ResponsiveTomcat1Node

java-tomcat - 1.6

version :1.0.0-SNAPSHOT

Java Web services are client and server applications that communicate over the World Wide Web?s (WWW) HyperText Transfer Protocol (HTTP).

COREMODULE

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

EXTERNALMODULE

jdom
JDOM is a way to represent an XML document for easy and efficient reading, manipulation, and writing.

Commons-logging
Commons logging package is an ultra-thin bridge between different logging implementation portable with any logging implementation at runtime. Commons Logging is a thin adapter allowing configurable bridging to other, well known logging systems. When writing a library it is very useful to log information. However there are many logging implementations out there, and a library cannot impose the use of a particular one on the overall application that the library is a part of Applications (rather than libraries) may also choose to use commons-logging. While logging-implementation independence is not as important for applications as it is for libraries, using commons-logging does allow the application to change to a different logging implementation without recompiling code. Note that commons-logging does not attempt to initialise or terminate the underlying logging implementation that is used at runtime; that is the responsibility of the application. However many popular logging implementations do automatically initialise themselves; in this case an application may be able to avoid containing any code that is specific to the logging implementation used.

Json-lib
JSON-lib is a java library for transforming beans, maps, collections, java arrays and XML to JSON and back again to beans and DynaBeans. Includes support for Groovy and JRuby as well. jsonlib has two functions of interest, read and write. It also defines some exception: Read error, Write error, and unknown serializer error.For compatibility with the standard library, read is aliased to loads and write is aliased to dumps. They do not have the same set of advanced parameters, but may be used interchangeably for simple invocations.

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.

Jersey-server

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.

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.

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.

gson

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. Gson support java generics and doesn't require any special annotations.

Commons-io

Commons IO library contains utility classes, stream implementations, file filters, file comparators and endian classes.Commons IO is a library of utilities to assist with developing IO functionality.

There are six main areas included:
1)Utility classes - with static methods to perform common tasks
2)Input - useful Input Stream and Reader implementations
3)Output - useful Output Stream and Writer implementations

4)Filters - various implementations of file filters
5)Comparators - various implementations of java.util.Comparator for files
6)File Monitor - a component for monitoring file system events

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.

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.

activation

JavaBeans Activation Framework (JAF) is a standard extension to the Java platform that lets you take advantage of standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and instantiate the appropriate bean to perform the operation(s). It also enables you to dynamically register types of arbitrary data and actions associated with particular kinds of data. Additionally, it enables a program to dynamically provide or retrieve JavaBeans that implement actions associated with some kind of data.

antlr

ANTLR allows generating parsers, lexers, tree parsers, and combined lexer-parsers. Parsers can automatically generate abstract syntax trees which can be further processed with tree parsers. ANTLR provides a single consistent notation for specifying lexers, parsers, and tree parsers. This is in contrast with other parser/lexer generators and adds greatly to the tool's ease of use. By default, ANTLR reads a grammar and generates a recognizer for the language defined by the grammar (i.e. a program that reads an input stream and generates an error if the input stream does not conform to the syntax specified by the grammar). If

there are no syntax errors, then the default action is to simply exit without printing any message. In order to do something useful with the language, actions can be attached to grammar elements in the grammar. These actions are written in the programming language in which the recognizer is being generated. When the recognizer is being generated, the actions are embedded in the source code of the recognizer at the appropriate points. Actions can be used to build and check symbol tables and to emit instructions in a target language, in the case of a compiler. As well as lexers and parsers, ANTLR can be used to generate tree parsers. These are recognizers that process abstract syntax trees which can be automatically generated by parsers. These tree parsers are unique to ANTLR and greatly simplify the processing of abstract syntax trees.

aopalliance

The AOP framework for Java. The AOP Alliance project is a joint open-source project between several software engineering people who are interested in AOP and Java. Aspect-Oriented Programming (AOP) offers a better solution to many problems than do existing technologies such as EJB. AOP Alliance intends to facilitate and standardize the use of AOP to enhance existing middleware environments (such as J2EE), or development environements (e.g. JBuilder, Eclipse). The AOP Alliance also aims to ensure interoperability between Java/J2EE AOP implementations to build a larger AOP community.

asm

ASM is an all purpose Java bytecode manipulation and analysis framework. It can be used to modify existing classes or dynamically generate classes, directly in binary form. Provided common transformations and analysis algorithms allow to easily assemble custom complex transformations and code analysis tools. ASM offer similar functionality as other bytecode frameworks, but it is focused on simplicity of use and performance. Because it was designed and implemented to be as small and as fast as possible, it makes it very attractive for using in dynamic systems Moreover ASM can be used in a static way too.

Asm-attrs

Asm-attrs provides an implementation for various optional class, field, method and bytecode attributes.

1)Signature used for Java generics (JSR-14)
2)SourceDebugExtension used for non Java source debugging (JSR-45)
3)StackMap (used for two phase bytecode verifier from CLDC 1.1 and J2SE 1.5 (JSR-30, JSR-139, JSR-202)
4)Several attibutes used for annotations (JSR-175)

By default ASM strips optional attributes, in order to keep them in the bytecode that is being readed should pass an array of required attribute instances to ClassReader.accept() method. In order to add custom attributes to the manually constructed bytecode concrete subclasses of the Attribute can be passed to the correspond visit methods of the ClassVisitor and CodeVisitor.

cglib
cglib is a powerful, high performance and quality Code Generation Library. It is used to extend JAVA classes and implements interfaces at runtime.cglib is a set of utility classes that can be used to generate and load Java classes at runtime.

Commons-beanutis
Commons-beanUtils provides an easy-to-use but flexible wrapper around reflection and introspection. BeanUtils is a small library that provides an easy-to-use but flexible wrapper around reflection and introspection. Most Java developers are used to creating Java classes that conform to the JavaBeans naming patterns for property getters and setters. However, there are some occasions where dynamic access to Java object properties (without compiled-in knowledge of the property getter and setter methods to be called) is needed. Most Java developers are used to creating Java classes that conform to the JavaBeans naming patterns for property getters and setters. It is natural to then access these methods directly, using calls to the corresponding getXxx and setXxx methods. However, there are some occasions where dynamic access to Java object properties (without compiled-in knowledge of the property getter and setter methods to be called) is needed.

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.

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-logging
Commons logging package is an ultra-thin bridge between different logging implementation portable with any logging implementation at runtime.

Commons Logging is a thin adapter allowing configurable bridging to other, well known logging systems. When writing a library it is very useful to log information. However there are many logging implementations out there, and a library cannot impose the use of a particular one on the overall application that the library is a part of Applications (rather than libraries) may also choose to use commons-logging. While logging-implementation independence is not as important for applications as it is for libraries, using commons-logging does allow the application to change to a different logging implementation without recompiling code. Note that commons-logging does not attempt to initialise or terminate the underlying logging implementation that is used at runtime; that is the responsibility of the application. However many popular logging implementations do automatically initialise themselves; in this case an application may be able to avoid containing any code that is specific to the logging implementation used.

dom4j

dom4j is an open source Java library for working with XML, XPath and XSLT. It is compatible with DOM, SAX and JAXP standards.

ehcache

ehcache is a pure Java, in-process cache with the following features:

1) Fast.
2) Simple.
3) Multiple eviction policies: LRU, LFU and FIFO.
4) Caches can be in memory or on disk.
5) Disk Stores can be persistent between VM restarts.
6) Distributed caching using multicast and RMI, with a pluggable API.
7) Cache and CacheManager listeners
8) Supports multiple Caches per CacheManager, and multiple CacheManagers per application.
9) Acts as a pluggable cache for Hibernate 3.1, 3 and 2.1.
10) Small foot print. Both in terms of size and memory requirements.
11) Minimal dependencies apart from J2SE.
12) Fully documented. See the online Documentation and the online JavaDoc.
13) Comprehensive Test Coverage. See the clover test report.
14) Available under the Apache 1.1 license. EHCache's copyright and licensing has been reviewed and approved by the Apache Software Foundation, making EHCache suitable for use in Apache projects.
15) Production tested. EHCache is used on a large and very busy eCommerce site.
16) Web caching, pull-through caches and other common caching implementations are provided in the ehcache-constructs module.

Ehcache is a widely used open source Java distributed cache for general purpose caching, Java EE and light-weight containers. It features memory and disk stores, replicate by copy and invalidate, listeners, cache loaders, cache extensions, cache

exception handlers, a gzip caching servlet filter, RESTful and SOAP APIs. Ehcache is available under an Apache open source license and is actively supported.

gson
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. Gson support java generics and doesn't require any special annotations.

Hamcrest-core
Hamcrest is a framework for writing matcher objects allowing 'match' rules to be defined declaratively. There are a number of situations where matchers are invaluble, such as UI validation, or data filtering, but it is in the area of writing flexible tests that matchers are most commonly used. When writing tests it is sometimes difficult to get the balance right between overspecifying the test (and making it brittle to changes), and not specifying enough (making the test less valuable since it continues to pass even when the thing being tested is broken). Having a tool that allows you to pick out precisely the aspect under test and describe the values it should have, to a controlled level of precision, helps greatly in writing tests that are "just right". Such tests fail when the behaviour of the aspect under test deviates from the expected behaviour, yet continue to pass when minor, unrelated changes to the behaviour are made.

hibernate
Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions. Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and attempts to relieve the developer from manual result set handling and object conversion and keep the application portable to all supported SQL databases with little performance overhead.

Hibernate-annotations
The Hibernate Annotations is based on the final release of the JPA 2 specification (aka JSR-317) and supports all its features (including the optional ones). Hibernate specific features and extensions are also available through unstandardized, Hibernate specific annotations.

Hibernate-commons-annotations
Hibernate - common annotation reflection code used in support of annotation processing.

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-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.

Jackson-core-asl
A core library of Jackson parser, a high-performance JSON processor (parser, generator). Core jar contains streaming JSON parser and generator interfaces and implementations.

Jackson-jaxrs
Jackson-jaxrs contains class(es) needed to make a JAX-RS implementation like Jersey use Jackson for simple, convenient and lightning fast binding of JSON to/from Java objects. Jaxrs package depends on mapper jar (and transitively on core as well). Jaxrs is only needed if you use Jackson via JAX-RS, and not for stand-alone usage.

Jackson-mapper-asl
Optional Mapper jar contains functionality for data binding. TreeMapper to build "trees" consisting of JsonNodes, and write them as JSON. ObjectMapper to construct Plain Old Java Objects (POJOs, aka Beans) from JSON, and similarly write Java objects as JSON.

Jackson-xc
Extensions that provide interoperability support for Jackson JSON processor's data binding functionality. Ability to use JAXB annotations (instead of or in addition to basic Jackson annotations)

Javassist
Javassist (Java programming assistant) is a Java library providing a means to manipulate the Java bytecode of an application. Javassist provides the support for structural reflection, i.e the ability to change the implementation of a class at run time. Bytecode manipulation is performed at load-time through a provided class loader.

Jaxb-api

JAXB consists of two parts. First, JAXB contains a compiler that reads a schema and produces the equivalent Java object model. This generated object model captures the structure of XML better than general-purpose APIs like DOM or SAX, making it a lot easier to manipulate XML content.
The second part is an API, through which applications communicate with generated code. This API hides provider-specific implementation code from applications and also provides a uniform way to do basic operations, such as marshalling or unmarshalling. The JAXB API, defined in the javax.xml.bind package, is a set of interfaces through which client applications communicate with code generated from a schema. The center of the JAXB API is JAXBContext, the client's entry point. It provides an abstraction for managing the XML-Java binding information necessary to implement the JAXB binding framework operations: unmarshal, marshal and validate.

These three aspects of JAXB are covered by three separate interfaces. Instances of those interfaces can be created from a JAXBContext object:
1)Unmarshaller: governs the process of deserializing XML data into Java content trees, optionally validating the XML data as it is unmarshalled;
2)Marshaller: governs the process of serializing Java content trees back into XML data;
3)Validator: performs the validation on an in-memory object graph.

JAXBContext is an abstract class defined in the API, so its actual implementation is vendor-dependent. To create a new instance of JAXBContext, you use the static newInstance method. This method takes a list of package names as a parameter. Each schema is compiled into a single package, which means that you can assemble them at run-time by providing multiple package names.

Jaxb-impl

JAXB (JSR 222) reference implementation. The goal of the JAXB project is to develop and evolve the code base for the Reference Implementation (RI) of JAXB, the Java Architecture for XML Binding. The JAXB specification is developed through the Java Community Process following the process described at jcp.org. This process involves an Expert Group with a lead that is responsible for delivering the specification, a reference implementation (RI) and a Technology Compatibility Kit (TCK). The primary goal of an RI is to support the development of the specification and to validate it. Specific RIs can have additional goals; the JAXB RI is a production-quality implementation that is used directly in a number of products by Oracle and other vendors.

Jersey-client

Jersey client class is the main configuration point for building a RESTful web service client. You use it to configure various client properties and features and indicate which resource providers to use. Creating an instance of a Client is an expensive operation, so try to avoid creating an unnecessary number of client instances. A good approach is to reuse an existing instance, when possible.

Jersey-core
The core library for the jersey server.

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.

Jersey-server
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.

jettison
A StAX implementation for JSON. Jettison is a collection of Java APIs (like STaX and DOM) which read and write JSON. This allows nearly transparent enablement of JSON based web services in services frameworks like CXF or XML serialization frameworks like XStream. There are currently two included conventions for mapping JSON to XML. The first, is BadgerFish which implements the full XML infoset in JSON using various techniques. The second, is called the "mapped" convention. It allows you to manually map XML namespaces to JSON element prefixes.

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

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.

Mysql-connector-java

MySQL provides connectivity for client applications developed in the Java programming language through a JDBC driver, which is called MySQL Connector/J. MySQL Connector/J is a JDBC Type 4 driver. Different versions are available that are compatible with the JDBC 3.0 and JDBC 4.0 specifications. The Type 4 designation means that the driver is pure-Java implementation of the

MySQL protocol and does not rely on the MySQL client libraries.

poi

Apache POI - Java API To Access Microsoft Format Files. The Apache POI Project's mission is to create and maintain Java APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). In short, you can read and write MS Excel files using Java. In addition, you can read and write MS Word and MS PowerPoint files using Java. Apache POI is your Java Excel solution (for Excel 97-2008). We have a complete API for porting other OOXML and OLE2 formats and welcome others to participate. A major use of the Apache POI api is for Text Extraction applications such as web spiders, index builders, and content management systems.

Poi-ooxml

Apache POI - Java API To Access Microsoft Format Files. POI supports the ISO/IEC 29500:2008 Office Open XML file formats since version 3.5. A significant contribution for OOXML support came from Sourcesense,an open source company which was commissioned by Microsoft to develop this contribution. This link spurred controversy, some POI contributors questioning POI OOXML patent protection regarding Microsoft's Open Specification Promise patent license.

Poi-ooxml-schemas

Apache POI - Java API To Access Microsoft Format Files. Poi-ooxml requires poi-ooxml-schemas. This is a substantially smaller version of the ooxml-schemas jar (ooxml-schemas-1.1.jar for POI 3.7 or later, ooxml-scheams-1.0.jar for POI 3.5 and 3.6). The larger ooxml-schemas jar is normally only required for development. The OOXML jars require a stax implementation. Previously we suggested "geronimo-stax-api_1.0_spec", but now "stax-api-1.0.1" is used for better compatibilty with other Apache projects. Any compliant implementation should work fine though.

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-simple

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.

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-beans

Spring beans is a library that provides container [BeanFactory] which instantiates, configures, and manages a number of beans.
These beans typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the BeanFactory (although some dependencies may not be visible as configuration data, but rather be a function of programmatic interactions between beans at runtime). A BeanFactory is represented by the interface org.springframework.beans.factory.BeanFactory, for which there are multiple implementations. The most commonly used simple BeanFactory implementation is org.springframework.beans.factory.xml.XmlBeanFactory. (This should be qualified with the reminder that ApplicationContexts are a subclass of BeanFactory, and most users end up using XML variants of ApplicationContext).For many usage scenarios, user code will not have to instantiate the BeanFactory or ApplicationContext, since Spring Framework code will do it. Bean definitions inside a DefaultListableBeanFactory variant (like XmlBeanFactory) are represented as BeanDefinition objects, which contain (among other information) the following details:a class name: this is normally

the actual implementation class of the bean being described in the bean definition. However, if the bean is to be constructed by calling a static factory method instead of using a normal constructor, this will actually be the class name of the factory class. bean behavioral configuration elements, which state how the bean should behave in the container (i.e. prototype or singleton, autowiring mode, dependency checking mode, initialization and destruction methods)

constructor arguments and property values to set in the newly created bean. An example would be the number of connections to use in a bean that manages a connection pool (either specified as a property or as a constructor argument), or the pool size limit. other beans a bean needs to do its work, i.e. collaborators (also specified as properties or as constructor arguments). These can also be called dependencies.

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-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.

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-dao

Spring Framework's Data Access Object (DAO) support provides integration of heterogeneous Java Database Connectivity (JDBC) and Object-Relational Mapping (ORM) products. Spring Framework makes it easy to integrate these solutions, easing tasks pertaining to resource management and transaction management. This allows for an environment in which multiple correlated projects may exist together, using DAO implementations that use different persistence strategies. With Spring Framework, DAO support is provided for JDBC, Hibernate, iBATIS, Java Data Objects (JDO), Java Persistence API (JPA), Oracle's TopLink and Common Client Interface (CCI). Support for other ORM solutions is provided through the Spring Modules project; currently, there's support for Apache's ObjectRelationalBridge (OJB) and O/R Broker. By

extending one of the abstract DaoSupport sub-classes, your DAO implementation will have available to it a Spring Framework template and some convenience methods for accessing resources. The Spring Framework template object provides convenience methods for common operations with the underlying persistence solution. It also provides an execute method which allows for executing arbitrary code in a application callback object within the constraints of a transaction.

Spring-hibernate3

Spring-hibernate 3 Library provides hibernate support to Spring framework.

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-oxm

Spring OXM stands for Spring Object XML Mappers and it is a module available in Spring to ease the mapping between java objects and XML documents.The module is extensible and hence it provides integration with various popular frameworks like Castor, JAXB, XmlBeans and XStream. Castor defines an xml binding definition file which defines how a java object should be mapped to its xml equivalent.Marshalling and un-marshaling in Spring are represented through separate interfaces 'Marshaller' and 'Unmarshaller' in the package 'org.springframework.oxm'. The class 'CastorMarshaller', though named as marshaller will do both marshalling and un-marshalling. One mandatory property of 'CastorMarshaller' is 'mappingLocation' which points to the xml file containing the binding definition.

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-ws-core

Spring Web Services (Spring-WS) is a product of the Spring community focused on creating document-driven Web services.Spring Web Services aims to facilitate contract-first SOAP service development, allowing for the creation of flexible web services using one of the many ways to manipulate XML payloads. The product is based on Spring itself, which means you can use the Spring concepts such as dependency injection as an integral part of your Web service. People use Spring-WS for many reasons, but most are drawn to it after finding alternative SOAP stacks lacking when it comes to following Web service best practices. Spring-WS makes the best practice an easy practice. This includes practices such as the WS-I basic profile, Contract-First development, and having a loose coupling between contract and implementation.

The other key features of Spring Web services are:
Powerful mappings: You can distribute incoming XML requests to any object, depending on message payload, SOAP Action header, or an XPath expression.
1)XML API support: Incoming XML messages can be handled not only with standard JAXP APIs such as DOM, SAX, and StAX, but also JDOM, dom4j, XOM, or even marshalling technologies.
2)Flexible XML Marshalling: The Object/XML Mapping module in the Spring Web Services distribution supports JAXB 1 and 2, Castor, XMLBeans, JiBX, and XStream. And because it is a separate module, you can use it in non-Web services code as well.

3)Reuses your Spring expertise: Spring-WS uses Spring application contexts for all configuration, which should help Spring developers get up-to-speed nice and quickly. Also, the architecture of Spring-WS resembles that of Spring-MVC.
4)Supports WS-Security: WS-Security allows you to sign SOAP messages, encrypt and decrypt them, or authenticate against them.
5)Integrates with Acegi Security: The WS-Security implementation of Spring Web Services provides integration with Acegi Security. This means you can use your existing Acegi configuration for your SOAP service as well.

Spring-xml

Spring - xml is an XML library for spring framework. The Spring framework is a dependency injection engine. Glue Stick has support for Spring bean definition XML files and several of the features of the Spring framework. The Spring framework documentation has detailed information about the Spring XML bean definition format. Using Spring XML bean definitionsTo use Spring XML bean definitions, put the Glue Stick core and Glue Stick Spring JAR:s on the classpath and add the Spring XML bean definition files to the BeanContainerLoader. If Glue Stick is run on Java 5, StAX XML parsing API and implementation JAR:s have to be present on the classpath (see the StAX reference implementation). StAX is included in Java 6 and newer.For the bean definition language auto detection to work, the files have to have the file name extension .xml. If they have any other extension, the BeanContainerLoader can be instructed to use the Spring XML bean definition loader by using the fully qualified name of the SpringXmlBeanDefinitionLoader class as the argument to the BeanContainerLoader.setBeanDefinitionLoaderClass(String) method.

Stax-api

StAX is a standard XML processing API that allows you to stream XML data from and to your application. Streaming API for XML (StAX) is an application programming interface (API) to read and write XML documents, originating from the Java programming language community.

Traditionally, XML APIs are either:
1)DOM based - the entire document is read into memory as a tree structure for random access by the calling application
2)event based - the application registers to receive events as entities are encountered within the source document.

Both have advantages; the former (for example, DOM) allows for random access to the document, the latter (e.g. SAX) requires a small memory footprint and is typically much faster. These two access metaphors can be thought of as polar opposites. A tree based API allows unlimited, random access and manipulation, while an event based API is a 'one shot' pass through the source document. StAX was designed as a median between these two opposites. In the StAX metaphor, the programmatic entry point is a cursor that represents a point within the document. The application moves the cursor forward - 'pulling' the information from the parser as it needs. This is different from an event based API - such as

SAX - which 'pushes' data to the application - requiring the application to maintain state between events as necessary to keep track of location within the document.

wsdl4j

Java stub generator for WSDL.The Web Services Description Language for Java Toolkit (WSDL4J) allows the creation, representation, and manipulation of WSDL documents. Is the reference implementation for JSR110 'JWSDL' (jcp.org).

Xml-apis

Java API for XML Processing (JAXP) enables applications to parse, transform, validate and query XML documents using an API that is independent of a particular XML processor implementation. JAXP provides a pluggability layer to enable vendors to provide their own implementations without introducing dependencies in application code. Using this software, application and tool developers can build fully-functional XML-enabled Java applications for e-commerce, application integration, and web publishing.

xmlbeans

XMLBeans is a tool that allows you to access the full power of XML in a Java friendly way. The idea is that you can take advantage of the richness and features of XML and XML Schema and have these features mapped as naturally as possible to the equivalent Java language and typing constructs. XMLBeans uses XML Schema to compile Java interfaces and classes that you can then use to access and modify XML instance data. Using XMLBeans is similar to using any other Java interface/class, you will see things like getFoo or setFoo just as you would expect when working with Java. While a major use of XMLBeans is to access your XML instance data with strongly typed Java classes there are also API's that allow you access to the full XML infoset (XMLBeans keeps XML Infoset fidelity) as well as to allow you to reflect into the XML schema itself through an XML Schema Object model.

Spring-orm

Spring Object/Relational Mapping

commons-collections

Java Collections Framework extensions. Java Commons Collections Framework accelerates the Java applications development, being an empowered data structure. For collection handling, it has now taken a position of standard in Java. Java Collections provides implementations, new interfaces & utilities. They are built at JDK classes top.