solution.

AOP is used in Spring:
1)To provide declarative enterprise services, especially as a replacement for EJB declarative services. The most important such service is declarative transaction management, which builds on Spring's transaction abstraction.
2)To allow users to implement custom aspects, complementing their use of OOP with AOP.

Thus you can view Spring AOP as either an enabling technology that allows Spring to provide declarative transaction management without EJB; or use the full power of the Spring AOP framework to implement custom aspects. If you are interested only in generic declarative services or other pre-packaged declarative middleware services such as pooling, you don't need to work directly with Spring AOP, and can skip most of this chapter.

Spring-context

Spring- context is a library that manages application context in spring framework intergration with Springs AOP features, message resource handling (for use in internationalization), event propagation, declarative mechanisms to create the ApplicationContext and optional parent contexts, and application-layer specific contexts such as the WebApplicationContext, among other enhancements.
Two of the most fundamental and important packages in Spring are the org.springframework.beans and org.springframework.context packages. Code in these packages provides the basis for Spring's Inversion of Control (alternately called Dependency Injection) features. The BeanFactory provides an advanced

configuration mechanism capable of managing beans (objects) of any nature, using potentially any kind of storage facility. The ApplicationContext builds on top of the BeanFactory (it's a subclass) and adds other functionality such as easier integration with Springs AOP features, message resource handling (for use in internationalization), event propagation, declarative mechanisms to create the ApplicationContext and optional parent contexts, and application-layer specific contexts such as the WebApplicationContext, among other enhancements.
In short, the BeanFactory provides the configuration framework and basic functionality, while the ApplicationContext adds enhanced capabilities to it, some of them perhaps more J2EE and enterprise-centric. In general, an ApplicationContext is a complete superset of a BeanFactory, and any description of BeanFactory capabilities and behavior should be considered to apply to ApplicationContexts as well.
Users are sometimes unsure whether a BeanFactory or an ApplicationContext are best suited for use in a particular situation. Normally when building most applications in a J2EE-environment, the best option is to use the ApplicationContext, since it offers all the features of the BeanFactory and adds on to it in terms of features, while also allowing a more declarative approach to use of some functionality, which is generally desirable. The main usage scenario when you might prefer to use the BeanFactory is when memory usage is the greatest concern (such as in an applet where every last kilobyte counts), and you don't need all the features of the ApplicationContext.

Spring Aspects
Spring-tx
Spring Transaction API

Hibernate-core
Hibernate-core artifact is needed to build applications using the native Hibernate APIs including defining metadata in both annotations as well as Hibernate's own hbm.xml format.

Hibernate-entitymanager
The EntityManager API is used to access a database in a particular unit of work. It is used to create and remove persistent entity instances, to find entities by their primary key identity, and to query over all entities. This interface is similar to the Session in Hibernate

Hibernate-validator
Hibernate Validator 4.x is the reference implementation for JSR 303 - Bean Validation of which Red Hat is the specification lead.JSR 303 - Bean Validation defines a metadata model and API for JavaBean validation. The default metadata source is annotations, with the ability to override and extend the meta-data through the use of XML validation descriptors. The API is not tied to a specific application tier or programming model. It is specifically not tied to either the web tier or the persistence tier, and is available for both server-side application programming, as well as rich client Swing application developer.

Validation-api
Bean Validation API

Cglib Nodep
jta
The Java Transaction API (JTA) is one of the Java Enterprise Edition (Java EE) APIs allowing distributed transactions to be done across multiple XA resources in a Java environment. JTA is a specification developed under the Java Community Process as JSR 907.

The JTA API consists of classes in two Java packages:
1)javax.transaction
2)javax.transaction.xa

The JTA is modelled on the X/Open XA architecture, but it defines two different APIs for demarcating transaction boundaries. It distinguishes between an application server such as an EJB server and an application component. It provides an interface, javax.transaction.TransactionManager, that is used by the application server itself to begin, commit and rollback the transactions. It provides a different interface, the javax.transaction.UserTransaction, that is used by general client code such as a servlet or an EJB to manage the transactions.

Java Transaction API:

The Java Transaction API consists of three elements: a high-level application transaction demarcation interface, a high-level transaction manager interface intended for an application server, and a standard Java mapping of the X/Open XA protocol intended for a transactional resource manager. The javax.transaction.UserTransaction interface provides the application the ability to control transaction boundaries programmatically. This interface may be used by Java client programs or EJB beans. The UserTransaction.begin method starts a global transaction and associates the transaction with the calling thread. The transaction-to-thread association is managed transparently by the Transaction Manager.
Support for nested transactions is not required. The UserTransaction.begin method throws the NotSupportedException when the calling thread is already associated with a transaction and the transaction manager implementation does not support nested transactions. Transaction context propagation between application programs is provided by the underlying transaction manager implementations on the client and server machines. The transaction context format used for propagation is protocol dependent and must be negotiated between the client and server hosts. For example, if the transaction manager is an implementation of the JTS specification, it will use the transaction context propagation format as specified in the CORBA OTS 1.1 specification. Transaction propagation is transparent to application programs. UserTransaction support in EJB server

Spring-jdbc
The Spring's DAO(Data access object) make it easy for us to use data access technologies like JDBC, Hibernate, JPA or JDO for accessing data from database. Using DAO you can switch any of the above persistence technology without any overhead. The Spring JDBC provides consistent access to the various database access technologies including JDBC, Hibernate, JPA or JDO.

Spring-orm
Spring Object/Relational Mapping

commons_pool
The Pool package makes it possible separate the way in which instances are pooled from the way in which instances are created and destroyed

Commons-dbcp
Commons Database Connection Pooling.

Openid4java
Spring Security Openid
Spring-web
Spring Web Flow is a Spring MVC extension that allows implementing the "flows" of a web application. A flow encapsulates a sequence of steps that guide a user through the execution of some business task. It spans multiple HTTP requests, has state, deals with transactional data, is reusable, and may be dynamic and long-running in nature. The sweet spot for Spring Web Flow are stateful web applications with controlled navigation such as checking in for a flight, applying for a loan, shopping cart checkout, or even adding a confirmation step to a form.

These scenarios have in common is one or more of the following traits:
1)There is a clear start and an end point.
2)The user must go through a set of screens in a specific order.
3)The changes are not finalized until the last step.
4)Once complete it shouldn't be possible to repeat a transaction accidentally.

Spring Web Flow provides a declarative flow definition language for authoring flows on a higher level of abstraction. It allows it to be integrated into a wide range of applications without any changes (to the flow programming model) including Spring MVC, JSF, and even Portlet web applications.

The following are common issues observed in stateful web applications with navigation requirements:
1)Visualizing the flow is very difficult.
2)The application has a lot of code accessing the HTTP session.
3)Enforcing controlled navigation is important but not possible.
4)Proper browser back button support seems unattainable.
5)Browser and server get out of sync with "Back" button use.

6)Multiple browser tabs causes concurrency issues with HTTP session data.
7)Spring Web Flow provides a solution to the above issues.

Spring-webmvc
The Spring Web MVC Framework is a robust, flexible, and well-designed framework for rapidly developing web applications using the MVC design pattern. The benefits achieved from using this Spring module are similar to those you get from the rest of the Spring Framework.
Bind directly to business objects—Spring MVC does not require your business (model) classes to extend any special classes; this enables to reuse business objects by binding them directly to the HTML forms fields.

Spring Js Resources
Commons Digester
commons_fileupload
The Commons FileUpload package makes it easy to add robust, high-performance, file upload capability to your servlets and web applications

Jstl Api
Jstl Impl
El Api
Joda Time
Jsp Api
Commons-codec
The codec package contains simple encoder and decoders for various formats such as Base64 and Hexadecimal. In addition to these widely used encoders and decoders, the codec package also maintains a Apache Commons Codec (TM) software provides implementations of common encoders and decoders such as Base64, Hex, Phonetic and URLs.

Tiles Core
Tiles Jsp
Spring-security-config
Spring Security provides comprehensive security services for J2EE-based enterprise software applications.

Spring-security-core
Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications

Spring Security Ldap
Spring Security Taglibs
Spring Security Web
Commons-lang
Commons.Lang, a package of Java utility classes for the classes that are in

java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang. The standard Java libraries fail to provide enough methods for manipulation of its core classes. Apache Commons Lang provides these extra methods. Lang provides a host of helper utilities for the java.lang API, notably String manipulation methods, basic numerical methods, object reflection, concurrency, creation and serialization and System properties. Additionally it contains basic enhancements to java.util.Date and a series of utilities dedicated to help with building methods, such as hashCode, toString and equals.
Note that Lang 3.0 (and subsequent versions) use a different package (org.apache.commons.lang3) than the previous versions (org.apache.commons.lang), allowing it to be used at the same time as an earlier version.

commons-collections
Types that extend and augment the Java Collections Framework.

Mockito All
Mockito Core
Spring Batch Core
Quartz
Ehcache-spring-annotations
Provides a simple model for integrating Ehcache in a Spring project via annotations

Spring-context-support
Spring -context is a library that manages application context in spring framework intergration with Springs AOP features, message resource handling (for use in internationalization), event propagation, declarative mechanisms to create the ApplicationContext and optional parent contexts, and application-layer specific contexts such as the WebApplicationContext, among other enhancements.

Solr Solrj
GoogleGSON
Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of

JsLibraries

jQuery-AMD
jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML.
jQuery is free, open source software, dual-licensed under the MIT License[Massachusetts Institute of Technology] or the GNU General Public

License, jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, CSS manipulation and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery library allows the creation of powerful and dynamic web pages and web applications.

jQuery contains the following features:
1)DOM element selections using the cross-browser open source selector engine Sizzle, a spin-off out of the jQuery project.
2)DOM traversal and modification (including support for CSS 1-3)
3)Events
4)CSS manipulation
5)Effects and animations
6)Ajax
7)Extensibility through plug-ins
8)Utilities - such as user agent information, feature detection
9)Compatibility methods that are natively available in modern browsers but need fallbacks for older ones - For example the inArray() and each() functions.
10)Cross-browser support

jQuery-ui-AMD

jQuery UI provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript Library, that you can use to build highly interactive web applications.jQuery UI is free, open source software, dual-licensed under the MIT License[Massachusetts Institute of Technology] and the GNU General Public License.

Interactions
1)Draggable - Make elements draggable
2)Droppable - Control where dragged elements may be dropped
3)Resizable - Make elements resizeable [5]:
4)Selectable - Advanced selection features for lists of elements
5)Sortable - Make a list of elements easily sortable

Widgets
All of jQuery UI's widgets are fully themeable using a consolidated, coordinated theme mechanism demonstrated by their ThemeRoller.
1)Accordion - Accordion containers
2)Autocomplete - Auto-complete boxes based on what the user types
3)Button - Enhanced button appearance, turn radio buttons and checkboxes into pushbuttons
4)Datepicker - Advanced date-picker
5)Dialog - Show dialog boxes on top of other content, easily and robustly
6)Progressbar - Progress bars, both animated and not

7)Slider - Fully customizable sliders with various features [6]:
8)Tabs - Tabbed user interface handling, with both inline and demand-loaded content

Effects
1)Color Animation - Animate the transition from one color to another
2)Toggle Class, Add Class, Remove Class, Switch Class - Animate the transition from one set of styles to another
3)Effect - A variety of effects (appear, slide-down, explode, fade-in, etc.)
4)Toggle - Toggle an effect on and off
5)Hide, Show - Using the effects above

Utilities
Position - Set an element's position relative to another element's position (alignment)

xml2json-AMD
xml2json converts xml to json and viceverza, using node-expat.

jsonpath-AMD
JSONPath provides XPath way of accessing data from JSON. JSONPath expressions always refer to a JSON structure in the same way as XPath expression are used in combination with an XML document. Since a JSON structure is usually anonymous and doesn't necessarily have a root member object JSONPath assumes the abstract name $ assigned to the outer level object. JSONPath expressions can use the dot–notation.