How to develop an Monolithic Application?
----
project: automobile manufacturing and distribution solution

vision: build an software system, that supports various different aspects in manfacturing automobiles like raw material procurement, inventory, production and aspects of distribution like distributors, marketing and sales etc

Since it is an large-scale enterprise system, it comprises of several functionalities as below:
1. raw material management / procurement
2. inventory management
3. production tracking
4. distributor network
5. sales and marketing

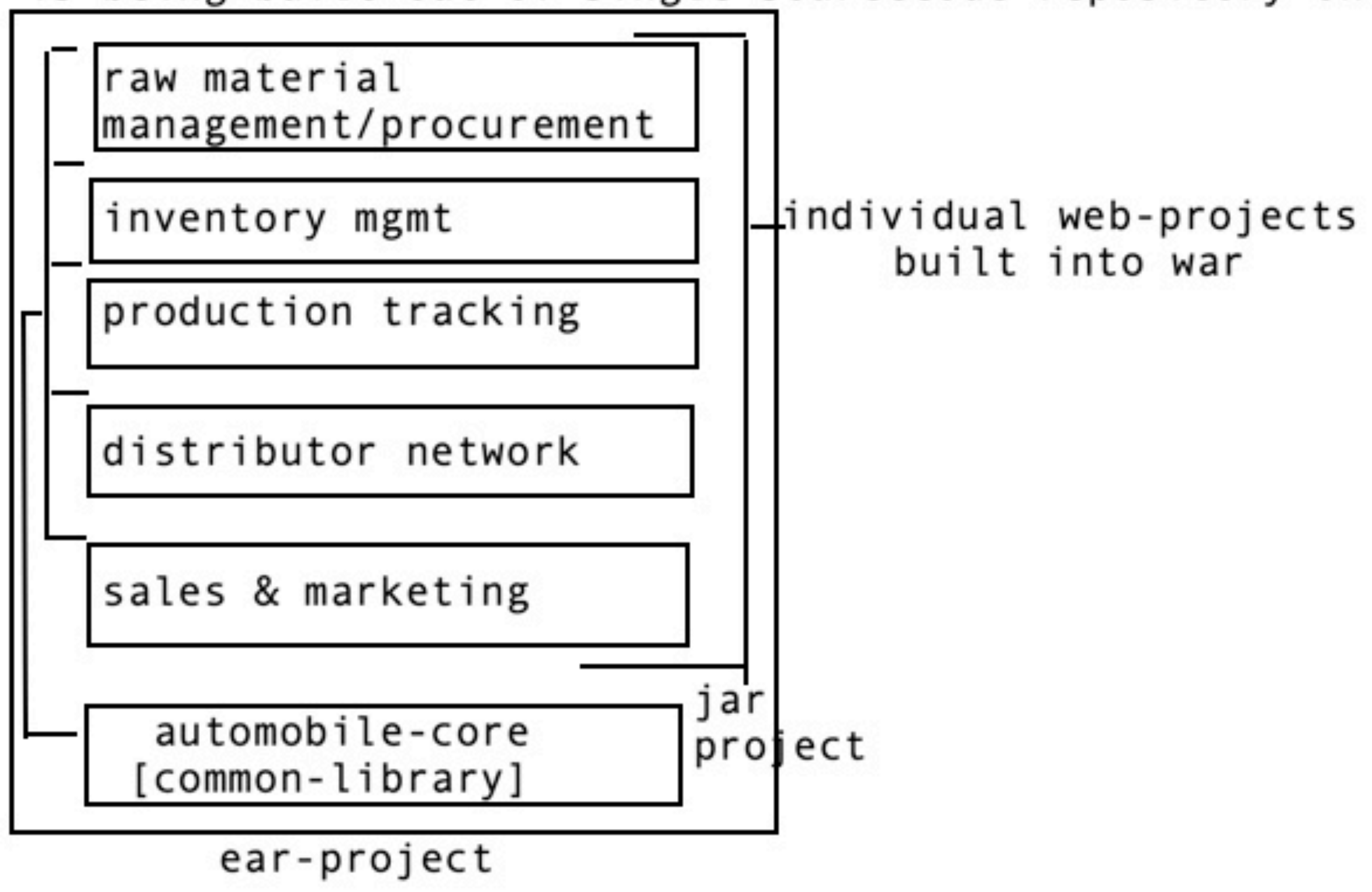
There are #2 approaces in building the above enterprise system with all the functionalities:
1. build the whole application as an single project that comprises of the entire source of all the modules or functionalities

```
automobilemgmtsuite
|-src
  |-main
    |-java [organize the code into package hierarchies based on the modules]
    |-resources
    |-webapp
      |-WEB-INF
|-pom.xml
|-target/
```

for eg..
we can build the functionalities related to raw material management and procurement by writing the code/components pertaining to it under the package hierarchy as “com.automobilemgmtsuite.rawmaterialproc.*

```
                                              |-controller
                                              |-dao
                                              |-service
                                              …
```

There are problems with this approach
1. The complexity in understanding the whole system is very high, since all the modules/functionalities are built into one big project
2. not all the functionalities should be made accessible publicly. there are few areas/functionalities we wanted to host in-house within the organization like raw material management, inventory, production tracking and few other modules should be made accessible publicly like sales/marketing and distributors network
if we have built all of these modules/functionalities into one single project, we cannot host them separately.
#2.
To build and manage the enterprise application with lot of functionalities we can break down the entire system into sub-modules. The entire application/system is being built out of single sourcecode repository only, but broken down each module into sub-projects as below.


raw material management/procurement
inventory mgmt
production tracking
distributor network
sales & marketing
automobile-core [common-library]
individual web-projects built into war
jar project
ear-project

```
automobilemgmtsuite.ear
|-lib
  |-automobile-core.jar
    [shared library]
|-rawmaterialproc.war
|-inventorymgmt.war
|-productiontracking.war
|-distributorntwk.war
|-salesmarketing.war
```

Instead of building the project out of a singlesource, we can break down the entire system into individual projects called “modules” which are pacakaged as either war/jars based on their nature. Across these modules there are common functionality between them, so each module can express the other as library dependency so that they can reuse the components of other modules, this means a component of one module can talk to the component of another module using their interfaces through classpath.

If there is a huge functionality that seems to be common across all the modules, then we can built it into a shared common library that can be reused across the modules.

All of these modules are assembled together into one single ear and will be deployed onto the target environment.
For developing such projects we need to build them use maven multi-module project type.