**Resumen framework**

Reuso de software

Conceptos e ideas que funcionan, diseños o estrategias de diseño (que imitamos, implementamos), patrones de diseño, algoritmos de búsqueda, arquitecturas, funciones y estructuras de datos, componentes y servicios (que invocamos), aplicaciones completas que adaptamos e integramos.

¿Por qué?

**Para aumentar la productividad**

* Reducir los costos de desarrollo y mantenimiento
* Reducir los tiempos de entrega y puesta en el mercado
* Agilidad de mercado y customización en masa

**Para aumentar la calidad**

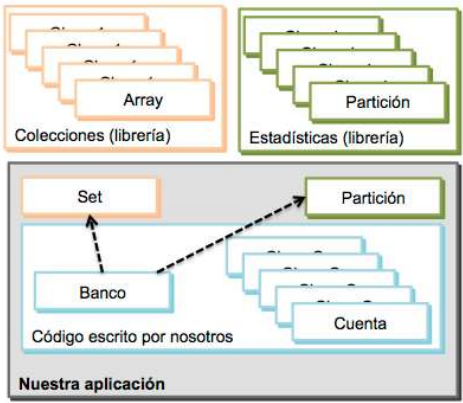
* El reúso aprovecha mejoras y correcciones
* el tiempo da madurez y confiabilidad
* Se reaprovecha el conocimiento de los especialistas (que queda encapsulado en componentes reusables)
* Ayuda a la implementación e imposición de estándares

Además de reutilizar el código, se reutiliza el tiempo del programador en pensar la idea, conocer el dominio del problema y encontrar los bugs.

**Librería**

Resuelven problemas comunes a la mayoría de las aplicaciones, por ejemplo: manejo de archivos, funciones aritméticas, colecciones, fechas. Cada clase en la librería resuelve un problema concreto, es independiente del contexto de uso, no espera nada de nuestro código, y generalmente independiente de otras clases en la librería. Nuestro código controla/usa a los objetos de las librerías.

Artefactos reutilizables, tienen algo en común, aunque sean varias se puede utilizar/ aprender solo una parte de la librería.



Ejemplos de librerias en java:

* Math (apache common): Clases que implementan componentes autocontenidos de matemáticas y estadísticas.
* java.útil.collections: Clases que implementan estructuras de datos (colecciones) y algoritmos para manipularlas

**Framework**

Un framework se puede ver como el resultado de refactorizar una aplicación, no solo para mejorar su calidad interna sino para abstraer lo que es común todas una familia de aplicaciones similares. En ese proceso de refactoring se identifica lo que se mantendrá constante y será, a partir de ese momento, responsabilidad de los desarrolladores del framework y lo que podrá variar de aplicación a aplicación y será responsabilidad de los desarrolladores de aplicaciones que usen el framework.

* Un framework es una aplicaciónón “semicompleta”, “reusable”, que puede ser especializada para producir aplicaciones a medida.
* Un conjunto de clases concretas y abstractas, relacionadas para proveer una arquitectura reusable para una familia de aplicaciones relacionadas.

**Frameworks Orientados a Objetos**

* Los desarrolladores incorporan el framework en sus aplicaciones y lo especializan para cubrir sus necesidades
* El framework define el esqueleto, y el desarrollador define sus propias características para completar el esqueleto
* A diferencia de un toolkit o librería un framework provee una estructura coherente en lugar de un conjunto de “clases útiles”

Se puede pensar como que el framework tiene cosas hechas y con eso podemos construir más cosas, el framework toma las decisiones de diseño, cuando lo usamos, aceptamos estas decisiones.

* Proveer una solución reusable para una familia de aplicaciones/problemas relacionados, ejemplo: interfaces web, editores gráficos, aplicaciones colaborativas, gestores de contenidos
* Las clases en el framework se relacionan (herencia, conocimiento, envío de mensajes) de manera que resuelven la mayor parte del problema en cuestión – conforman un todo
* El código del framework controla al nuestro. (Inversión de control)

La arquitectura en tiempo de ejecuciónón de un framework se caracteriza por una inversión de control. Esta arquitectura permite que los pasos de procesamiento de aplicaciones canónicas sean personalizados por objetos manejadores de eventos que son invocados a través del mecanismo de despacho reactivo del framework. Cuando ocurren eventos, el despachador del framework reacciona invocando métodos de gancho en objetos manejadores pre-registrados, los cuales realizan un procesamiento específico de la aplicación en los eventos. La inversión de control permite que el framework (en lugar de cada aplicación) determine qué conjunto de métodos específicos de la aplicación invocar en respuesta a eventos externos (como mensajes de ventana enviados por usuarios finales o paquetes que llegan a puertos de comunicación específicos).

Inversión de control y falta de flujo de control explícito. Las aplicaciones escritas con frameworks pueden ser difíciles de depurar ya que el flujo de control "invertido" del framework oscila entre la infraestructura del framework independiente de la aplicación y las devoluciones de llamada de los métodos específicos de la aplicación. Esto aumenta la dificultad de "avanzar paso a paso" a través del comportamiento en tiempo de ejecución de un framework dentro de un depurador, ya que el flujo de control de la aplicación es impulsado implícitamente por devoluciones de llamada y los desarrolladores pueden no entender o tener acceso al código del framework.

**Frameworks de infraestructura**

* Estos frameworks ofrecen una infraestructura portable y eficiente sobre la cual construir una gran variedad de aplicaciones
* Algunos de los focos de este tipo de frameworks son: interfaces de usuario (desktop, web, moviles), seguridad, contenedores de aplicación, procesamiento de imágenes, procesamiento de lenguaje, comunicaciónes
* Atacan problemas generales del desarrollo de software, que por lo general no son percibidos completamente por el usuario de la aplicación (es decir, resuelven problemas de los programadores, no de los usuarios)
* Es común que se los incluya como parte de plataformas de desarrollo (Java tiene los suyos, al igual de .NET)

**Frameworks de integración**

* Estos frameworks se utilizan comunmente para integrar componentes de aplicación distribuidos (p.e., la base de datos con la aplicación y ésta con su cliente liviano)
* Los frameworks de integración (o frameworks middleware) estandiseñados para aumentar la capacidad de los desarrolladores de modularizar, reusar y extender su infraestrcutura de software para que trabaje transparentemente en un ambiente distribuido
* El mercado de los frameworks de integración es muy activo y, al igual que ciertos frameworks de infraestructura, se han vuelto commodities(mercancía de uso común)
* Ejemplos: Object Relational Mapping, Message Oriented Middleware, Orchestration Frameworks

**Frameworks de aplicación (o Enterprise)**

* Estos frameworks atancan dominios de aplicación amplios que son pilares fundamentales de las actividades de las empresas
* Ejemplos de frameworks enterprise son aquellos en el dominio de los ERP (Enterprise Resource Planning), CRM (Customer RelationshipManagement) Gestión de documentos, Cálculos financieros.
* Los problemas que atacan derivan directamente de las necesidades de los usuarios de las aplicaciones y por tanto hacen que el retorno de la inversión en su desarrollo/adquisición sea mas evidente y justificado
* Un framework entreprise puede encapsular el conocimiento y experiencia de muchos años de una empresa, tranformandose en la clave de su ventaja competitiva y su mas preciado capital
* Son la base para la implementación de líneas de productos (SPL)

Algunos frameworks conocidos

* jUnit: Framework de aplicación/infraestructura que resuelve la automatización de tests de unidad
* libGDX: Framework de aplicación, para construir juegos en Java, multiplataforma.

**Frozenspots de un framework**: Todas las aplicaciones construídas con un mismo framework tienen aspectos en común que no podemos cambiar.

**Hotspots de un framework**: El framework ofrece puntos de extensión que nos permiten introducir variantes y así construir aplicaciones diferentes.

Caja blanca vs Caja negra

* Los puntos de extensión pueden implementarse en base a herencia o en base a composición
* A los frameworks que utilizan herencia en sus puntos de extensión, les llamamos de Caja Blanca (Whitebox)
* A los que utilizan composición les llamamos de Caja Negra (blackbox)
* La mayoría de los frameworks están en algún lugar en el medio– Algunos usuarios los ven como caja negra – Otros como caja blanca

Para separar la implementación de la parte constante (a lo que llamamos **frozenspot)** y de las partes que varían (a las que llamamos **hotspots)**, se introducen en el framework puntos de extensión. Estos tienen la forma de plantillas que ofrecen ganchos para que, mediante herencia o composición, los programadores de aplicaciones definan el comportamiento variable. Algunos patrones de diseño (particularmente el template method y el strategy) son frecuentemente utilizados para introducir esos puntos de extensión.

 La construcción y uso de frameworks es una estrategia para modularizar y reutilizar, que mejora la calidad del software y aumenta la productividad de quienes lo hacen. Un framework, visto como módulo de software, no solo encapsula operaciones y estructuras de datos reusables sino que las combina para resolver, con demostrada eficacia, los requerimientos comunes a una familia de aplicaciones. En pocas palabras, encapsula el diseño y la implementación de la parte más compleja de dichas aplicaciones y permite su reutilización.

Un framework es en sí mismo un módulo de software reutilizable. Muchas aplicaciones se construirán sobre él. El framework será independiente de las aplicaciones. Sin embargo, las aplicaciones dependerán de que la interface (conjunto de puntos de extensión) y comportamiento del framework se mantenga estable. Esto es importante dado que, a lo largo de su vida, el framework será foco de contínuo refactoring y de mejoras para eliminar bugs y hacerlo más efectivo.