InternalCoreModule

Main module is created in **InternalCoreModuleFactory** (packages/core/injector/internal-core-module/internal-core-module-factory.ts)

It’s main role is to initialize all main providers that will be required to perform core functionalities

# ExternalContextCreator

Once this module is created it creates contextCreator and consumer for all enhancers

Example:

* PipesContextCreator -> This class helps in getting all pipes from current instance and callback and from global context, which later will be applied to current callback  
  Also this method helps in validating and getting pipes instances
* GuardConsumer-> This class applies calls guards sequence and share an context over each of guards

Also **ExternalContextCreator** has method **create** which role is to apply all enhancers to current callback method, it follows next steps:

1. Get parent module instance (Controller instance)
2. Get controller method metadata
3. Get enhancers instances that should be applied to current method
4. Get method params
5. Make new functions that will apply pipes and guards
6. Create instance handler that will call controller method and will apply params which were taken from pipes
7. Make final target method which applies guards interceptors and calls our controller method
8. Final step is to apply error filters

ModulesContainer

This storage (Map) of all modules. It stores module by token

# HttpAdapterHost

This is express or fastify http adapter

Nest initialize concrete instance of http adapter and abstraction by adapter name

# LazyModuleLoader

This module offer ability to load module only when needed (in if block)

**const moduleRef = await this.lazyModuleLoader.load(() => LazyModule);**

SerializedGraph

SerializedGraph in this case is graph that describes how entire structure works and components interact