APUNTE TEORIA FRAMEWORKS

Vimos escribir o utilizar código reusable, porque es tan frecuente reutilizar código? .. Beneficios y desafíos. Y que forma de código re utilizable hay.

Tambien re utilizar diseños (Patrones)

Frameworks y librerías.

Vemos Librerias para entender loq hace un framework especial o hace especial a cada uno.

Librerias: conjunto de clases **concretas**, que podemos utilizar una a una. Agrupadas en lbireria porq responden a un tema en común. No esq hay q saber utilizarlas a todas. Cuando utilizo librerías de clases aveces toco 1 o 2 nada mas. **No hay inversión de control**

Frameworks: conjunto de clases pero ya no son concretas para ser utilizadas de 1, sino q **hay abstractas, concretas, herenci**a y se mandan mensajes entre si. Esas clases q hacen al framework resuelven una gran parte de la aplicación. Es semi-completo , falta q escriba loq hace especial a MI APLICACIÓN. **Hay inversión de control**

Gran parte del tiempo el control esta en el código framework y cada tanto envía mensajes a objetos q yo tuve q programar para usar ese framework. Para usar ese framework dependiendo loq yo necesite (Puede ser solo la interfaz, o solo la persistencia de datos etc) Dependiendo eso el hilo de ejecución (o control) este en gran medida en el framework q cada tanto envía mensajes al código q yo tengo q escribir.

Inversión de control: El código de framework controla el código q yo escribo.

Cuando uno habla de artefactos re-utilizables debemos pensarlo con 2 gorros . El que desarrolla ese framework, quien lo usa.

El que lo construye: Tiene la complejidad de construir algo q sea re-usable y en cierta medida es incompleto y el usuario tenga q implementar clases, métodos para llegar a algo completo. Me debo a ciertos compromisos q tomo con ellos. (Por ejemplo si yo una primera versión de mi framework para implementar una funcionalidad hay q subclasificar ciertas clases e implementar ciertos métodos no deberia cambiarlo) Esta es otra dificultad, las decisiones de diseño me atan a futuro, no son tan difíciles de cambiar como las del Que lo usa

El que lo Usa: Hay ciertas decisiones q no se pueden cambiar acerca de como se construyen, como funciona.. Solo puedo cambiar o agregar variabilidad desde el lado donde elq diseño el framework

dejo previsto. **FROZENSPOT:** Parte q ofrece el framework q no va cambiar . **HOTSPOT :** Aquellos aspectos de loq puedo construir en el framework q puedo cambiar en el uso a uso en la aplicacion)

Spoiler próxima clase: Template method en frameworks como UNA FORMA de generar inversión de control y dejar GANCHOS para los q usen el framework agreguen funcionalidad. Ver el Logging y ver si hay patrones.

Caja negra y caja blanca: terminología q se usa mucho en la literatura para caractersiar un framework. Lo interesante de este concepto esq es una forma de describir la madurez del framework . Al principio para usarlo requeire q uno entiende como esta diseñado o construido, donde pasan las cosas, que objetos de que clases andan dando vueltas y por lo general en la vida de un framework se usa mucho subclasificacion… A medida que el framework va teniendo cada vez mas usos,. Se tiene mas experiencia va evolucionando a composición de objetos y configuración en donde cada vez hay que saber menos de lo que pasa adentro. La forma en que se decidió describir esos dos extremos.

Caja negra: Sabes cada vez menos, instancias y configuras objetos. Es más fácil de utilizar para usuarios finales.

Caja blanca: Cuando tenes que saber mucho de loq pasa dentro y tenes que usar herencia. Subclasificar, saber que cosas heredas, que mensajes existen, etc.

Lo que termina pasando en la realidad es que los frameworks van de caja blanca a caja negra . O son EN EL MISMO para algunos casos caja negra y para cosas mas interesantes “como escribir los handlers o los propios formatos” extenderé el framework como caja blanca.