분산프로그래밍 2 발표

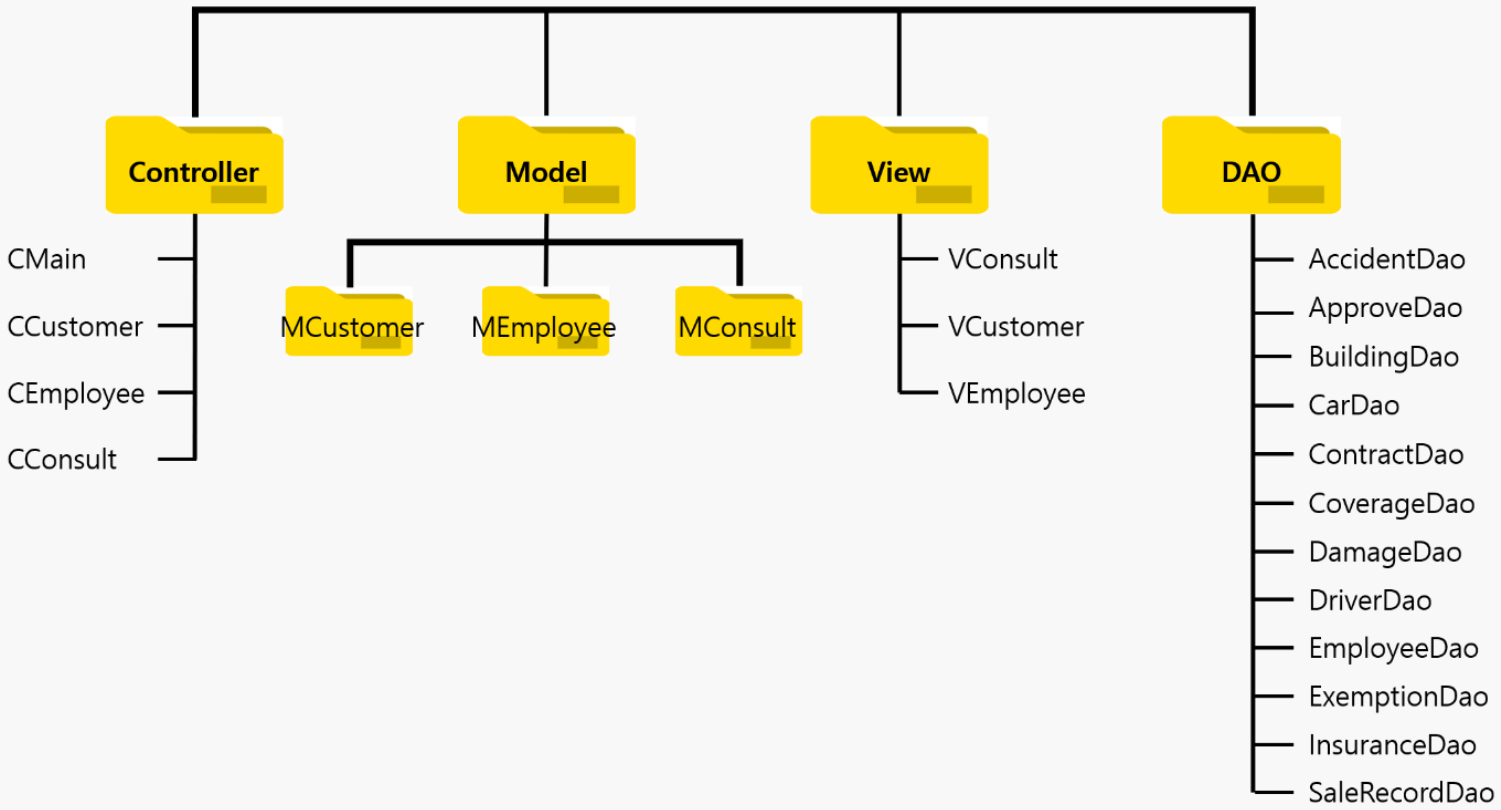
큘비팀
김규리 이은비

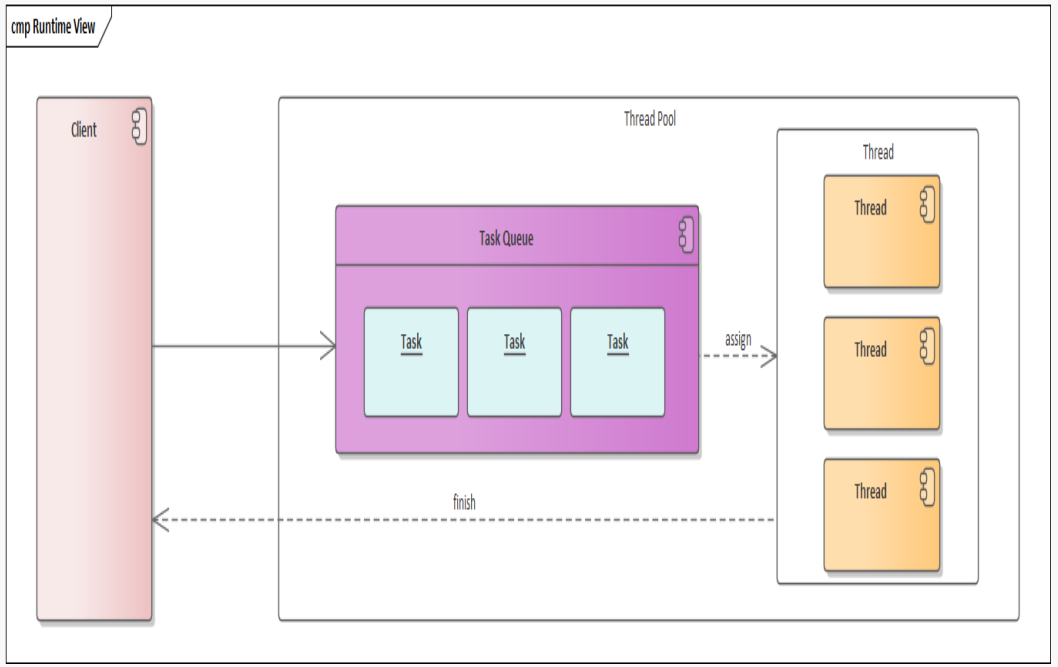
INDEX

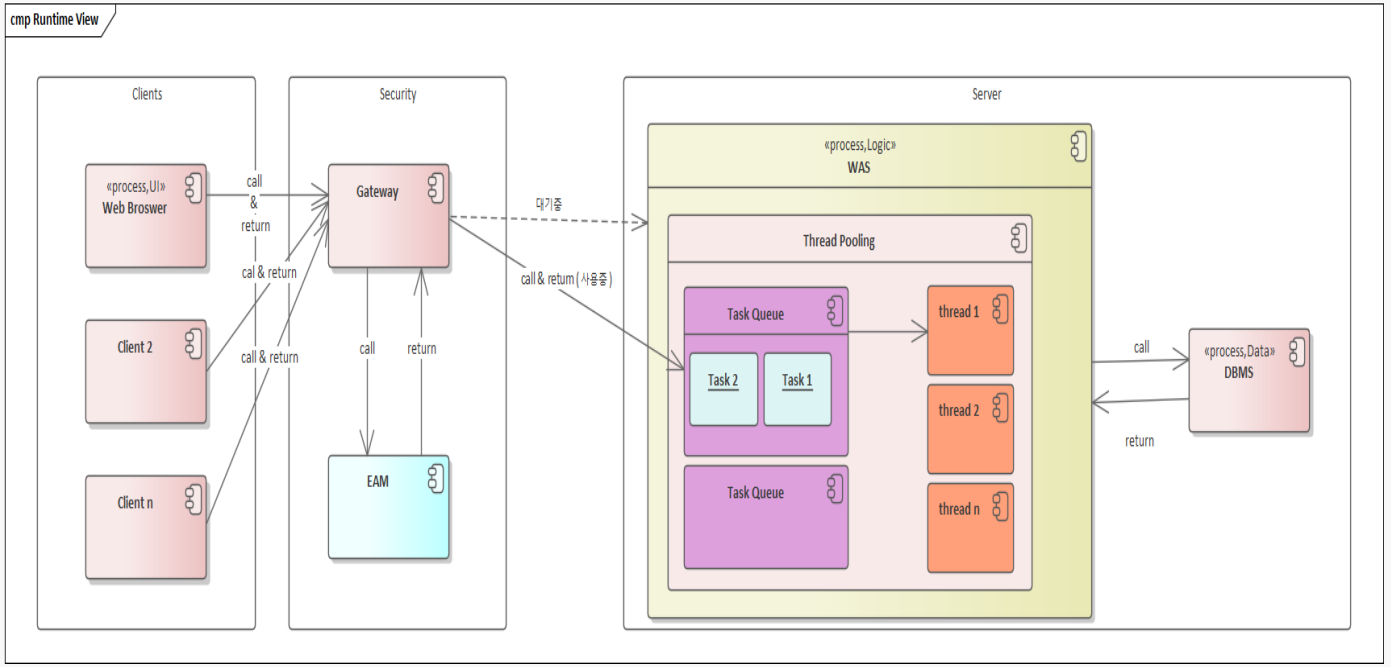
아키텍처

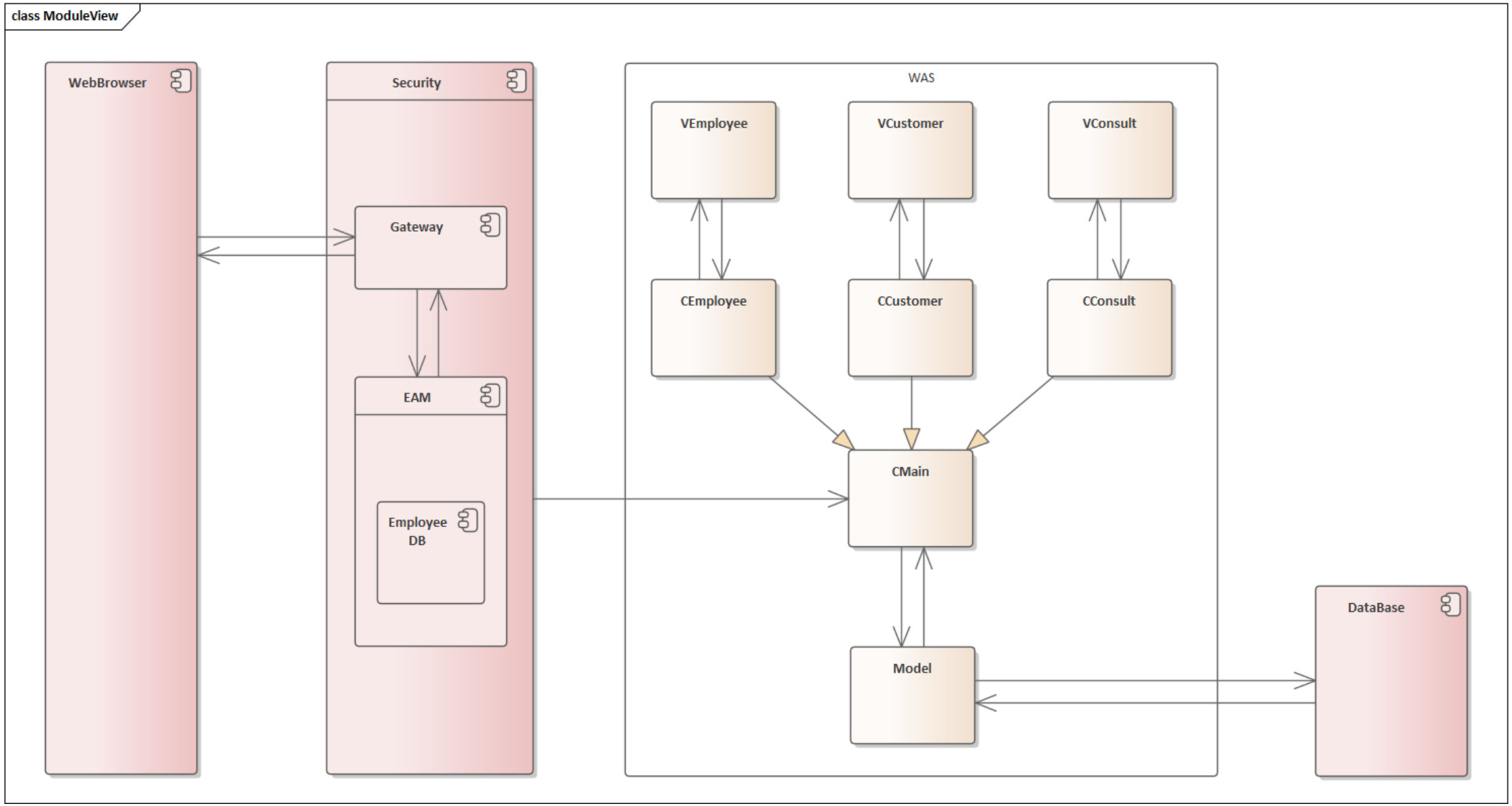
코드 리뷰

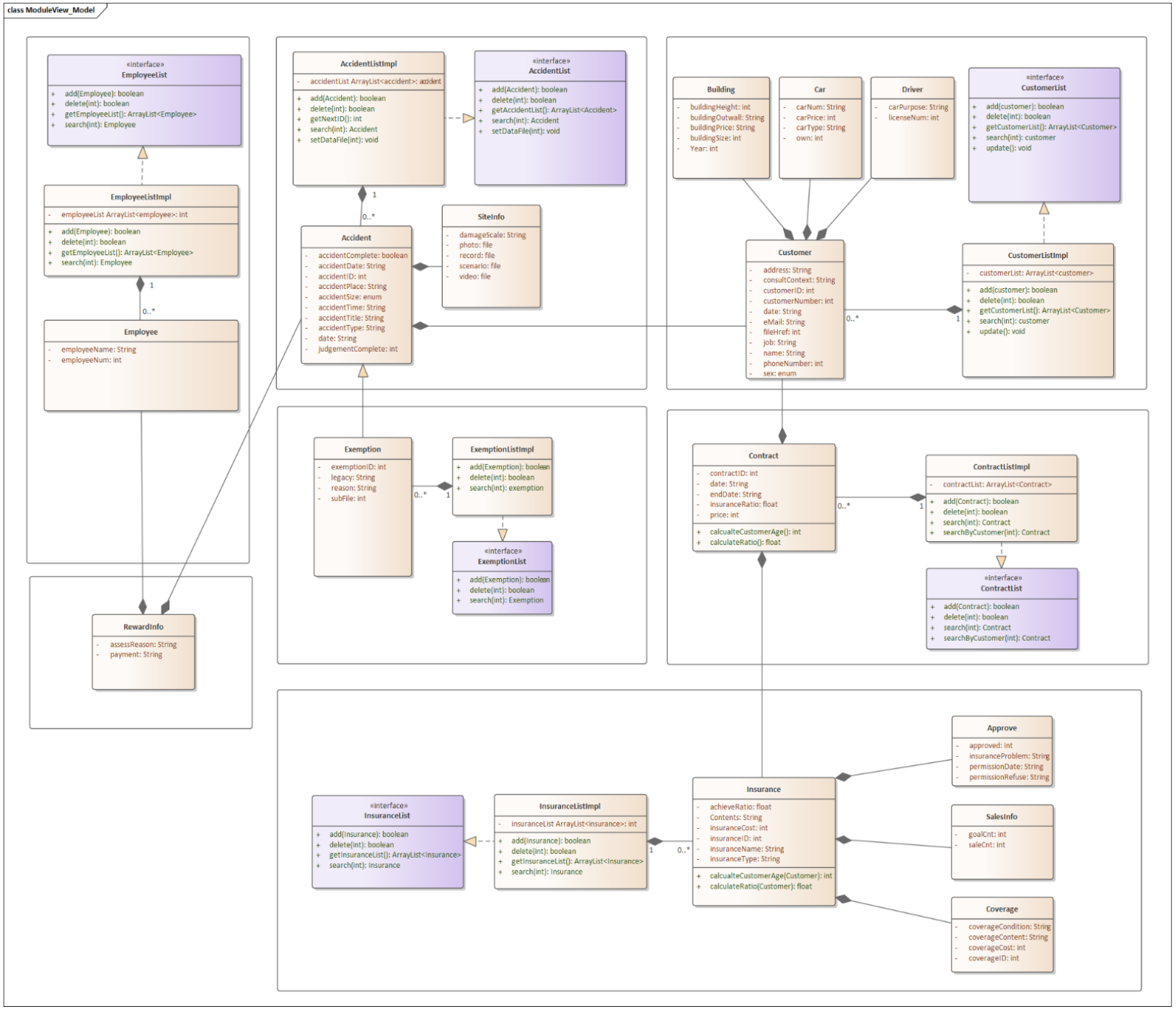
아키텍처_중간 레포트


Controller
CMain
CCustomer
CEmployee
CConsult
Model
MCustomer
MEmployee
MConsult
View
VConsult
VCustomer
VEmployee
DAO
AccidentDao
ApproveDao
BuildingDao
CarDao
ContractDao
CoverageDao
DamageDao
DriverDao
EmployeeDao
ExemptionDao
InsuranceDao
SaleRecordDao

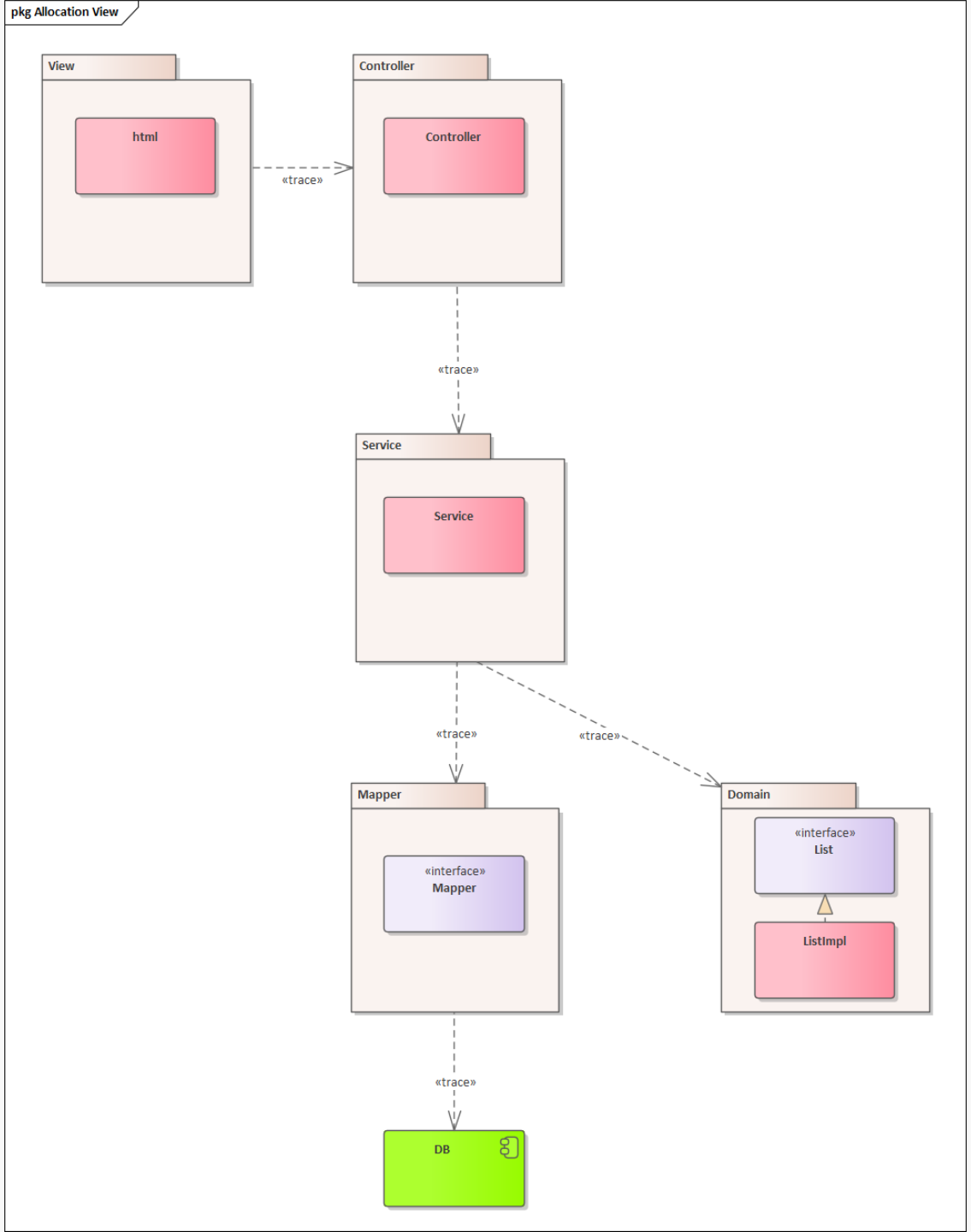

cmp Runtime View
Client
Thread Pool
Task Queue
Task
Task
Task
assign
Thread
Thread
Thread
Thread
finish

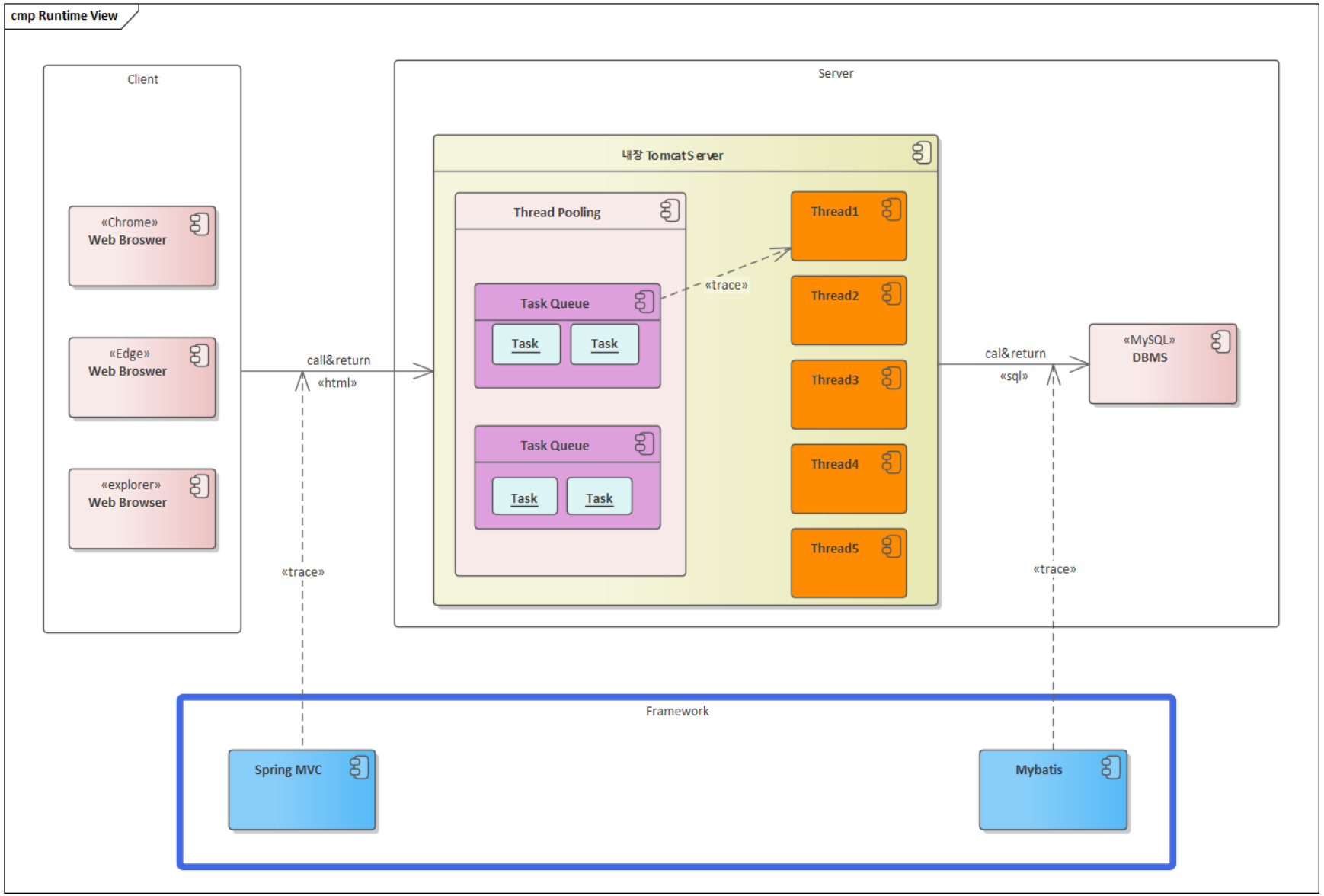

cmp Runtime View
Clients
«process,UI»
Web Broswer
Client 2
Client n
call
&
return
cal & return
cal & return
Security
Gateway
call
return
EAM
대기열
call & return (사용량)
Server
«process,Logic»
WAS
Thread Pooling
Task Queue
Task 2
Task 1
Task Queue
thread 1
thread 2
thread n
call
return
«process,Data»
DBMS

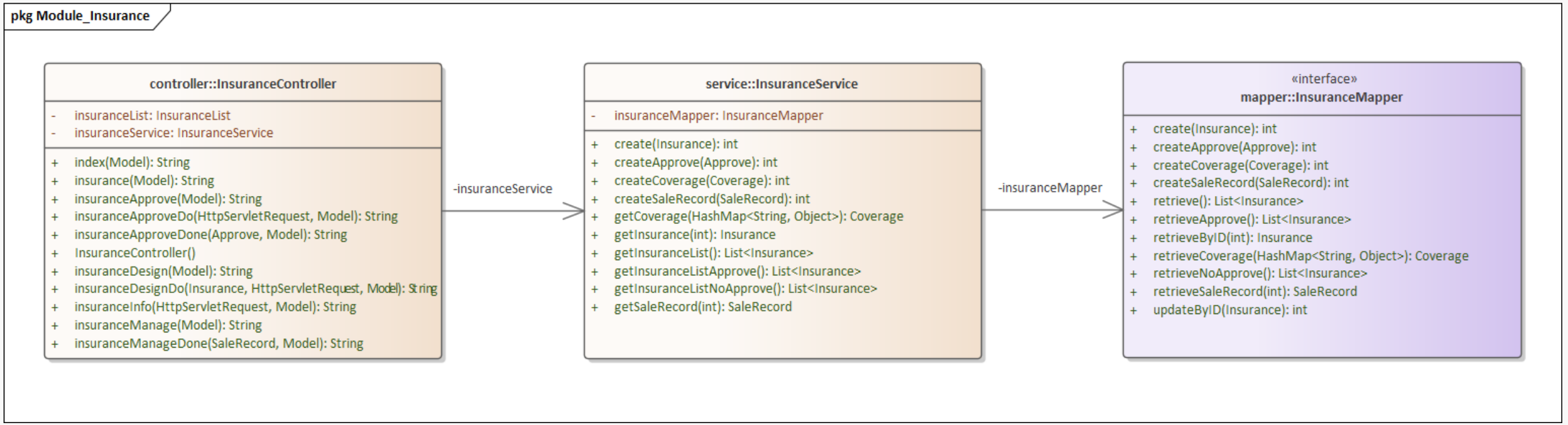

class ModuleView
WebBrowser
Security
Gateway
EAM
Employee
DB
WAS
VEmployee
VCustomer
VConsult
CEmployee
CCustomer
CConsult
CMain
Model
DataBase

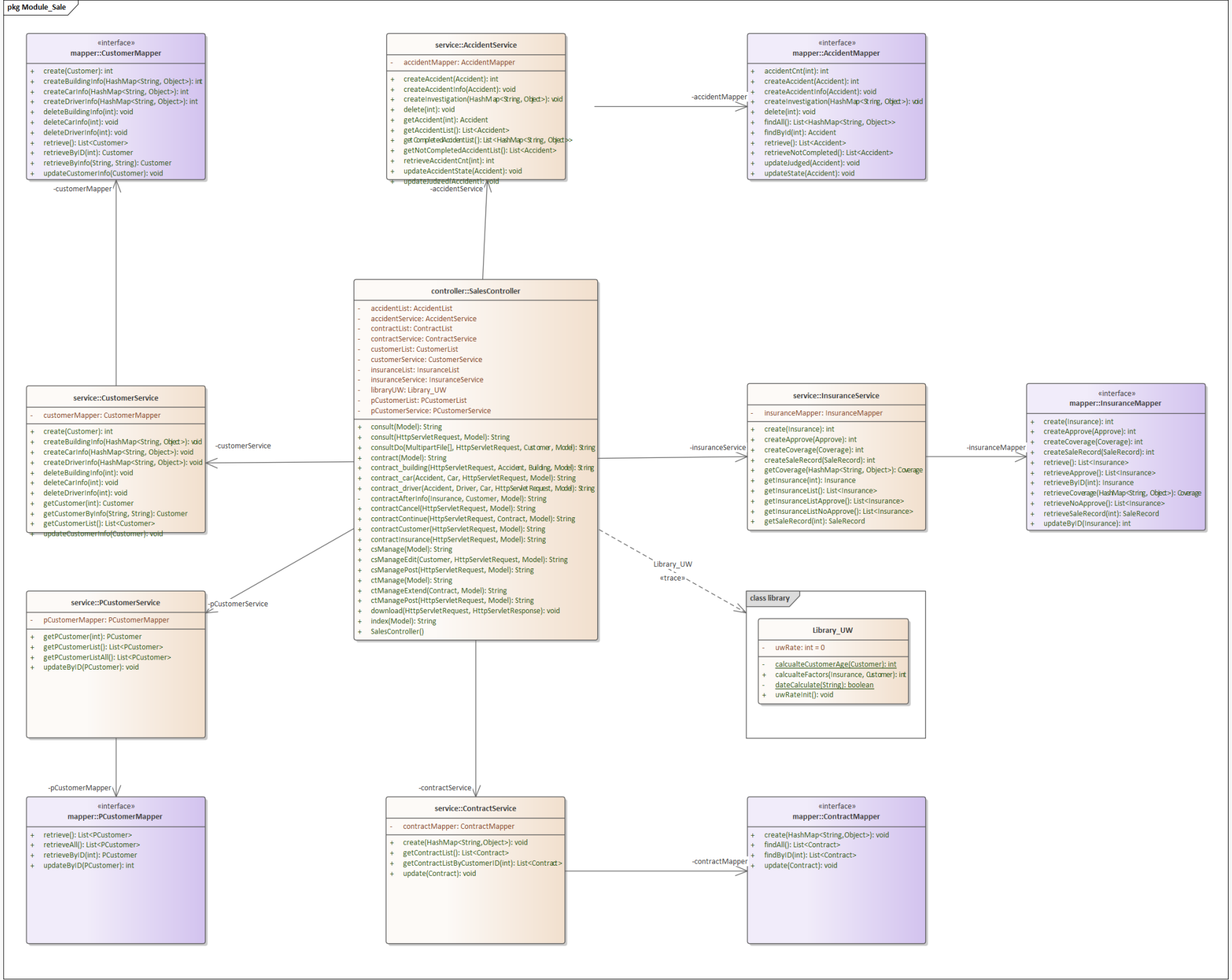

class ModuleView_Model
«interface»
EmployeeList
+ add(Employee): boolean
+ delete(int): boolean
+ getEmployeeList(): ArrayList<Employee>
+ search(int): Employee
EmployeeListImpl
- employeeList ArrayList<employee>: int
+ add(Employee): boolean
+ delete(int): boolean
+ getEmployeeList(): ArrayList<Employee>
+ search(int): Employee
1
0..*
Employee
- employeeName: String
- employeeNum: int
RewardInfo
- assessReason: String
- payment: String
AccidentListImpl
- accidentList ArrayList<accident>: accident
+ add(Accident): boolean
+ delete(int): boolean
+ getNextID(): int
+ search(int): Accident
+ setDataFile(int): void
«interface»
AccidentList
+ add(Accident): boolean
+ delete(int): boolean
+ getAccidentList(): ArrayList<Accident>
+ search(int): Accident
+ setDataFile(int): void
1
0..*
Accident
- accidentComplete: boolean
- accidentDate: String
- accidentID: int
- accidentPlace: String
- accidentSize: enum
- accidentTime: String
- accidentTitle: String
- accidentType: String
- date: String
- judgementComplete: int
SiteInfo
- damageScale: String
- photo: file
- record: file
- scenario: file
- video: file
Building
- buildingHeight: int
- buildingOutwall: String
- buildingPrice: String
- buildingSize: int
- Year: int
Car
- carNum: String
- carPrice: int
- carType: String
- own: int
Driver
- carPurpose: String
- licenseNum: int
«interface»
CustomerList
+ add(customer): boolean
+ delete(int): boolean
+ getCustomerList(): ArrayList<Customer>
+ search(int): customer
+ update(): void
Customer
- address: String
- consultContext: String
- customerID: int
- customerNumber: int
- date: String
- eMail: String
- fileHref: int
- job: String
- name: String
- phoneNumber: int
- sex: enum
0..*
1
CustomerListImpl
- customerList: ArrayList<customer>
+ add(customer): boolean
+ delete(int): boolean
+ getCustomerList(): ArrayList<Customer>
+ search(int): customer
+ update(): void
Exemption
- exemptionID: int
- legacy: String
- reason: String
- subFile: int
0..*
1
ExemptionListImpl
+ add(Exemption): boolean
+ delete(int): boolean
+ search(int): exemption
«interface»
ExemptionList
+ add(Exemption): boolean
+ delete(int): boolean
+ search(int): Exemption
Contract
- contractID: int
- date: String
- endDate: String
- insuranceRatio: float
- price: int
+ calcualteCustomerAge(): int
+ calculateRatio(): float
0..*
1
ContractListImpl
- contractList: ArrayList<Contract>
+ add(Contract): boolean
+ delete(int): boolean
+ search(int): Contract
+ searchByCustomer(int): Contract
«interface»
ContractList
+ add(Contract): boolean
+ delete(int): boolean
+ search(int): Contract
+ searchByCustomer(int): Contract
«interface»
InsuranceList
+ add(Insurance): boolean
+ delete(int): boolean
+ getInsuranceList(): ArrayList<Insurance>
+ search(int): Insurance
InsuranceListImpl
- insuranceList ArrayList<insurance>: int
+ add(Insurance): boolean
+ delete(int): boolean
+ getInsuranceList(): ArrayList<Insurance>
+ search(int): Insurance
1
0..*
Insurance
- achieveRatio: float
- Contents: String
- insuranceCost: int
- insuranceID: int
- insuranceName: String
- insuranceType: String
+ calcualteCustomerAge(Customer): int
+ calculateRatio(Customer): float
Approve
- approved: int
- insuranceProblem: String
- permissionDate: String
- permissionRefuse: String
SalesInfo
- goalCnt: int
- saleCnt: int
Coverage
- coverageCondition: String
- coverageContent: String
- coverageCost: int
- coverageID: int

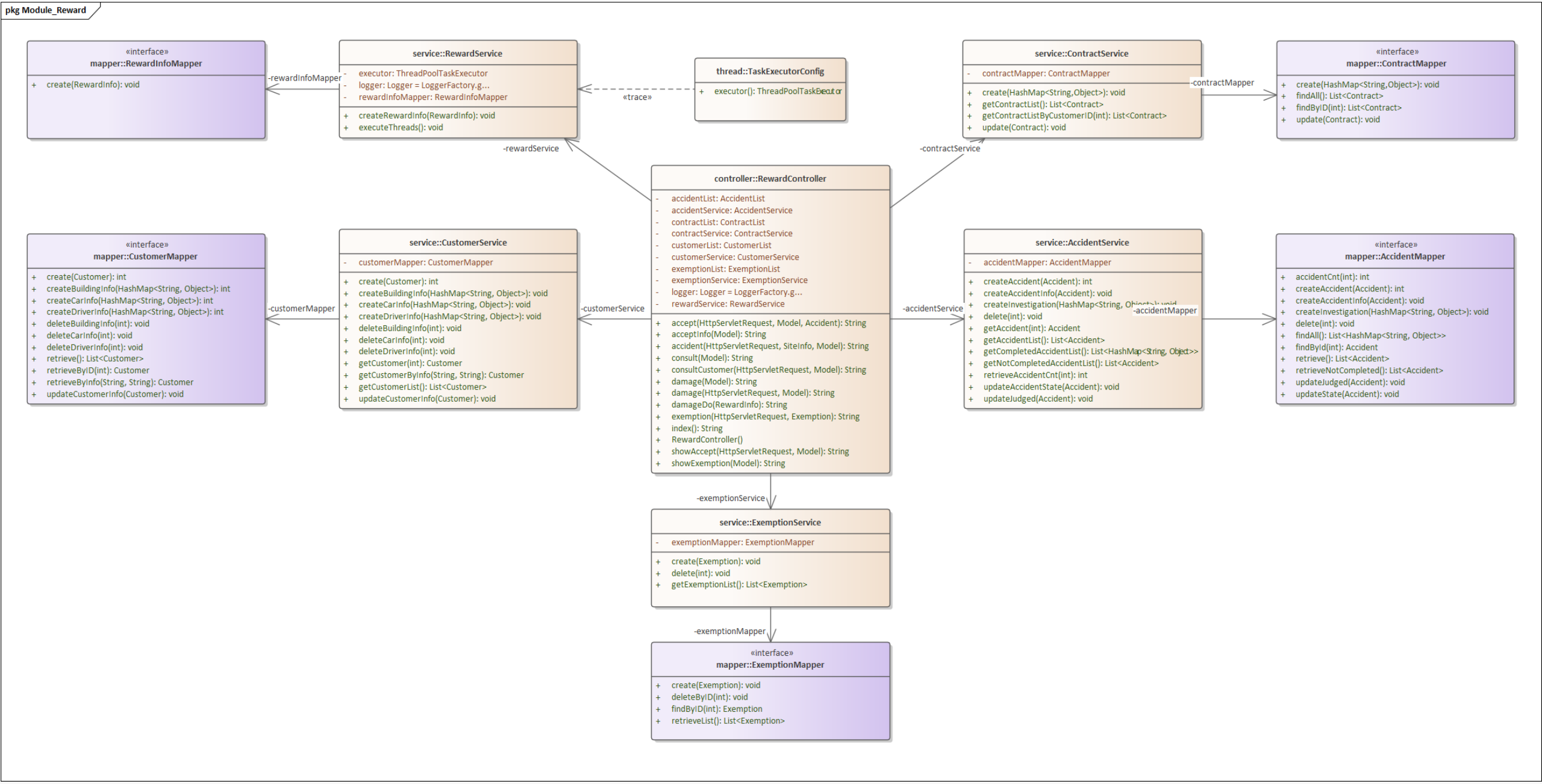
아키텍처_구현

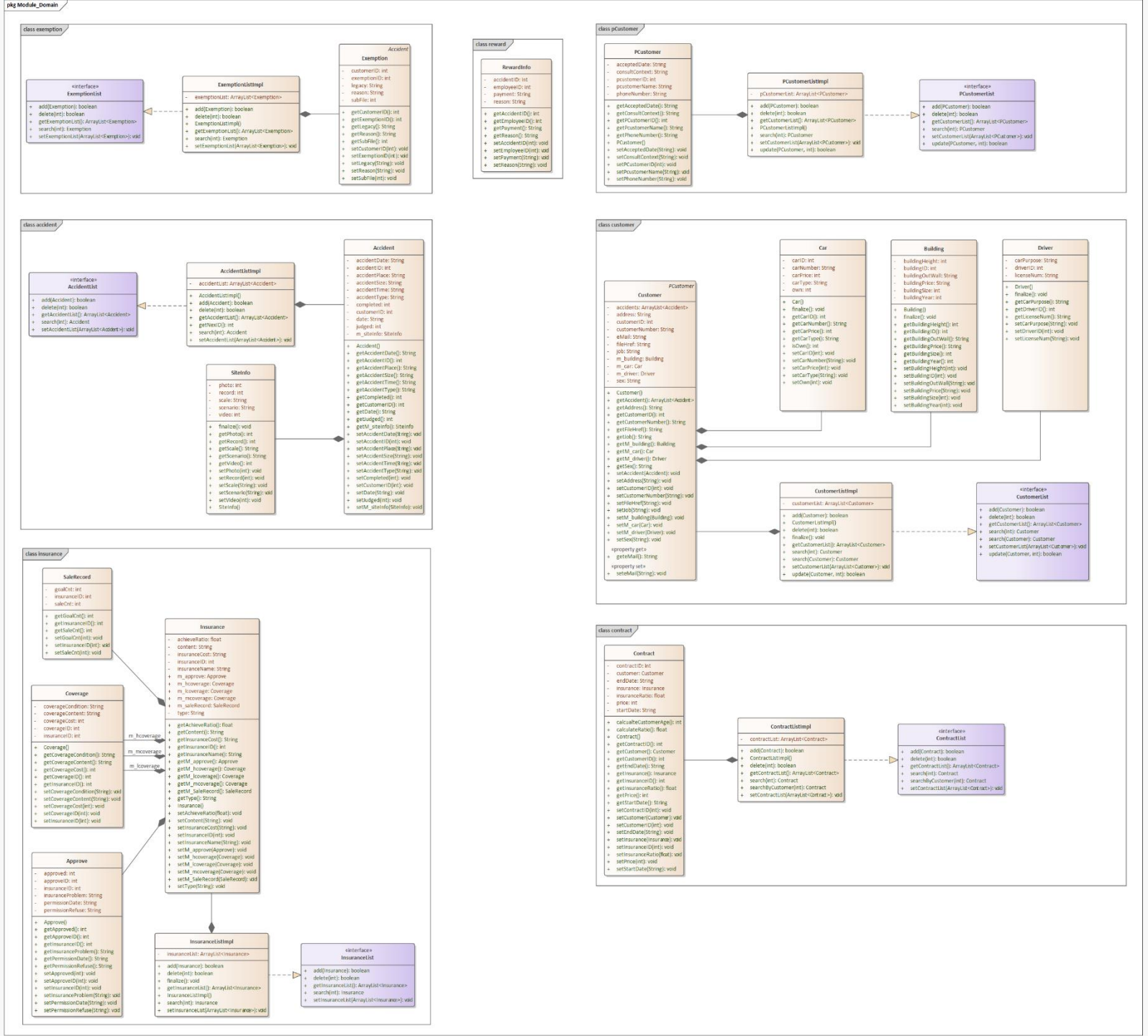

pkg Allocation View
View
html
«trace»
Controller
Controller
«trace»
Service
Service
«trace»
«trace»
Mapper
«interface»
Mapper
Domain
«interface»
List
ListImpl
«trace»
DB


cmp Runtime View
Client
«Chrome»
Web Broswer
«Edge»
Web Broswer
«explorer»
Web Browser
call&return
«html»
«trace»
Server
내장 TomcatServer
Thread Pooling
Task Queue
Task
Task
Task Queue
Task
Task
«trace»
Thread1
Thread2
Thread3
Thread4
Thread5
cal&return
«sql»
«MySQL»
DBMS
«trace»
Framework
Spring MVC
Mybatis


pkg Module_Insurance
controller::InsuranceController
- insuranceList: InsuranceList
- insuranceService: InsuranceService
+ index(Model): String
+ insurance(Model): String
+ insuranceApprove(Model): String
+ insuranceApproveDo(HttpServletRequest, Model): String
+ insuranceApproveDone(Approve, Model): String
+ InsuranceController()
+ insuranceDesign(Model): String
+ insuranceDesignDo(Insurance, HttpServletRequest, Model): String
+ insuranceInfo(HttpServletRequest, Model): String
+ insuranceManage(Model): String
+ insuranceManageDone(SaleRecord, Model): String
-insuranceService
service::InsuranceService
- insuranceMapper: InsuranceMapper
+ create(Insurance): int
+ createApprove(Approve): int
+ createCoverage(Coverage): int
+ createSaleRecord(SaleRecord): int
+ getCoverage(HashMap<String, Object>): Coverage
+ getInsurance(int): Insurance
+ getInsuranceList(): List<Insurance>
+ getInsuranceListApprove(): List<Insurance>
+ getInsuranceListNoApprove(): List<Insurance>
+ getSaleRecord(int): SaleRecord
-insuranceMapper
«interface»
mapper::InsuranceMapper
+ create(Insurance): int
+ createApprove(Approve): int
+ createCoverage(Coverage): int
+ createSaleRecord(SaleRecord): int
+ retrieve(): List<Insurance>
+ retrieveApprove(): List<Insurance>
+ retrieveByID(int): Insurance
+ retrieveCoverage(HashMap<String, Object>): Coverage
+ retrieveNoApprove(): List<Insurance>
+ retrieveSaleRecord(int): SaleRecord
+ updateByID(Insurance): int


pkg Module_Sale
«interface»
mapper::CustomerMapper
+ create(Customer): int
+ createBuildingInfo(HashMap<String, Object>): int
+ createCarInfo(HashMap<String, Object>): int
+ createDriverInfo(HashMap<String, Object>): int
+ deleteBuildingInfo(int): void
+ deleteCarInfo(int): void
+ deleteDriverInfo(int): void
+ retrieve(): List<Customer>
+ retrieveByID(int): Customer
+ retrieveByInfo(String, String): Customer
+ updateCustomerInfo(Customer): void
-customerMapper
service::AccidentService
- accidentMapper: AccidentMapper
+ createAccident(Accident): int
+ createAccidentInfo(Accident): void
+ createInvestigation(HashMap<String, Object>): void
+ delete(int): void
+ getAccident(int): Accident
+ getAccidentList(): List<Accident>
+ getCompletedAccidentList(): List<HashMap<String, Object>>
+ getNotCompletedAccidentList(): List<Accident>
+ retrieveAccidentCnt(int): int
+ updateAccidentState(Accident): void
+ updateJudged(Accident): void
-accidentService
-accidentMapper
«interface»
mapper::AccidentMapper
+ accidentCnt(int): int
+ createAccident(Accident): int
+ createAccidentInfo(Accident): void
+ createInvestigation(HashMap<String, Object>): void
+ delete(int): void
+ findAll(): List<HashMap<String, Object>>
+ findById(int): Accident
+ retrieve(): List<Accident>
+ retrieveNotCompleted(): List<Accident>
+ updateJudged(Accident): void
+ updateState(Accident): void
controller::SalesController
- accidentList: AccidentList
- accidentService: AccidentService
- contractList: ContractList
- contractService: ContractService
- customerList: CustomerList
- customerService: CustomerService
- insuranceList: InsuranceList
- insuranceService: InsuranceService
- libraryUW: Library_UW
- pCustomerList: PCustomerList
- pCustomerService: PCustomerService
+ consult(Model): String
+ consult(HttpServletRequest, Model): String
+ consultDo(MultipartFile[], HttpServletRequest, Customer, Model): String
+ contract(Model): String
+ contract_building(HttpServletRequest, Accident, Building, Model): String
+ contract_car(Accident, Car, HttpServletRequest, Model): String
+ contract_driver(Accident, Driver, Car, HttpServletRequest, Model): String
- contractAfterInfo(Insurance, Customer, Model): String
+ contractCancel(HttpServletRequest, Model): String
+ contractContinue(HttpServletRequest, Contract, Model): String
+ contractCustomer(HttpServletRequest, Model): String
+ contractInsurance(HttpServletRequest, Model): String
+ csManage(Model): String
+ csManageEdit(Customer, HttpServletRequest, Model): String
+ csManagePost(HttpServletRequest, Model): String
+ ctManage(Model): String
+ ctManageExtend(Contract, Model): String
+ ctManagePost(HttpServletRequest, Model): String
+ download(HttpServletRequest, HttpServletResponse): void
+ index(Model): String
+ SalesController()
service::CustomerService
- customerMapper: CustomerMapper
+ create(Customer): int
+ createBuildingInfo(HashMap<String, Object>): void
+ createCarInfo(HashMap<String, Object>): void
+ createDriverInfo(HashMap<String, Object>): void
+ deleteBuildingInfo(int): void
+ deleteCarInfo(int): void
+ deleteDriverInfo(int): void
+ getCustomer(int): Customer
+ getCustomerByInfo(String, String): Customer
+ getCustomerList(): List<Customer>
+ updateCustomerInfo(Customer): void
-customerService
service::InsuranceService
- insuranceMapper: InsuranceMapper
+ create(Insurance): int
+ createApprove(Approve): int
+ createCoverage(Coverage): int
+ createSaleRecord(SaleRecord): int
+ getCoverage(HashMap<String, Object>): Coverage
+ getInsurance(int): Insurance
+ getInsuranceList(): List<Insurance>
+ getInsuranceListApprove(): List<Insurance>
+ getInsuranceListNoApprove(): List<Insurance>
+ getSaleRecord(int): SaleRecord
-insuranceService
-insuranceMapper
«interface»
mapper::InsuranceMapper
+ create(Insurance): int
+ createApprove(Approve): int
+ createCoverage(Coverage): int
+ createSaleRecord(SaleRecord): int
+ retrieve(): List<Insurance>
+ retrieveApprove(): List<Insurance>
+ retrieveByID(int): Insurance
+ retrieveCoverage(HashMap<String, Object>): Coverage
+ retrieveNoApprove(): List<Insurance>
+ retrieveSaleRecord(int): SaleRecord
+ updateByID(Insurance): int
Library_UW
«trace»
class library
Library_UW
- uwRate: int = 0
- calcualteCustomerAge(Customer): int
+ calcualteFactors(Insurance, Customer): int
- dateCalculate(String): boolean
+ uwRateInit(): void
service::PCustomerService
- pCustomerMapper: PCustomerMapper
+ getPCustomer(int): PCustomer
+ getPCustomerList(): List<PCustomer>
+ getPCustomerListAll(): List<PCustomer>
+ updateByID(PCustomer): void
-pCustomerService
-pCustomerMapper
«interface»
mapper::PCustomerMapper
+ retrieve(): List<PCustomer>
+ retrieveAll(): List<PCustomer>
+ retrieveByID(int): PCustomer
+ updateByID(PCustomer): int
-contractService
service::ContractService
- contractMapper: ContractMapper
+ create(HashMap<String,Object>): void
+ getContractList(): List<Contract>
+ getContractListByCustomerID(int): List<Contract>
+ update(Contract): void
-contractMapper
«interface»
mapper::ContractMapper
+ create(HashMap<String,Object>): void
+ findAll(): List<Contract>
+ findByID(int): List<Contract>
+ update(Contract): void


pkg Module_Reward
«interface»
mapper::RewardInfoMapper
+ create(RewardInfo): void
-rewardInfoMapper
service::RewardService
- executor: ThreadPoolTaskExecutor
- logger: Logger = LoggerFactory.g...
- rewardInfoMapper: RewardInfoMapper
+ createRewardInfo(RewardInfo): void
+ executeThreads(): void
«trace»
thread::TaskExecutorConfig
+ executor(): ThreadPoolTaskExecutor
service::ContractService
- contractMapper: ContractMapper
+ create(HashMap<String,Object>): void
+ getContractList(): List<Contract>
+ getContractListByCustomerID(int): List<Contract>
+ update(Contract): void
-contractMapper
«interface»
mapper::ContractMapper
+ create(HashMap<String,Object>): void
+ findAll(): List<Contract>
+ findByID(int): List<Contract>
+ update(Contract): void
-rewardService
-contractService
controller::RewardController
- accidentList: AccidentList
- accidentService: AccidentService
- contractList: ContractList
- contractService: ContractService
- customerList: CustomerList
- customerService: CustomerService
- exemptionList: ExemptionList
- exemptionService: ExemptionService
- logger: Logger = LoggerFactory.g...
- rewardService: RewardService
+ accept(HttpServletRequest, Model, Accident): String
+ acceptInfo(Model): String
+ accident(HttpServletRequest, SiteInfo, Model): String
+ consult(Model): String
+ consultCustomer(HttpServletRequest, Model): String
+ damage(Model): String
+ damage(HttpServletRequest, Model): String
+ damageDo(RewardInfo): String
+ exemption(HttpServletRequest, Exemption): String
+ index(): String
+ RewardController()
+ showAccept(HttpServletRequest, Model): String
+ showExemption(Model): String
«interface»
mapper::CustomerMapper
+ create(Customer): int
+ createBuildingInfo(HashMap<String, Object>): int
+ createCarInfo(HashMap<String, Object>): int
+ createDriverInfo(HashMap<String, Object>): int
+ deleteBuildingInfo(int): void
+ deleteCarInfo(int): void
+ deleteDriverInfo(int): void
+ retrieve(): List<Customer>
+ retrieveByID(int): Customer
+ retrieveByInfo(String, String): Customer
+ updateCustomerInfo(Customer): void
-customerMapper
service::CustomerService
- customerMapper: CustomerMapper
+ create(Customer): int
+ createBuildingInfo(HashMap<String, Object>): void
+ createCarInfo(HashMap<String, Object>): void
+ createDriverInfo(HashMap<String, Object>): void
+ deleteBuildingInfo(int): void
+ deleteCarInfo(int): void
+ deleteDriverInfo(int): void
+ getCustomer(int): Customer
+ getCustomerByInfo(String, String): Customer
+ getCustomerList(): List<Customer>
+ updateCustomerInfo(Customer): void
-customerService
-accidentService
service::AccidentService
- accidentMapper: AccidentMapper
+ createAccident(Accident): int
+ createAccidentInfo(Accident): void
+ createInvestigation(HashMap<String, Object>): void
+ delete(int): void
+ getAccident(int): Accident
+ getAccidentList(): List<Accident>
+ getCompletedAccidentList(): List<HashMap<String, Object>>
+ getNotCompletedAccidentList(): List<Accident>
+ retrieveAccidentCnt(int): int
+ updateAccidentState(Accident): void
+ updateJudged(Accident): void
-accidentMapper
«interface»
mapper::AccidentMapper
+ accidentCnt(int): int
+ createAccident(Accident): int
+ createAccidentInfo(Accident): void
+ createInvestigation(HashMap<String, Object>): void
+ delete(int): void
+ findAll(): List<HashMap<String, Object>>
+ findById(int): Accident
+ retrieve(): List<Accident>
+ retrieveNotCompleted(): List<Accident>
+ updateJudged(Accident): void
+ updateState(Accident): void
-exemptionService
service::ExemptionService
- exemptionMapper: ExemptionMapper
+ create(Exemption): void
+ delete(int): void
+ getExemptionList(): List<Exemption>
-exemptionMapper
«interface»
mapper::ExemptionMapper
+ create(Exemption): void
+ deleteByID(int): void
+ findByID(int): Exemption
+ retrieveList(): List<Exemption>


pkg Module_Domain
class exemption
«interface»
ExemptionList
+ add(Exemption): boolean
+ delete(int): boolean
+ getExemptionList(): ArrayList<Exemption>
+ search(int): Exemption
+ setExemptionList(ArrayList<Exemption>): void
ExemptionListImpl
- exemptionList: ArrayList<Exemption>
+ add(Exemption): boolean
+ delete(int): boolean
+ ExemptionListImpl()
+ getExemptionList(): ArrayList<Exemption>
+ search(int): Exemption
+ setExemptionList(ArrayList<Exemption>): void
Accident
Exemption
- customerID: int
- exemptionID: int
- legacy: String
- reason: String
- subFile: int
+ getCustomerID(): int
+ getExemptionID(): int
+ getLegacy(): String
+ getReason(): String
+ getSubFile(): int
+ setCustomerID(int): void
+ setExemptionID(int): void
+ setLegacy(String): void
+ setReason(String): void
+ setSubFile(int): void
class reward
RewardInfo
- accidentID: int
- employeeID: int
- payment: String
- reason: String
+ getAccidentID(): int
+ getEmployeeID(): int
+ getPayment(): String
+ getReason(): String
+ setAccidentID(int): void
+ setEmployeeID(int): void
+ setPayment(String): void
+ setReason(String): void
class pCustomer
PCustomer
- acceptedDate: String
- consultContext: String
- pcustomerID: int
- pcustomerName: String
- phoneNumber: String
+ getAcceptedDate(): String
+ getConsultContext(): String
+ getPCustomerID(): int
+ getPcustomerName(): String
+ getPhoneNumber(): String
+ PCustomer()
+ setAcceptedDate(String): void
+ setConsultContext(String): void
+ setPCustomerID(int): void
+ setPcustomerName(String): void
+ setPhoneNumber(String): void
PCustomerListImpl
- pCustomerList: ArrayList<PCustomer>
+ add(PCustomer): boolean
+ delete(int): boolean
+ getCustomerList(): ArrayList<PCustomer>
+ PCustomerListImpl()
+ search(int): PCustomer
+ setCustomerList(ArrayList<PCustomer>): void
+ update(PCustomer, int): boolean
«interface»
PCustomerList
+ add(PCustomer): boolean
+ delete(int): boolean
+ getCustomerList(): ArrayList<PCustomer>
+ search(int): PCustomer
+ setCustomerList(ArrayList<PCustomer>): void
+ update(PCustomer, int): boolean
class accident
«interface»
AccidentList
+ add(Accident): boolean
+ delete(int): boolean
+ getAccidentList(): ArrayList<Accident>
+ search(int): Accident
+ setAccidentList(ArrayList<Accident>): void
AccidentListImpl
- accidentList: ArrayList<Accident>
+ AccidentListImpl()
+ add(Accident): boolean
+ delete(int): boolean
+ getAccidentList(): ArrayList<Accident>
+ getNext(): int
+ search(int): Accident
+ setAccidentList(ArrayList<Accident>): void
SiteInfo
- photo: int
- record: int
- scale: String
- scenario: String
- video: int
+ finalize(): void
+ getPhoto(): int
+ getRecord(): int
+ getScale(): String
+ getScenario(): String
+ getVideo(): int
+ setPhoto(int): void
+ setRecord(int): void
+ setScale(String): void
+ setScenario(String): void
+ setVideo(int): void
+ SiteInfo()
Accident
- accidentDate: String
- accidentID: int
- accidentPlace: String
- accidentSize: String
- accidentTime: String
- accidentType: String
- completed: int
- customerID: int
- date: String
- judged: int
- m_siteInfo: SiteInfo
+ Accident()
+ getAccidentDate(): String
+ getAccidentID(): int
+ getAccidentPlace(): String
+ getAccidentSize(): String
+ getAccidentTime(): String
+ getAccidentType(): String
+ getCompleted(): int
+ getCustomerID(): int
+ getDate(): String
+ getJudged(): int
+ getM_siteInfo(): SiteInfo
+ setAccidentDate(String): void
+ setAccidentID(int): void
+ setAccidentPlace(String): void
+ setAccidentSize(String): void
+ setAccidentTime(String): void
+ setAccidentType(String): void
+ setCompleted(int): void
+ setCustomerID(int): void
+ setDate(String): void
+ setJudged(int): void
+ setM_siteInfo(SiteInfo): void
class customer
PCustomer
Customer
- accidents: ArrayList<Accident>
- address: String
- customerID: int
- customerNumber: String
- eMail: String
- fileHref: String
- job: String
- m_building: Building
- m_car: Car
- m_driver: Driver
- sex: String
+ Customer()
+ getAccident(): ArrayList<Accident>
+ getAddress(): String
+ getCustomerID(): int
+ getCustomerNumber(): String
+ getFileHref(): String
+ getJob(): String
+ getM_building(): Building
+ getM_car(): Car
+ getM_driver(): Driver
+ getSex(): String
+ setAccident(Accident): void
+ setAddress(String): void
+ setCustomerID(int): void
+ setCustomerNumber(String): void
+ setFileHref(String): void
+ setJob(String): void
+ setM_building(Building): void
+ setM_car(Car): void
+ setM_driver(Driver): void
+ setSex(String): void
«property get»
+ geteMail(): String
«property set»
+ seteMail(String): void
Car
- carID: int
- carNumber: String
- carPrice: int
- carType: String
- own: int
+ Car()
+ finalize(): void
+ getCarID(): int
+ getCarNumber(): String
+ getCarPrice(): int
+ getCarType(): String
+ isOwn(): int
+ setCarID(int): void
+ setCarNumber(String): void
+ setCarPrice(int): void
+ setCarType(String): void
+ setOwn(int): void
Building
- buildingHeight: int
- buildingID: int
- buildingOutWall: String
- buildingPrice: String
- buildingSize: int
- buildingYear: int
+ Building()
+ finalize(): void
+ getBuildingHeight(): int
+ getBuildingID(): int
+ getBuildingOutWall(): String
+ getBuildingPrice(): String
+ getBuildingSize(): int
+ getBuildingYear(): int
+ setBuildingHeight(int): void
+ setBuildingID(int): void
+ setBuildingOutWall(String): void
+ setBuildingPrice(String): void
+ setBuildingSize(int): void
+ setBuildingYear(int): void
Driver
- carPurpose: String
- driverID: int
- licenseNum: String
+ Driver()
+ finalize(): void
+ getCarPurpose(): String
+ getDriverID(): int
+ getLicenseNum(): String
+ setCarPurpose(String): void
+ setDriverID(int): void
+ setLicenseNum(String): void
CustomerListImpl
- customerList: ArrayList<Customer>
+ add(Customer): boolean
+ CustomerListImpl()
+ delete(int): boolean
+ finalize(): void
+ getCustomerList(): ArrayList<Customer>
+ search(int): Customer
+ search(Customer): Customer
+ setCustomerList(ArrayList<Customer>): void
+ update(Customer, int): boolean
«interface»
CustomerList
+ add(Customer): boolean
+ delete(int): boolean
+ getCustomerList(): ArrayList<Customer>
+ search(int): Customer
+ search(Customer): Customer
+ setCustomerList(ArrayList<Customer>): void
+ update(Customer, int): boolean
class insurance
SaleRecord
- goalCnt: int
- insuranceID: int
- saleCnt: int
+ getGoalCnt(): int
+ getInsuranceID(): int
+ getSaleCnt(): int
+ setGoalCnt(int): void
+ setInsuranceID(int): void
+ setSaleCnt(int): void
Coverage
- coverageCondition: String
- coverageContent: String
- coverageCost: int
- coverageID: int
- insuranceID: int
+ Coverage()
+ getCoverageCondition(): String
+ getCoverageContent(): String
+ getCoverageCost(): int
+ getCoverageID(): int
+ getInsuranceID(): int
+ setCoverageCondition(String): void
+ setCoverageContent(String): void
+ setCoverageCost(int): void
+ setCoverageID(int): void
+ setInsuranceID(int): void
m_hcoverage
m_mcoverage
m_lcoverage
Approve
- approved: int
- approveID: int
- insuranceID: int
- insuranceProblem: String
- permissionDate: String
- permissionRefuse: String
+ Approve()
+ getApproved(): int
+ getApproveID(): int
+ getInsuranceID(): int
+ getInsuranceProblem(): String
+ getPermissionDate(): String
+ getPermissionRefuse(): String
+ setApproved(int): void
+ setApproveID(int): void
+ setInsuranceID(int): void
+ setInsuranceProblem(String): void
+ setPermissionDate(String): void
+ setPermissionRefuse(String): void
Insurance
- achieveRatio: float
- content: String
- insuranceCost: String
- insuranceID: int
- insuranceName: String
+ m_approve: Approve
+ m_hcoverage: Coverage
+ m_lcoverage: Coverage
+ m_mcoverage: Coverage
+ m_saleRecord: SaleRecord
- type: String
+ getAchieveRatio(): float
+ getContent(): String
+ getInsuranceCost(): String
+ getInsuranceID(): int
+ getInsuranceName(): String
+ getM_approve(): Approve
+ getM_hcoverage(): Coverage
+ getM_lcoverage(): Coverage
+ getM_mcoverage(): Coverage
+ getM_SaleRecord(): SaleRecord
+ getType(): String
+ Insurance()
+ setAchieveRatio(float): void
+ setContent(String): void
+ setInsuranceCost(String): void
+ setInsuranceID(int): void
+ setInsuranceName(String): void
+ setM_approve(Approve): void
+ setM_hcoverage(Coverage): void
+ setM_lcoverage(Coverage): void
+ setM_mcoverage(Coverage): void
+ setM_SaleRecord(SaleRecord): void
+ setType(String): void
InsuranceListImpl
- insuranceList: ArrayList<insurance>
+ add(insurance): boolean
+ delete(int): boolean
+ finalize(): void
+ getInsuranceList(): ArrayList<Insurance>
+ InsuranceListImpl()
+ search(int): insurance
+ setInsuranceList(ArrayList<insurance>): void
«interface»
InsuranceList
+ add(insurance): boolean
+ delete(int): boolean
+ getInsuranceList(): ArrayList<Insurance>
+ search(int): insurance
+ setInsuranceList(ArrayList<insurance>): void
class contract
Contract
- contractID: int
- customer: Customer
- endDate: String
- insurance: Insurance
- insuranceRatio: float
- price: int
- startDate: String
+ calcualteCustomerAge(): int
+ calculateRatio(): float
+ Contract()
+ getContractID(): int
+ getCustomer(): Customer
+ getCustomerID(): int
+ getEndDate(): String
+ getInsurance(): Insurance
+ getInsuranceID(): int
+ getInsuranceRatio(): float
+ getPrice(): int
+ getStartDate(): String
+ setContractID(int): void
+ setCustomer(Customer): void
+ setCustomerID(int): void
+ setEndDate(String): void
+ setInsurance(Insurance): void
+ setInsuranceID(int): void
+ setInsuranceRatio(float): void
+ setPrice(int): void
+ setStartDate(String): void
ContractListImpl
- contractList: ArrayList<Contract>
+ add(Contract): boolean
+ ContractListImpl()
+ delete(int): boolean
+ getContractList(): ArrayList<Contract>
+ search(int): Contract
+ searchByCustomer(int): Contract
+ setContractList(ArrayList<Contract>): void
«interface»
ContractList
+ add(Contract): boolean
+ delete(int): boolean
+ getContractList(): ArrayList<Contract>
+ search(int): Contract
+ searchByCustomer(int): Contract
+ setContractList(ArrayList<Contract>): void

코드 리뷰

```java
// 상품 인가하기
@GetMapping("insurance/approve")
public String insuranceApprove( Model model ){
    this.insuranceList.setInsuranceList( (ArrayList<Insurance>) insuranceService.getInsuranceListNoApprove() );
    model.addAttribute( attributeName: "insuranceNoApprove", this.insuranceList.getInsuranceList() );
    return "insurance/approve";
}
@PostMapping("insurance/approve")
public String insuranceApproveDo( HttpServletRequest hsRequest, Model model ){
    Insurance insurance = this.insuranceList.search( Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ) );
    HashMap<String, Object> coverage = new HashMap<~>();
    coverage.put("insuranceID", Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ));
    coverage.put("coverageCondition", "high");

    insurance.setM_hcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "middle" );
    insurance.setM_mcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "low" );
    insurance.setM_lcoverage( insuranceService.getCoverage( coverage ) );

    model.addAttribute( attributeName: "insurance", insurance );
    return "insurance/approveDo";
}
@PostMapping("insurance/approveDone")
public String insuranceApproveDone(Approve approve, Model model ){
    insuranceService.createApprove( approve );

    this.insuranceList.search( approve.getInsuranceID() ).setM_approve( approve );

    model.addAttribute( attributeName: "msg", attributeValue: "상품 인가가 완료되었습니다." );
    return index( model );
}
```

코드 리뷰

```
// 상품 인가하기
@GetMapping("insurance/approve")
public String insuranceApprove( Model model ){
    this.insuranceList.setInsuranceList( (ArrayList<Insurance>) insuranceService.getInsuranceListNoApprove() );
    model.addAttribute( attributeName: "insuranceNoApprove", this.insuranceList.getInsuranceList() );
    return "insurance/approve";
}
@PostMapping("insurance/approve")
public String insuranceApproveDo( HttpServletRequest hsRequest, Model model ){
    Insurance insurance = this.insuranceList.search( Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ) );
    HashMap<String, Object> coverage = new HashMap<~>();
    coverage.put("insuranceID", Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ));
    coverage.put("coverageCondition", "high");

    insurance.setM_hcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "middle" );
    insurance.setM_mcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "low" );
    insurance.setM_lcoverage( insuranceService.getCoverage( coverage ) );

    model.addAttribute( attributeName: "insurance", insurance );
    return "insurance/approveDo";
}
@PostMapping("insurance/approveDone")
public String insuranceApproveDone(Approve approve, Model model ){
    insuranceService.createApprove( approve );

    this.insuranceList.search( approve.getInsuranceID() ).setM_approve( approve );

    model.addAttribute( attributeName: "msg", attributeValue: "상품 인가가 완료되었습니다." );
    return index( model );
}
```

코드 리뷰

```java
@Service
public class InsuranceService {
    @Autowired
    private InsuranceMapper insuranceMapper;

    public int create(Insurance insurance) { return insuranceMapper.create( insurance ); }
    public int createCoverage(Coverage coverage) { return insuranceMapper.createCoverage( coverage ); }
    public int createApprove(Approve approve) { return insuranceMapper.createApprove( approve ); }
    public List<Insurance> getInsuranceList(){ return insuranceMapper.retrieve(); }
    public List<Insurance> getInsuranceListNoApprove(){ return insuranceMapper.retrieveNoApprove(); }
    public List<Insurance> getInsuranceListApprove(){return insuranceMapper.retrieveApprove(); }
    public Insurance getInsurance(int insuranceID ){ return insuranceMapper.retrieveByID( insuranceID ); }
    public Coverage getCoverage( HashMap<String, Object> coverageInfo ){ return insuranceMapper.retrieveCoverage( coverageInfo ); }

    public int createSaleRecord(SaleRecord saleRecord) { return insuranceMapper.createSaleRecord(saleRecord); }
    public SaleRecord getSaleRecord( int insuranceID ) { return insuranceMapper.retrieveSaleRecord( insuranceID ); }
}
```

코드 리뷰

```java
@Mapper
public interface InsuranceMapper {
    List<Insurance> retrieve();
    List<Insurance> retrieveNoApprove();
    List<Insurance> retrieveApprove();
    int create( Insurance insurance );
    int createCoverage( Coverage coverage );
    int createApprove( Approve approve );
    int createSaleRecord(SaleRecord saleRecord);
    Insurance retrieveByID( int insuranceID );

    Coverage retrieveCoverage(HashMap<String, Object> coverageInfo);
    SaleRecord retrieveSaleRecord( int insuranceID );
```

코드 리뷰

```xml
<mapper namespace="com.insurance.Insurance_spring.mapper.InsuranceMapper">
    <insert id="create" parameterType="com.insurance.Insurance_spring.domain.insurance.Insurance" useGeneratedKeys="true" keyProperty="insuranceID">
        INSERT INTO insurance ( insuranceName, insuranceCost, content, type ) values ( #{insuranceName}, #{insuranceCost}, #{content}, #{type} )
    </insert>
    <insert id="createCoverage" parameterType="com.insurance.Insurance_spring.domain.insurance.Coverage" useGeneratedKeys="true" keyProperty="coverageID">
        INSERT INTO coverage ( insuranceID, coverageCondition, coverageContent, coverageCost ) values ( #{insuranceID}, #{coverageCondition}, #{coverageContent}, #{coverageCost} )
    </insert>
    <insert id="createApprove" parameterType="com.insurance.Insurance_spring.domain.insurance.Approve" useGeneratedKeys="true" keyProperty="approveID">
        INSERT INTO approve ( insuranceID, approved, permissionDate, permissionRefuse, insuranceProblem ) values ( #{insuranceID}, #{approved}, #{permissionDate}, #{permissionRefuse}, #{insuranceProblem} )
    </insert>
    <insert id="createSaleRecord" parameterType="com.insurance.Insurance_spring.domain.insurance.SaleRecord" useGeneratedKeys="true" keyProperty="insuranceID">
        INSERT INTO  salerecord ( insuranceID, goalCnt, saleCnt ) values (#{insuranceID}, #{goalCnt}, #{saleCnt} )
    </insert>

    <select id="retrieve" resultType="com.insurance.Insurance_spring.domain.insurance.Insurance">
        SELECT * FROM insurance
    </select>
    <select id="retrieveNoApprove" resultType="com.insurance.Insurance_spring.domain.insurance.Insurance">
        select * from insurance left outer join approve using(insuranceID) where approve.insuranceID IS NULL
    </select>
    <select id="retrieveByID" resultType="com.insurance.Insurance_spring.domain.insurance.Insurance">
        SELECT * FROM insurance WHERE insuranceID = #{insuranceID} LIMIT 1
    </select>
    <select id="retrieveCoverage" parameterType="HashMa" resultType="com.insurance.Insurance_spring.domain.insurance.Coverage">
        SELECT * FROM coverage WHERE insuranceID = #{insuranceID} AND coverageCondition = #{coverageCondition} LIMIT 1
    </select>
    <select id="retrieveApprove" resultType="com.insurance.Insurance_spring.domain.insurance.Insurance">
        select * from insurance left outer join approve using(insuranceID) where approve.approved = 1
    </select>
    <select id="retrieveSaleRecord" resultType="com.insurance.Insurance_spring.domain.insurance.SaleRecord">
        select * from salerecord where insuranceID=#{insuranceID} limit 1
    </select>

    <update id="updateByID" parameterType="com.insurance.Insurance_spring.domain.insurance.Insurance">
        update pcustomer set insuranceName=#{insuranceName}, insuranceCost=#{insuranceCost}, content=#{content} where insuranceID=#{insuranceID}
    </update>
</mapper>
```

코드 리뷰

```java
// 상품 인가하기
@GetMapping("insurance/approve")
public String insuranceApprove( Model model ){
    this.insuranceList.setInsuranceList( (ArrayList<Insurance>) insuranceService.getInsuranceListNoApprove() );
    model.addAttribute( attributeName: "insuranceNoApprove", this.insuranceList.getInsuranceList() );
    return "insurance/approve";
}
@PostMapping("insurance/approve")
public String insuranceApproveDo( HttpServletRequest hsRequest, Model model ){
    Insurance insurance = this.insuranceList.search( Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ) );
    HashMap<String, Object> coverage = new HashMap<~>();
    coverage.put("insuranceID", Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ));
    coverage.put("coverageCondition", "high");

    insurance.setM_hcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "middle" );
    insurance.setM_mcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "low" );
    insurance.setM_lcoverage( insuranceService.getCoverage( coverage ) );

    model.addAttribute( attributeName: "insurance", insurance );
    return "insurance/approveDo";
}
@PostMapping("insurance/approveDone")
public String insuranceApproveDone(Approve approve, Model model ){
    insuranceService.createApprove( approve );

    this.insuranceList.search( approve.getInsuranceID() ).setM_approve( approve );

    model.addAttribute( attributeName: "msg", attributeValue: "상품 인가가 완료되었습니다." );
    return index( model );
}
```

코드 리뷰

```
<body>
<div class="container">
  <button type="button" style="..." onclick="location.href='/'">
    <img style="..." src="http://localhost:8080/home4.png">
  </button>
  <form class="d-none" id="form_approve" action="/insurance/approve" method="post">
    <input type="hidden" name="insuranceID">
  </form>
  <div class="row px-2 mt-3">
    <h2>인가되지 않은 보험 목록</h2>
    <table class="table">
      <thead>
      <tr>
        <th>#</th>
        <th>보험이름</th>
        <th></th>
      </tr>
      </thead>
      <tbody>
      <tr th:each="insurance : ${insuranceNoApprove}">
        <td th:text="${insurance.insuranceID}"></td>
        <td th:text="${insurance.insuranceName}"></td>
        <td>
          <button class="btn btn-secondary" th:onclick="approve( [[${insurance.insuranceID}]] )">선택</button>
        </td>
      </tr>
      </tbody>
    </table>
  </div>
</div>

</body>
```

인가되지 않은 보험 목록

#	보험이름	
8	오늘만든건물화재	선택

코드 리뷰

```java
// 상품 인가하기
@GetMapping("insurance/approve")
public String insuranceApprove( Model model ){
    this.insuranceList.setInsuranceList( (ArrayList<Insurance>) insuranceService.getInsuranceListNoApprove() );
    model.addAttribute( attributeName: "insuranceNoApprove", this.insuranceList.getInsuranceList() );
    return "insurance/approve";
}
@PostMapping("insurance/approve")
public String insuranceApproveDo( HttpServletRequest hsRequest, Model model ){
    Insurance insurance = this.insuranceList.search( Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ) );
    HashMap<String, Object> coverage = new HashMap<~>();
    coverage.put("insuranceID", Integer.parseInt( hsRequest.getParameter( name: "insuranceID" ) ));
    coverage.put("coverageCondition", "high");

    insurance.setM_hcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "middle" );
    insurance.setM_mcoverage( insuranceService.getCoverage( coverage ) );
    coverage.put("coverageCondition", "low" );
    insurance.setM_lcoverage( insuranceService.getCoverage( coverage ) );

    model.addAttribute( attributeName: "insurance", insurance );
    return "insurance/approveDo";
}
@PostMapping("insurance/approveDone")
public String insuranceApproveDone(Approve approve, Model model ){
    insuranceService.createApprove( approve );

    this.insuranceList.search( approve.getInsuranceID() ).setM_approve( approve );

    model.addAttribute( attributeName: "msg", attributeValue: "상품 인가가 완료되었습니다." );
    return index( model );
}
```

코드 리뷰

```java
public class InsuranceListImpl implements InsuranceList {
    // attribute
    private ArrayList<Insurance> insuranceList;

    public InsuranceListImpl() { this.insuranceList = new ArrayList<Insurance>(); }
    // getter & setter
    public ArrayList<Insurance> getInsuranceList() {return insuranceList;}
    public void setInsuranceList(ArrayList<Insurance> insuranceList) {  this.insuranceList = insuranceList; }

    public void finalize() throws Throwable {

    }
    public boolean add(Insurance insurance){
        if(this.insuranceList.add(insurance)) return true;
        else return false;
    }
    public boolean delete(int insuranceId){
        for(Insurance insurance : this.insuranceList) {
            if(insurance.getInsuranceID() == insuranceId) {
                if(this.insuranceList.remove(insurance)) return true;
                break;
            }
        }
        return false;
    }
    public Insurance search(int insuranceId){
        for(Insurance insurance : this.insuranceList) {
            if(insurance.getInsuranceID() == insuranceId) return insurance;
        }
        return null;
    }
```

THANK YOU

감사합니다.