Vertx

Le sue principali caratteristiche sono:

- modularità e compattezza delle dipendenze
- event-driven, fortemente asincrono
- poliglotta
- altamente scalabile
- approccio non-standard
- "unopinionated"
- semplice e pragmatico
- open source (Apache 2)
- supportato dalla fondazione Eclipse

L'approccio del framework Vert.X è tipico dei framework asincroni: ci viene fornito un ambiente all'interno del quale possiamo specificare gli handler che verranno richiamati in seguito a eventi definiti dal framework.

Le conseguenze sono:

- con un po' di disciplina, possiamo mantenere il codice che interagisce con il framework all'interno dell'handler
- gli handler compiono effetti collaterali, non sono quindi funzioni pure
- la struttura dichiarativa costringe ad esternalizzare l'organizzazione del codice
- è in carico a noi la gestione coerente dello stato condiviso

Astrazione

L'astrazione che il modulo vertx-web ci mette a disposizione è un router che ci consente di associare ad una URL una lambda.

Questa può modificare la risposta impostando esito e contenuto. Il modo in cui il nostro codice interagisce con il framework è definito da questa astrazione.

Perchè usare un framework

Un framework fornisce un ambiente all'interno del quale un insieme di casi d'uso fondamentali è reso facile, efficiente e sicuro da implementare.

Un framework per applicazioni web, per esempio, rende facile:

- specificare le rotte a cui rispondere
- costruire le risposte

Lo stesso framework web, cercherà di rendere trasparente, ed eventualmente configurabile:

- la gestione dei dettagli del protocollo
- la sicurezza nel trattamento della comunicazione
- la suddivisione delle risorse fra le varie parti del sistema

Gli autori del framework scelgono le loro priorità fra semplicità d'uso, sicurezza ed efficienza. Come utenti, noi otteniamo automaticamente ogni incremento su ciascuna di queste dimensioni.

Perchè non usare un framework

Un caso d'uso che non è fra quelli prescelti come importanti dagli autori del framework, può essere molto, molto complesso da implementare.

Un framework per applicazioni web, per esempio, può rendere difficile:

- rispondere a richieste esotiche
- controllare finemente l'erogazione della risposta

Lo stesso framework web, potrebbe rendere oscuro, oppure inaspettatamente:

- fallire l'esecuzione di una richiesta a causa di un baco nella sua implementazione
- aprire una falla di sicurezza a causa di un default errato
- permettere il sovraccarico del sistema a causa di una mancata limitazione delle risorse

Gli autori del framework scelgono le loro priorità fra semplicità d'uso, sicurezza ed efficienza.

Come utenti, dipendiamo completamente da loro per l'ordine con cui queste caratteristiche sono implementate, e dobbiamo sopportare i costi di aggiornamento se l'interfaccia fra il framework ed il nostro codice cambia.

Se il framework prende una direzione che non si allinea alle nostre esigenze, oppure cambia radicalmente interfacce esposte richiedendo manutenzione del nostro codice, o ancora peggio se viene abbandonato, possiamo incorrere in problemi anche gravi. La scelta del framework richiede quindi una conoscenza non solo della tecnica, ma anche del mercato e dell'ecosistema che lo circonda.