ISA Modular Framework
&
Guidelines

TOC – Transfer Of Concepts

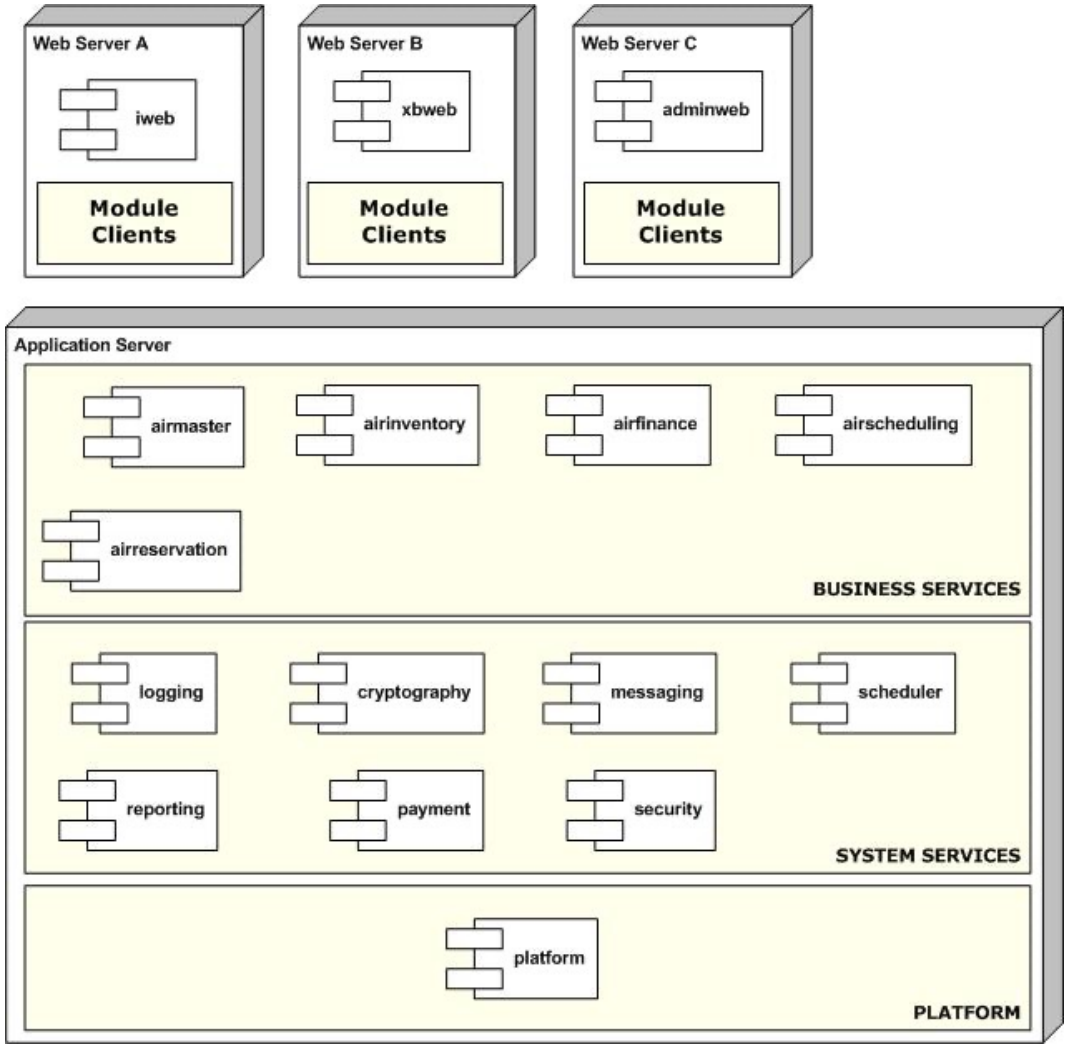
By Mohamed Nasly

Version 1.0 Draft

Overview

- The overall architecture is driven by the concept of **Product Line Engineering**, which relies on well defined elements and processes for assembling a working whole from individual elements
- A key concept in developing flexible and extensible architecture is strong encapsulation of coherently related functionalities into logically separable units – **modules**
- **ISA platform** provides services to build and manage modules

ISA Framework



Key Strategies

- Layering & Controlled specialization

  The elements with constrained dependencies are grouped into layers. The upper-level layers depend only on the lower-level layers. The controlled specialization enforces the concept of defining common services at lower-level layers while specialized services per vertical/deployment are defined at the higher-level layers.

- Modularization

  Key to scalable and usable architecture. A module provides a well managed and well defined set of service interfaces and contracts to the application.

- Dynamic Configuration

  Application is assembled wiring modules through the configurations loaded from a central configuration repository at start up time.

What is a module?

- A large grained unit of software; Encapsulates coherently related functionalities into logically and physically separable unit
- System architecture is described in terms of modules and their interactions
- A module abstracts unit of design, unit of development, unit of configuration and unit of management

Platform Module

- Provides services for modules' configurations initialization, lifecycle management and inter-module communication.
- Serves as the container for the modules.
- Built on top of Spring IoC Container using bean factories.
- Takes care of loading modules by reading configurations from a central configuration repository.

Platform Module Services

- Modules Framework initialization

  The platform module takes care of initializing modules loading modules' configurations from the central configuration repository at the application start up. Configuration repository provides file system based storage for configuration information

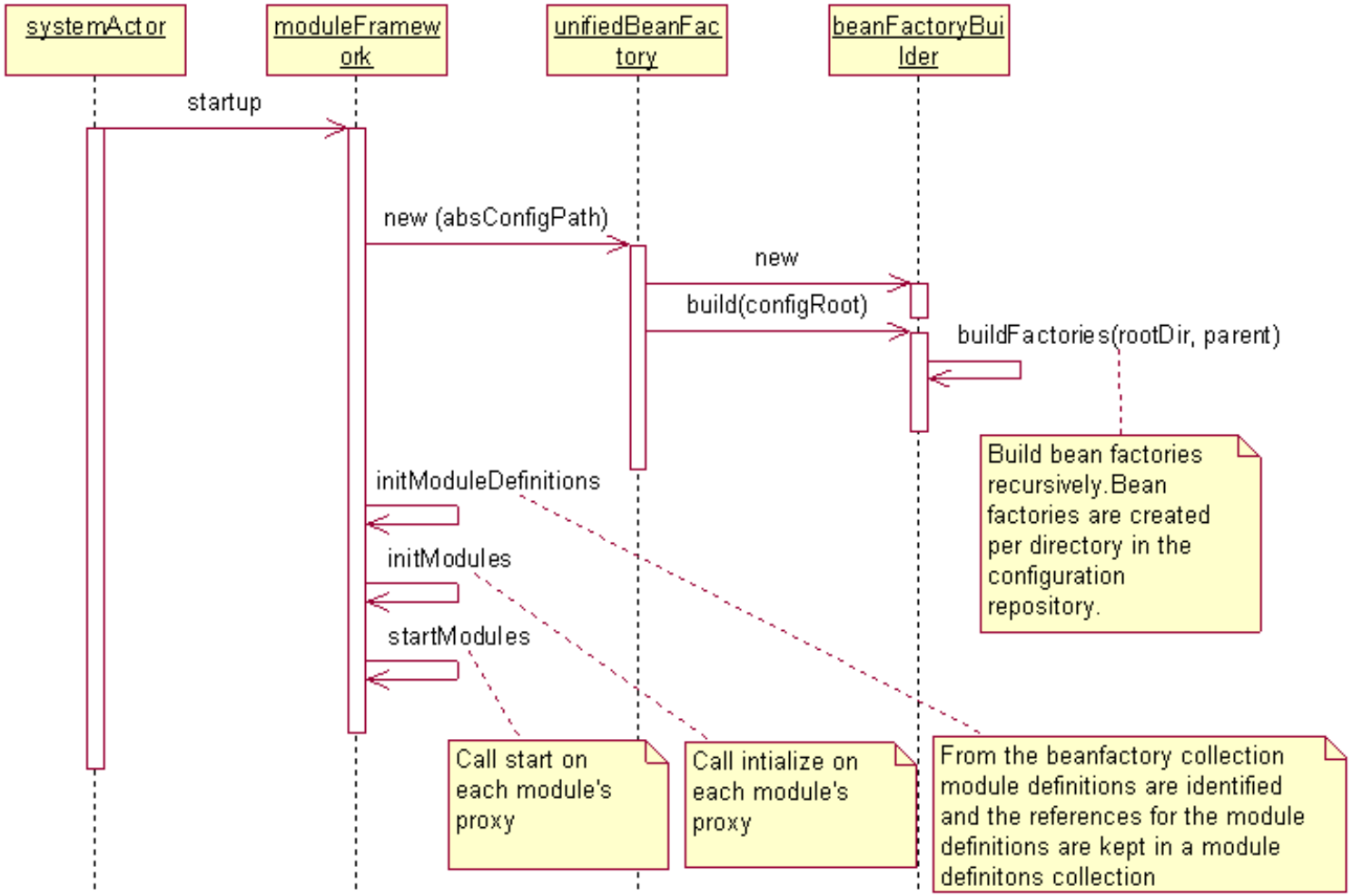
- Module lookup service

  To get service from a module, clients need to obtain the service interface of the module. Look up service facilitates this allowing clients to request for modules' service reference by providing the module name

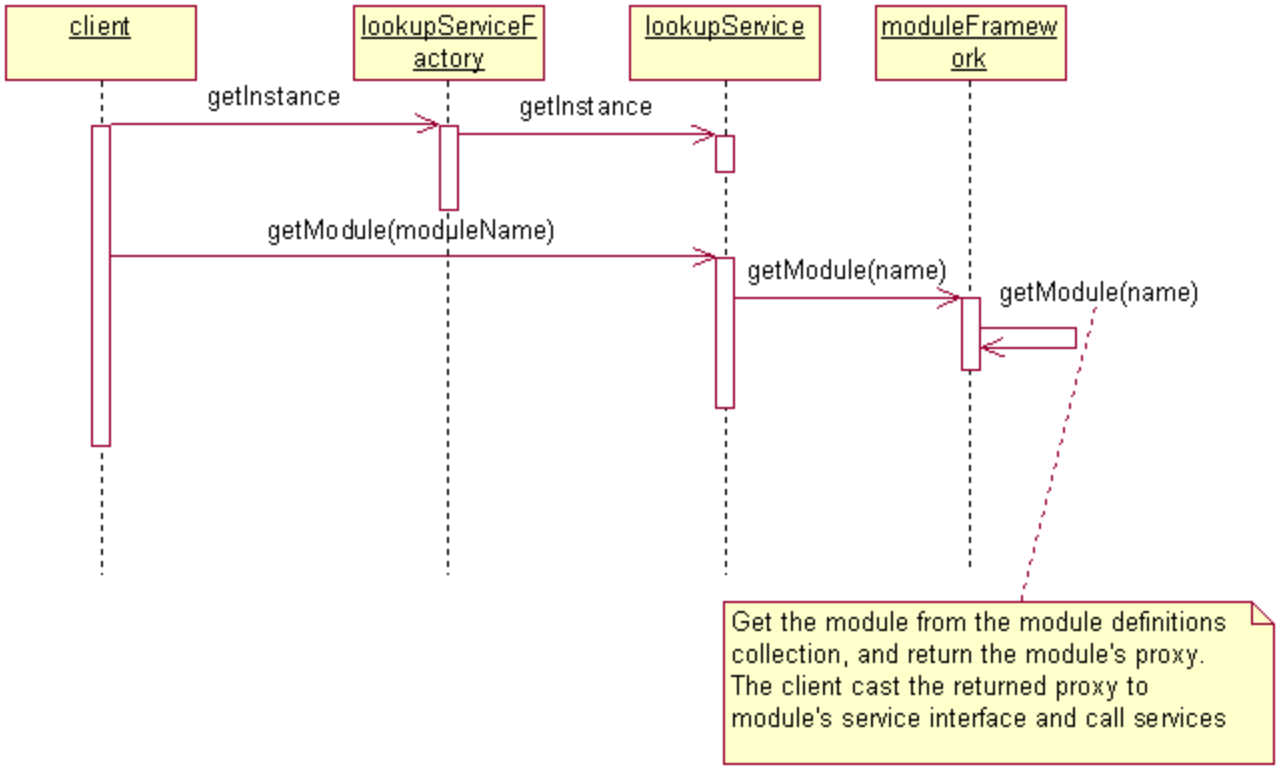
- Bean lookup service

  Look up services also provides facilities for getting references for individual beans configured in module configurations

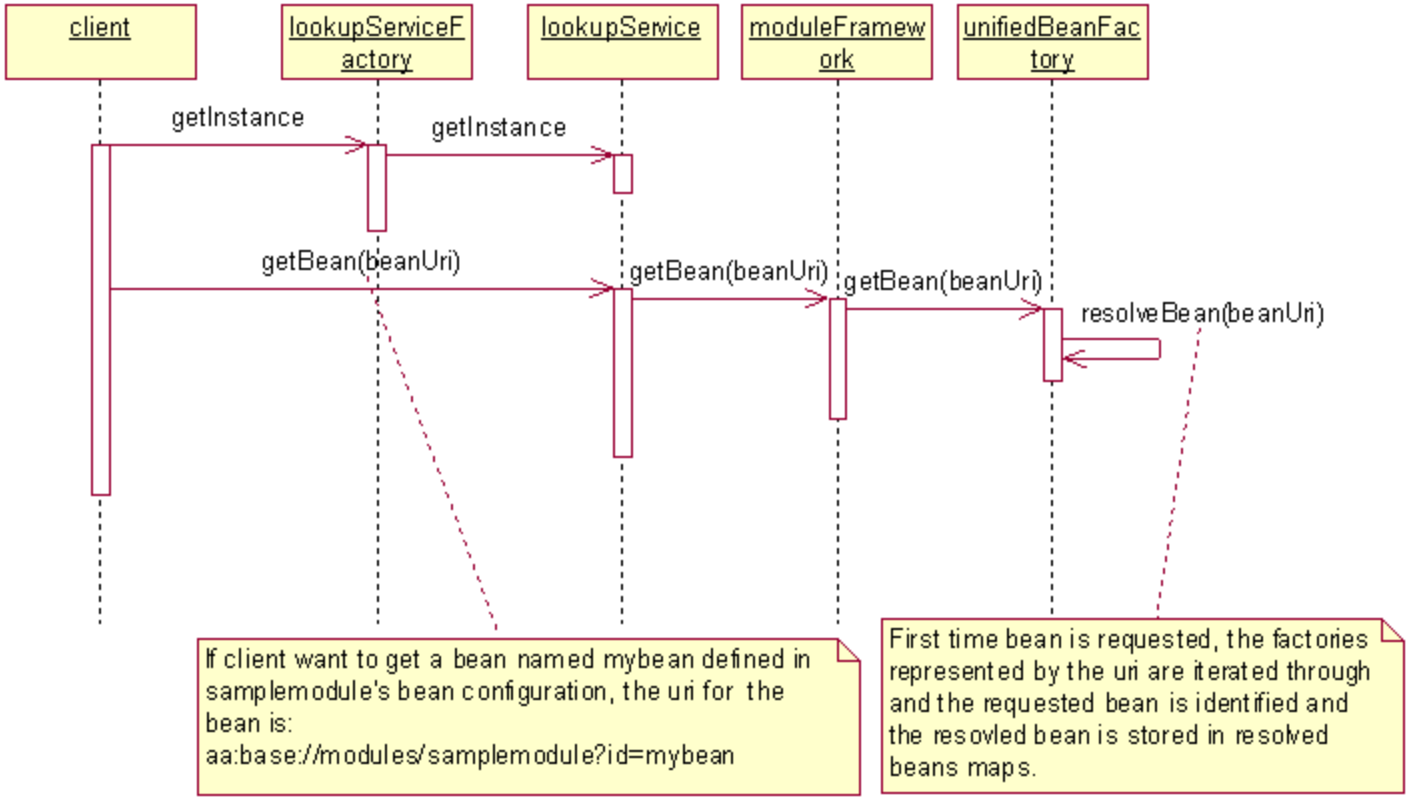
Modules Framework Initialization



Lookup Module



Lookup Bean



Configuration repository

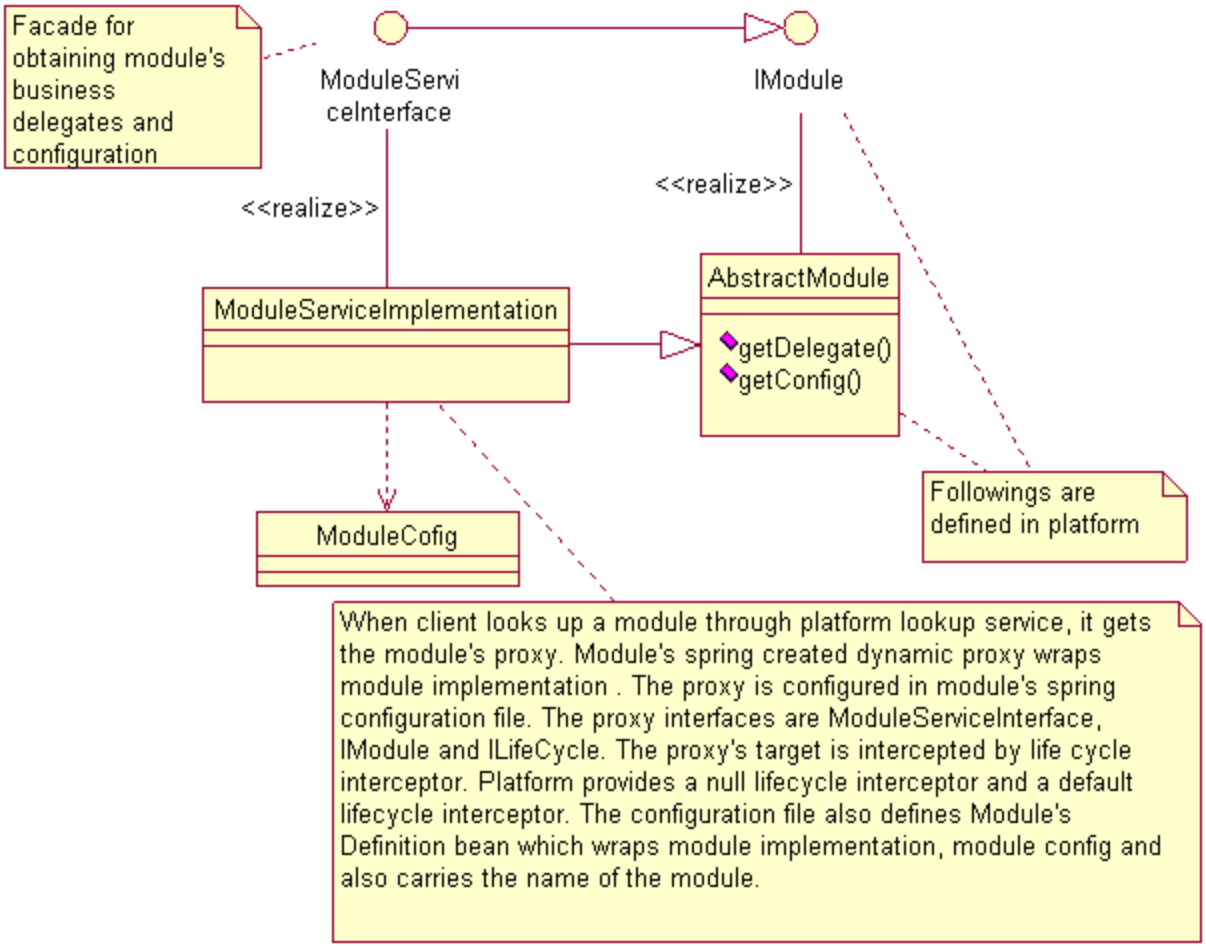
- Central location to keep all modules' configurations; A file system storage for configuration data
- Configurations are specified as spring bean definitions
- Clients refer to resources in the repository with a logical identifier called a URI.
  - URI for a configuration bean defined with bean id sampleBean in samplemodule:

    isa:base//modules/sampemodule?id=sampleBean
- Configuration files has .mod.xml extension

Anatomy of a Module

- Each module has it own configurations, source and build
- A module may share globally defined configurations as well
- A module has Module Definition which specifies module's meta-data used by platform module
- Module has a well defined service interface through which it exposes services to clients; A module may define a custom service interface extending IModule interface provided by platform module
- Module service interface implementation must subclass DefaultModule class provided by platform module

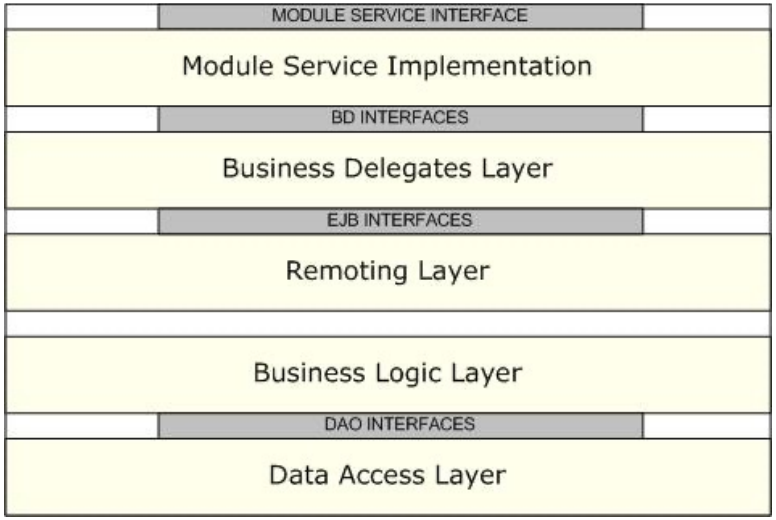
Module Contracts



Module's Configuration

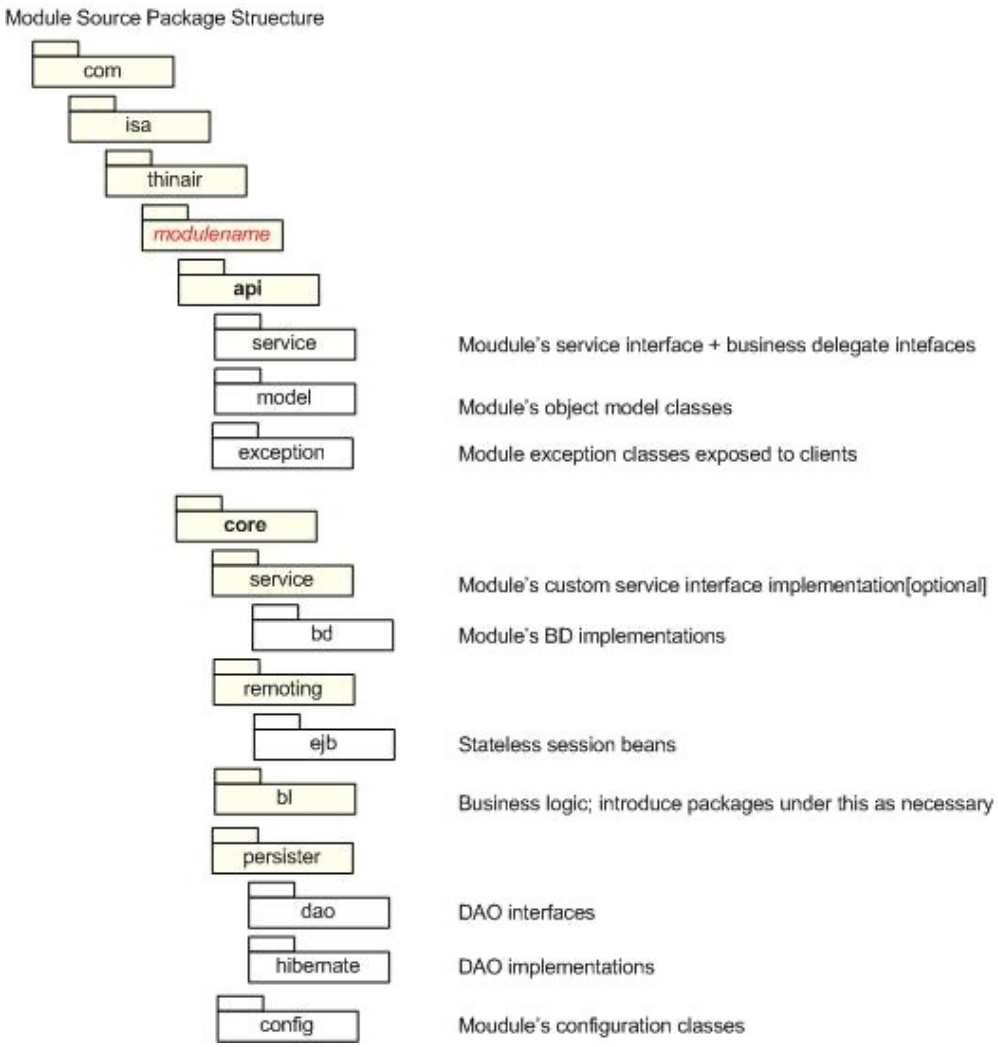
- Carries module definition, hibernate configurations and any other custom spring bean configurations
- Module definition specifies meta-data about the module and followings are defined:
  - Module Name
  - Implementation
  - Proxy
  - Description
- Clients get the reference to module service through platform module lookup service specifying the module name
- Implementation specifies the module's service implementation
- Proxy wraps module's service implementation with interceptors

Layering with in Module



- Data Access layer takes care of data persistence
- Key business functionalities are implemented in business logic layer
- Remoting layer wraps business logic in stateless session beans; both locally and remotely accessible
- Business delegates layer provides location transparency
- Module's service implementation exposes the business delegates

Module's Java Source Packaging



Module's Build

- Jakarta Ant is used for build automation and unit test automation
- Module build is defined reusing the generic targets defined in the global build file – modulebuildutils.xml; Module's build may override globally defined build properties to customize the its build
- Third party libraries needed for the module build are referred from <project-root>/repository/lib
- Module distributions are copied to <project-root>/repository/modules so that other modules can share
- Refer comments in modulebuilddepends.xml and modulebuildutils.xml for further information

General Coding Guidelines

- Provide structured javadoc comments for every operation specifying followings:
  - The operation
  - Parameters passed
  - Exceptions
  - Return values
- Do not duplicate the javadoc comment in the interface and the corresponding implementation class
- Name of the interfaces should start with prefix “I”
- Follow Sun Java coding standards

Exception Handling Guidelines

- Exceptions occurring within a module should be propagated to the client wrapping the exception in module's API level exception. i.e. Each module should expose only one checked exception and one unchecked exception.
- Each checked exception should subclass generic checked exception defined by the platform, and each unchecked exception defined at module level should subclass generic unchecked exception defined by the platform.
- If the service requestor has no clue of recovering from the exception and exceptional condition occurred is fatal, wrap the exception in the module's unchecked exception and throw to the requestor.
- With in a module, all the persistence layer exceptions are wrapped in unchecked data access exception defined by the platform, all the business layer exception are wrapped in unchecked business exception defined by platform before propagating to the higher context.
- Business delegates layer should ensure all the exceptions propagated to the higher context is of type module exception.

Logging Guidelines

- Provide enough context information in log messages so that logging assists system diagnosis
- A given exception should be logged only at one place
- Use log levels – debug, info, warn, error, fatal - appropriately

JUnit Testing Guidelines

- All the JUnit test must sub class PlatformTestCase class defined in platform; It takes care of loading the configurations
- All the test classes should be named with prefix “Test”
- Do not specify try-catch blocks with in test methods; rather declare throws
- When tests access persistence store, make sure state of the database is NOT changed by the test execution
- Make tests independent of existing data in the database
- Store test resources in src/test/resources folder; this folder is added to classpath when tests are run through ant build
- Minimize the tests for remoting and above layers as much as possible by implementing tests thoroughly for business logic and below layers.

System Modification Guidelines

- Always do the changes on the latest version checked out from the CVS
- Before checking in changes to the CVS make sure to unit test the changes thoroughly, to ensure integrity of CVS repository
- Provide explicit, meaningful comments when checking in changed sources
- Do not check in any ant build generated files

References

- Spring documentation
- Product line engineering (SEI website - http://www.sei.cmu.edu)

- Platform & Sample module implementations are found at

  CVS server = /projects/air_arabia, project = ThinAir