# **SW 기본 정보**

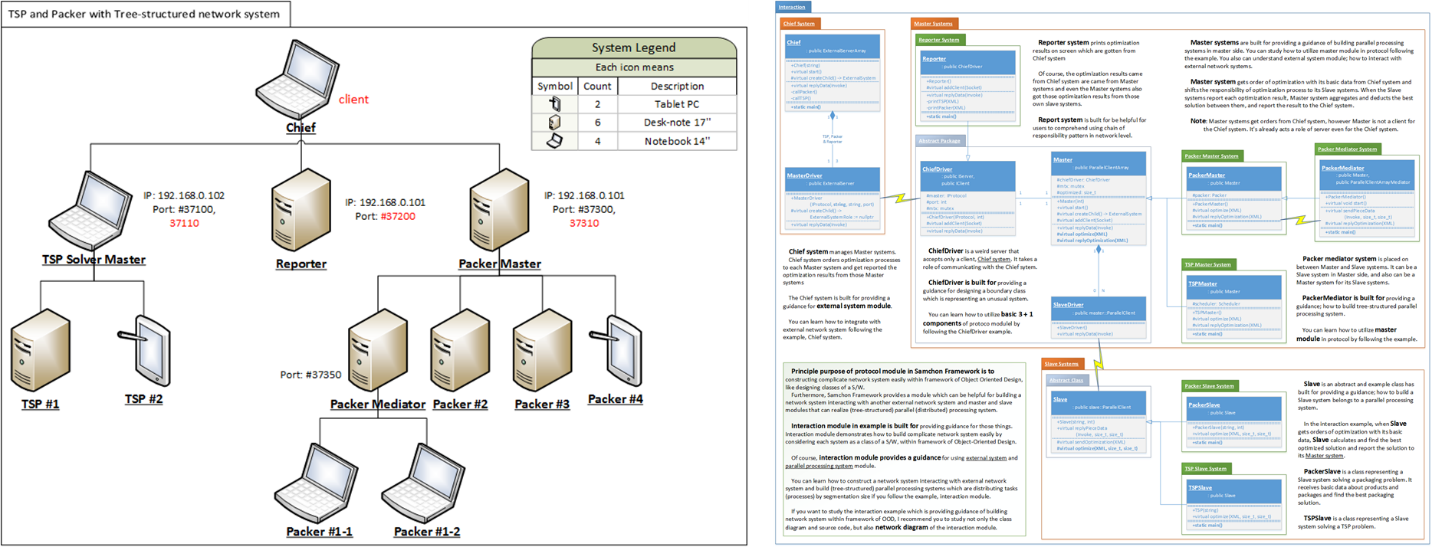
## **Samchon Framework**

Samchon Framework 는 네트워크 시스템을 S/W, 객체지향적 (Object-Oriented) 인 관점에서 구현할 수 있게 해 주는, 객체지향네트워크 (OON, Object Oriented Network) 프레임워크입니다.

주식 시뮬레이션과 시스템 트레이딩에 관한 클라우드 솔루션 (Samchon Simulation) 을 4년간 만들어 왔습니다. 클라우드 서버니 분산-병렬처리시스템이니 하는 것들을 설계하고 구현하면서, 모종의 필요성을 느껴 제작하게 된 것이 바로 이 Samchon Framework 입니다. 나름 완성도와 체계성에 자신이 있어, 이렇게 오픈소스로 공개하게 되었습니다.

다음은 삼촌 프레임워크의 활용 방안 및 목적입니다.

### 네트워크 시스템을 S/W, 객체지향적인 관점에서 구현



Samchon Framework 는 네트워크 시스템을 객체지향적인 관점에서, 마치 S/W 오브젝트를 다루듯이, 구축할 수 있게 해 줍니다. 여타 시스템과의 네트워크 통신이나 연동같은 것은 생각하지 않으셔도 됩니다. 네트워크 통신과 연동은 Samchon Framework 가 알아서 해 드립니다. 내 시스템과 네트워크를 통해 연동되는 여타 시스템들, 이들은 그냥 **객체**입니다.

내가 구성하려는 네트워크 시스템이, 각 시스템 노드간에 그 얼마나 복잡한 연결관계를 지니던, 각 시스템 노드는 그냥 하나의 객체입니다. 객체에서 객체로, 그저 타 객체의 함수를 부르고 파라미터를 전달하듯, 그렇게 만들어 나가시면 됩니다.

### 시스템 템플릿, 재사용성의 극대화



반복컨데, Samchon Framework 는 “객체지향 네트워크 프레임워크” 입니다. 네트워크 시스템을 객체지향적인 관점에서 구현할 수 있으니 응당, 자주 쓰이는 네트워크 시스템들을, 모듈화 (템플릿화) 하여 언제든 재사용하는 것 또한 가능합니다. 이 것이 바로 **System Template** 입니다.

클라우드 시스템이 필요하면 Cloud Service 이라는 시스템 템플릿을, 분산-병렬처리시스템을 구축해야 한다면 Distributed System 이라는 시스템 템플릿을 가져다 쓰시면 됩니다. 여러분이 자주 쓰시는 특유의 네트워크 시스템 구조가 있다면, 그 또한 모듈화 (템플릿화) 하여 재사용 하십시오.

아래는 Samchon Framework 에서 기본적으로 제공되는 System Template 의 목록입니다.

* Cloud Service: 클라우드 서버 템플릿
* External System: 1: N (일대다) 의 네트워크 시스템 제어에 관한 템플릿
  + Parallel System: 병렬처리시스템 템플릿
  + Distributed System: 분산-병렬처리시스템 템플릿

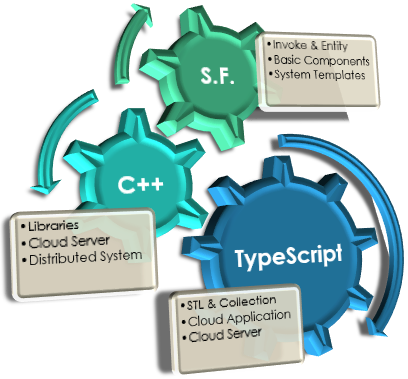
### C++ 과 TypeScript 를 연동한 통합시스템 구축

Samchon Framework 는 두 가지 언어, C++ 과 TypeScript 를 지원합니다.

TypeScript 는 높은 생산성을 자랑하되, 성능에서 열위에 있습니다. 반대로 C++ 은 생산성에서 떨어지나 성능에서 우위를 보입니다. 이렇게 서로 다른 특색을 보여주는 두 언어를, Samchon Framework 는 다음과 같이 활용하기를 권장합니다.

* **빠른 런칭**
  + TypeScript 와 HTML 로 클라우드 어플리케이션 (프론트, UI) 를 제작
  + TypeScript 와 NodeJS 로 클라우드 서버를 제작
* **퍼포먼스 개선**
  + 무거운 작업을 클라우드 서버 (NodeJS) 에서 C++ 시스템에 분산하여 처리
  + TypeScript 로 만들어진 클라우드 서버의 기능을 조금씩 C++ 로 이관시켜 나감

#### TypeScript-STL

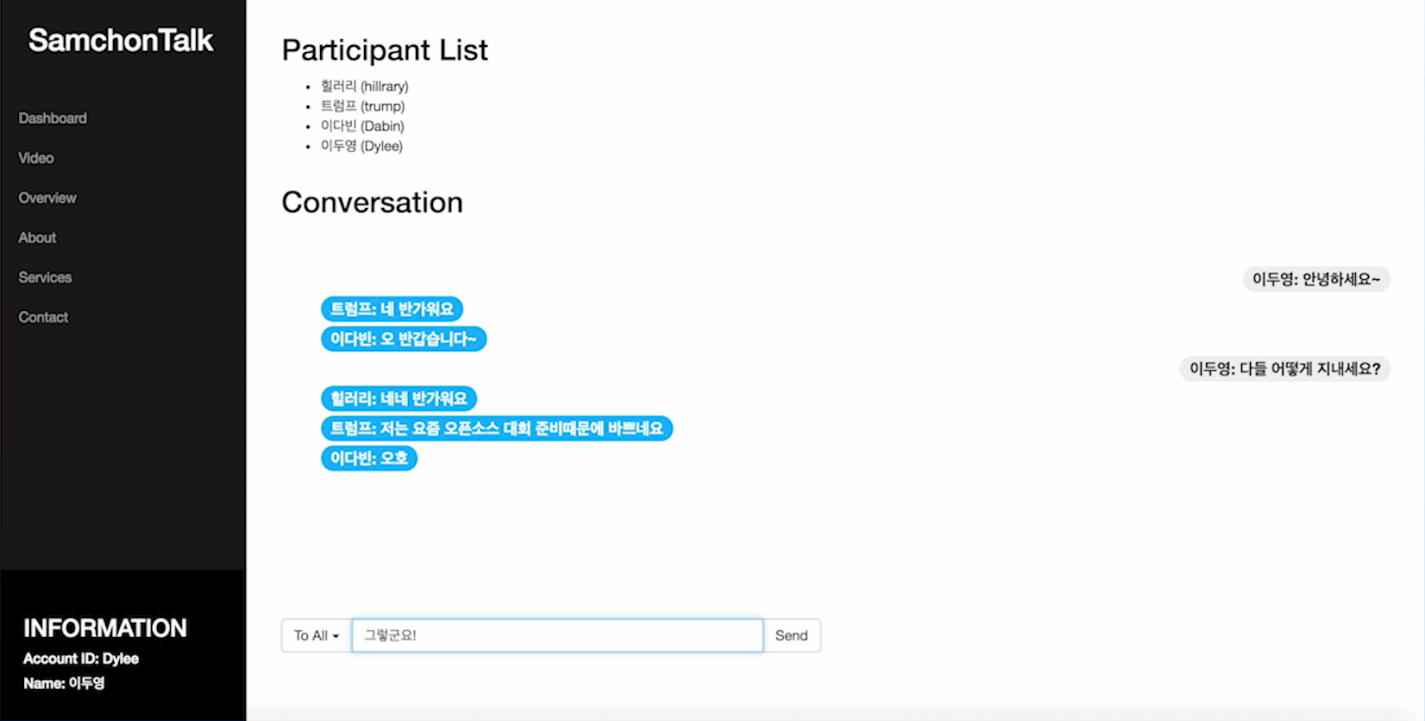
[](https://github.com/samchon/framework#interaction)서로 다른 언어로 만들어진 이종 시스템을 통합하고 마이그레이션함에 있어, 두 언어와 시스템이 같은 문법과 설계를 공유한다면, 그보다 좋을 수가 없습니다. 그리고 그보다 좋을 수 없을 바로 그 방안을, Samchon Framework 는 제공합니다.

같은 설계는 바로 Samchon Framework 의 객체지향네트워크 패러다임을 이용하는 것, 그리고 같은 문법은 **TypeScript-STL** 이라는 서브 프로젝트를 통해 지원합니다.

TypeScript-STL 은 C++ 의 STL (Standard Template Library) 을 TypeScript 로 마이그레이션 한 프로젝트입니다. C++ STL 의 컨테이너와 알고리즘 일체를 TypeScript 에서 사용할 수 있습니다.

이로써, Samchon Framework 를 사용하면, C++ 과 TypeScript 이 두 언어는 설계 뿐 아니라, 세부 문법과 자료구조 및 알고리즘까지 통일해 사용할 수 있습니다. TypeScript 를 이용하여 빠르게 런칭한 서비스를, C++로 마이그레이션함에 있어 어려울 것은 하나 없습니다. 그 반대 역시 마찬가지이구요.

## **Chatting Application**



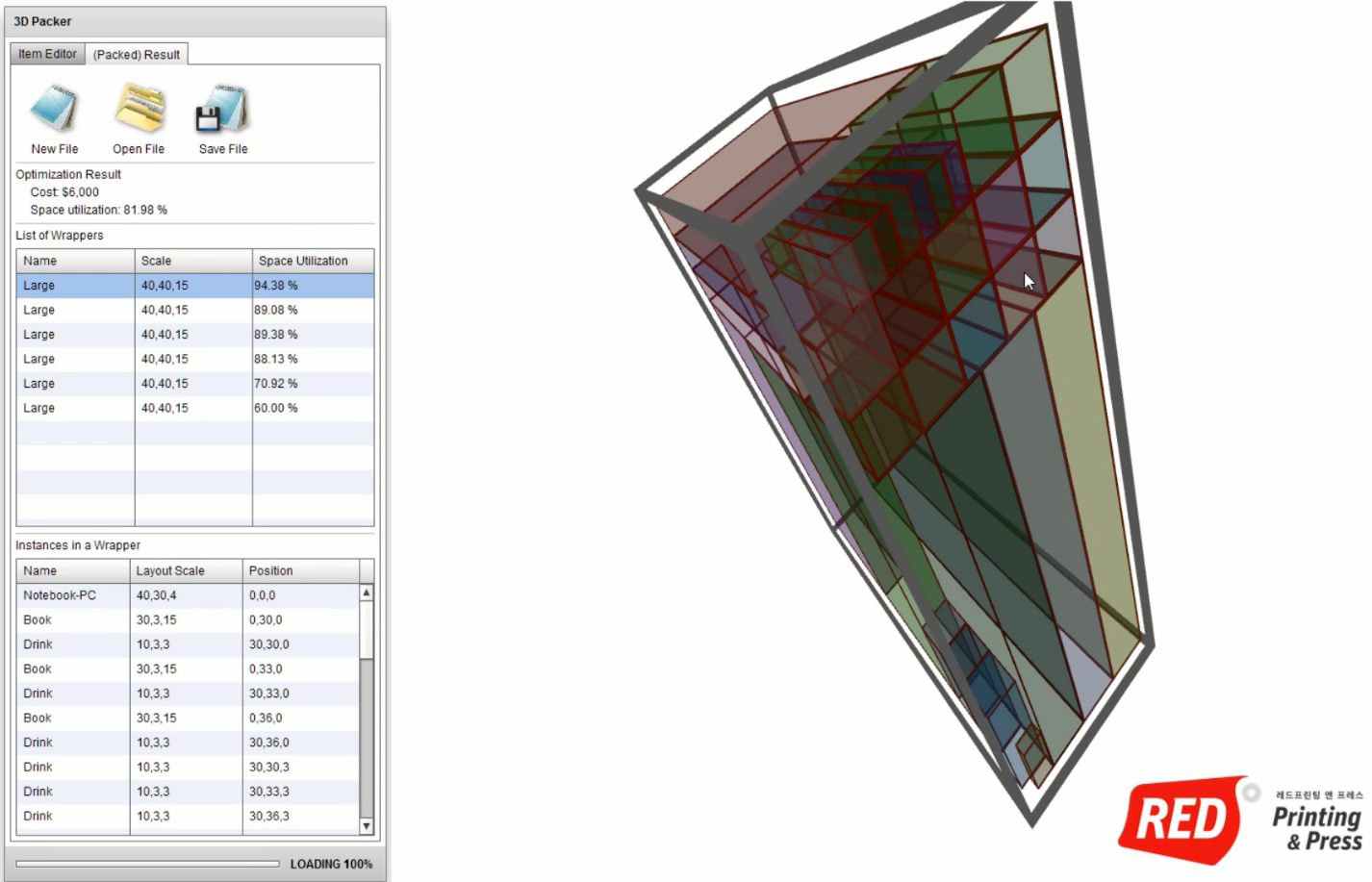
* 데모: <http://samchon.org/chat>
* 가이드 문서: <https://github.com/samchon/framework/wiki/TypeScript-Templates-Cloud_Service>
* API 문서: <http://samchon.github.io/framework/api/ts/modules/samchon.templates.service.html>

System Template 中 Cloud Service 모듈을 사용해 만든 채팅 어플리케이션 예제입니다.

채팅 서비스 특유의 푸쉬 서비스나 실시간 연동과 같은 소켓, 네트워크적인 개념들을 일절 고려하지 않고, 삼촌 프레임워크의 OON 패러다임에 따라 채팅, 그 자체의 로직에만 집중해서 간단히 만들어낸 (실시간) 채팅 서비스입니다.

Chatting Application 의 가이드 문서를 보고 코드를 따라 입력하면서, 삼촌 프레임워크의 OON 패러다임과 구체적인 활용방법을 배울 수 있는 예제입니다.

## **3D Bin Packing**



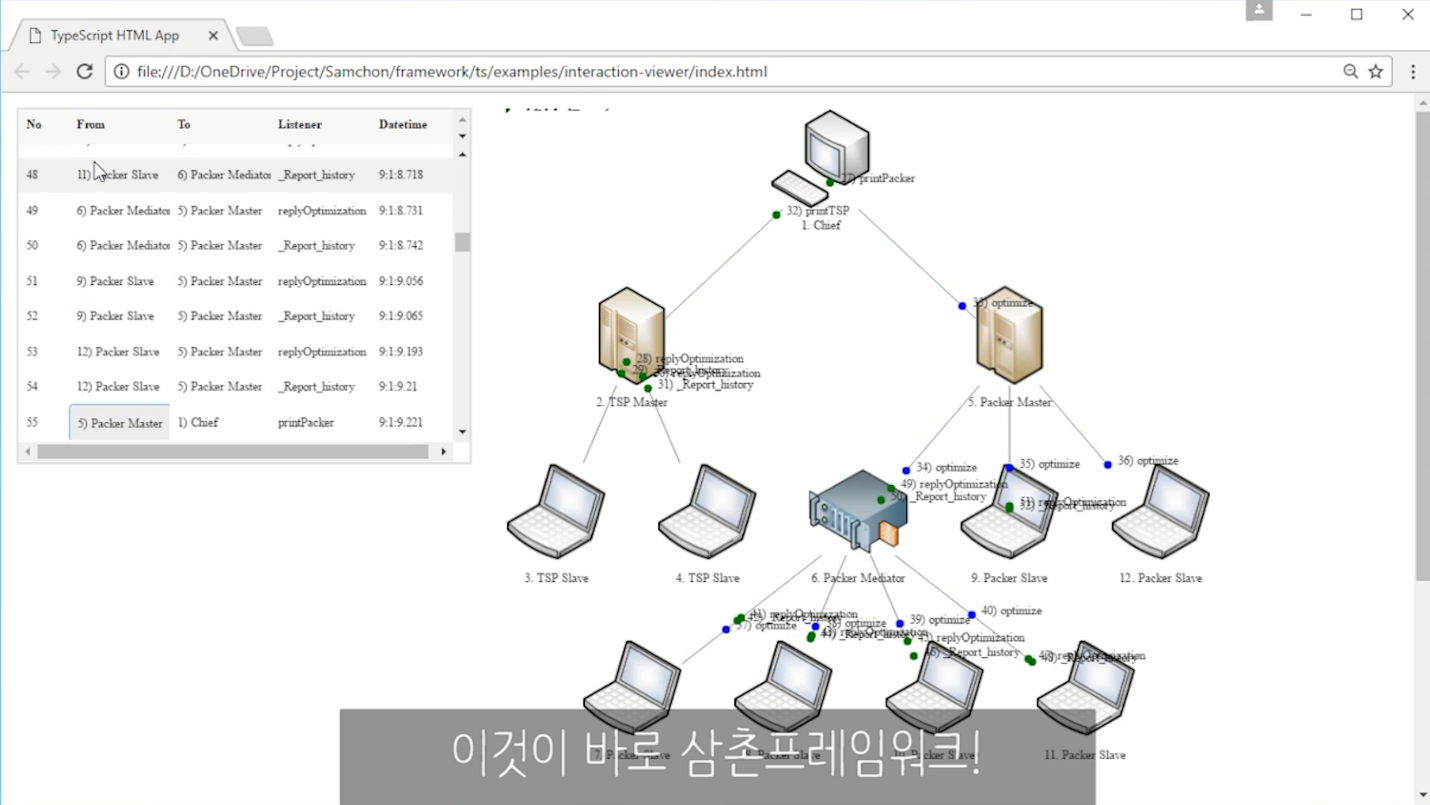
* 데모: <http://betterwaysystems.github.io/packer/demo>
* 깃허브: <https://github.com/betterwaysystems/packer>

삼촌 프레임워크와 TypeScript-STL 을 사용해 만든 클라우드 3D Bin Packing 솔루션입니다.

TypeScript (JavaScript) 혼자서 자체적인 3D Bin Packing 연산을 수행할 수 있는 standalone 모드와, C++ 에 연동하여 보다 정교하고 빠른 연산을 수행할 수 있는 분산처리 모드가 있습니다. TypeScript standalone 모드의 경우, TypeScript-STL 을 적극 활용하여, C++ 과 문법적으로 대동소이합니다.

위 3D-Bin-Packing 은, 이종의 언어를 통합, 마이그레이션함에 있어, 문법과 알고리즘을 통일하는 것이 얼마나 유용한지를 잘 증명해주는 예제 프로젝트입니다.

## **Interaction**

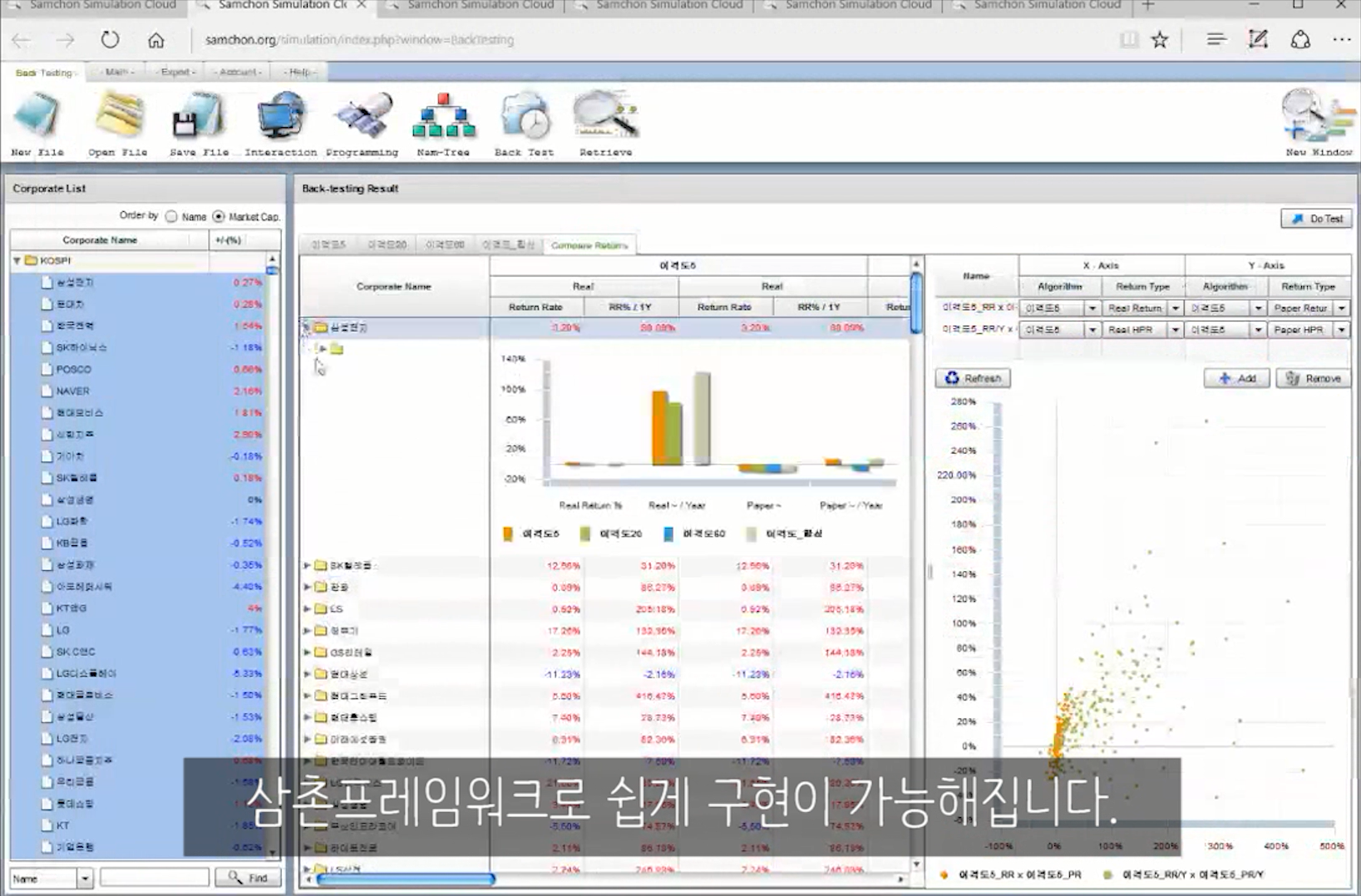


* 가이드 문서
  + External System: <https://github.com/samchon/framework/wiki/TypeScript-Templates-External_System>
  + Parallel System: <https://github.com/samchon/framework/wiki/TypeScript-Templates-Parallel_System>
  + Distributed System: <https://github.com/samchon/framework/wiki/TypeScript-Templates-Distributed_System>
* API 문서
  + External System: <http://samchon.github.io/framework/api/ts/modules/samchon.templates.external.html>
  + Parallel System: <http://samchon.github.io/framework/api/ts/modules/samchon.templates.parallel.html>
  + Distributed System: <http://samchon.github.io/framework/api/ts/modules/samchon.templates.distributed.html>

Interaction 은 트리구조의 분산-병렬처리시스템 및 이에 관한 모니터링 어플리케이션을 구현한, 삼촌 프레임워크와 OON (Object-Oriented Network) 패러다임의 궁극적인 활용을 보여주는 궁극의 예제 프로젝트입니다.

사용자는 Interaction 에 관한 가이드 문서를 보고, 설계도를 따라 그리면서 OON 의 패러다임을 심도있게 이해할 수 있고, Interaction 의 아키텍처에 대한 전반적인 이해를 거친 후, 예제 코드를 따라쳐 봄으로써 삼촌 프레임워크의 고급 사용자로 거듭날 수 있습니다.

## **Samchon Simulation**



* 주제: 주식 시뮬레이션 및 시스템 트레이딩에 관한 클라우드 솔루션
* 데모: <http://samchon.org/simulation>
* (매뉴얼 첨부됨)

Samchon Simulation Cloud 는 주식 시뮬레이션 및 시스템 트레이딩에 관한 클라우드 솔루션입니다. 더불어 Samchon Simulation Cloud 는 Samchon Framework 를 활용하여 만든 프로젝트 中 가장 모범적인 프로젝트입니다.

기본적으로 주식 시뮬레이션이나 시스템 트레이딩은, 그 로직이나 알고리즘 자체가 상당히 어렵습니다. 여기에 클라우드 시스템이니, 분산처리시스템이니 네트워크 연동이니 하는 것들이 덕지덕지 붙기 시작하면, 제작 난이도가 겉잡을 수 없이 높아져 버립니다.

때문에, 네트워크 시스템 구축에 대한 부담을 덜어버리기 위하여, 즉 비즈니스 로직과 네트워크 로직을 분리키 위하여 Samchon Framework 를 활용, Samchon Simulation Cloud 를 제작하게 됩니다.

# **기능 시험항목**

## **Samchon Framework**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류** | **중분류** | **소분류** | **시험항목, 활용예제** | **Pass & Fail** |
| TypeScript-STL | Containers | Linear | 프로젝트 전반 |  |
| Adaptor |  |
| Associative |  |
| Global | Algorithm |  |
| Exception |  |
| Functional |  |
| Utility |  |
| Collection | Containers | Linear | Packer |  |
| Associative |  |
| Library | Data | XML | 전년도 테스트 항목 |  |
| SQL interface |  |
| Utilities | |  |
| Math | Case Generator | Interaction |  |
| Genetic Algorithm | TSP, 3D-Bin-Packing |  |
| Protocol | Basic Components | Server | 프로젝트 전반 |  |
| Communicator |  |
| Iprotocol |  |
| Standard Message | Entity |  |
| Invoke |  |
| System Templates | Cloud Service | | Chatting |  |
| External System | External System | Interaction |  |
| Parallel System |  |
| Distributed System |  |
| Slave System |  |
| 문서 | 개발가이드 | |  |  |
| API 문서 | C++ |  |
| TypeScript |  |
| 설계도 | C++ |  |
| TypeScript |  |

## **Chatting Application**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류** | **중분류** | **소분류** | **시험항목** | **Pass & Fail** |
| 로그인 | | | 아이디와 닉네임을 정해 채팅 서비스에 참여 |  |
| 방 목록 | 리스트 | 방 목록 | 개설된 방 목록과 각 방의 참여자 목록을 보여준다 |  |
| 실시간 변동 | 채팅방 및 참여자들의 변동사항이 실시간으로 반영된다 |  |
| 방 개설 | | 채팅방을 개설한다 |  |
| 참여하기 | | 채팅방에 참여한다 |  |
| 채팅방 | 리스트 | 참여자 목록 | 현 채팅방의 참여자 목록을 보여준다 |  |
| 실시간 변동 | 현 채팅방 참여자들의 변동사항이 실시간으로 반영된다 |  |
| 대화하기 | 전체에게 | 채팅방 전체 참여자에게 대화를 한다 |  |
| 귓속말 | 특정 유저에게 귓속말을 한다 |  |
| 나가기 | | 현 채팅방에서 나간다 |  |

## **3D Bin Packing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류** | **중분류** | **소분류** | **시험항목** | **Pass & Fail** |
| 에디터 | 파일 입출력 | | 에디터 혹은 포장 결과를 저장하거나 불러옴 |  |
| 편집 | 물품 관리 | 포장하려는 물품 및 수량 목록을 관리함 |  |
| 포장지 관리 | 포장지 리스트를 관리함 |  |
| 연산 | 포장하기 | 단일 포장 | 한 종류의 포장지를 사용하는 포장비용 최적화 |  |
| 다중 포장 | 여러 종류의 포장지를 사용한 포장비용 최적화 |  |
| 연동 | 자체 연산 | 3D Bin Packing 을 Browser (JavaScript) 가 자체 수행 |  |
| 분산 연산 | C++ 서버에서 3D Bin Packing 을 수행한 후 결과보고 |  |
| 결과 (뷰어) | 그리드 | 물품 그리드 | 각 포장지 내 물품들의 배치 좌표 및 방향을 담은 그리드 |  |
| 포장지 그리드 | 사용된 포장지들에 대한 정보를 담은 그리드 |  |
| 3D 뷰어 | 전체보기 | 포장지와 포장지 내 물품내역을 3D 모델로 볼 수 있다. |  |
| 부분보기 | 포장지 내 일부 품목에 한하여 3D 모델로 봄 |  |
| 확대 및 돌려보기 | 3D 모델을 확대하거나 여러 각도로 돌려서 볼 수 있다. |  |

## **Interaction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류** | **중분류** | **소분류** | **시험항목** | **Pass & Fail** |
| 분산처리 | 역할 고정 |  | 각 슬레이브가, 고유의 역할을 가지고 프로세스를 독점 |  |
| 역할 변동 | 프로세스 분산 | 가장 유휴상태에 있는 슬레이브에 프로세스를 명령함 |  |
| 리소스 평가 | 각 프로세스의 리소스 (필요 성능량) 을 측정, 평가함 |  |
| 슬레이브 I/O | 슬레이브의 추가 및 접속끊김에 대응한다. 슬레이브가  프로세스를 수행 중에 갑자기 나갈 경우, 해당 프로세스를 회수하여 다른 최 유휴 시스템이 분산함 |  |
| 병렬처리 | 병렬처리 | 작업 분할 | 작업을 각 시스템에 알맞게 분할 (각 시스템의 평가된 성능 수치에 근거) 하여 병렬처리케 함 |  |
| 성능 평가 | 각 시스템의 성능을 측정, 평가함 |  |
| 중개자 | 마스터 | 중개자는 슬레이브를 거느리는 마스터 역할을 수행 |  |
| 슬레이브 | 마스터에게 있어, 중개자는 슬레이브이다. |  |
| 슬레이브 I/O | IN | 슬레이브가 새로 들어옴에 대응 |  |
| OUT | 슬레이브가 병렬처리 작업을 수행 중에 갑자기 나갈 경우, 명령한 작업을 회수하여 다른 슬레이브들에 분배 |  |
| 모니터링 | 시스템 구조 | 구조도 | 분산병렬처리시스템의 구조를 다이어그램으로 표현 |  |
| 시스템 I/O | 시스템 I/O 가 발생할 때마다, 구조도가 알맞게 변경됨 |  |
| 데이터 흐름 | | 각 시스템 노드간 데이터 전송을 시각화하여 보여줌 |  |

## **Samchon Simulation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류** | **중분류** | **소분류 (시험항목)** | **시험항목** | **P / F** |
| 공통기능 | 문서화 | 문서로 출력 | 현재 보고있는 화면 혹은 데이터를 문서로 출력함 |  |
| 그림으로 출력 |  |
| 기업 목록 | 기업목록 출력 | 타깃이 되는 회사를 검색하고 선택한다. |  |
| 기업 검색 |  |
| 기업 선택 |  |
| 시세 조회 | 주가 조회 | 일일주가 조회 | 타깃 회사의 일일주가 (캔들스틱차트) 를 조회함 |  |
| 재무 요약정보 조회 | 타깃 회사의 재무정보에 대한 요약정보를 본다 |  |
| 재무정보 조회 | 투자지표 조회 | 타깃 회사에 대한 투자 및 재무정보를 본다 |  |
| 재무정보 조회 |  |
| 비교그래프 | 선택한 재무 및 투자항목들에 대하여 비교도표를 그림 |  |
| 비교정보 조회 | 비교기업 구성 | 비교할 기업들을 선택하거나, 추천 목록을 받아온다 |  |
| 상대주가 조회 | 선택된 기업들 간의 상대주가 차트를 본다 |  |
| 상대투자지표 조회 | 선택된 기업들 간의 투자지표를 비교해본다 |  |
| 시뮬레이션 (공통) | 설정 정보 | 설정정보 저장 | 시뮬레이션에 대한 설정정보를 저장하거나 불러옴 |  |
| 설정정보 불러오기 |  |
| 파라미터 창 | 시뮬레이션 설정정보 편집기 |  |
| 백 테스팅 (시뮬레이션) | 거래 조건 구성 | 알고리즘 리스트 관리 | 거래 알고리즘 리스트 (다중 시뮬레이션) 를 관리한다. |  |
| 매수 알고리즘 구성 | 주식 거래 알고리즘을 구성한다. |  |
| 매도 알고리즘 구성 |  |
| 남트리 파일 관리 | 거래 알고리즘을 구성하는 남트리 파일을 관리한다. |  |
| 모의 거래 | 시뮬레이션 수행 | 주어진 조건 (남트리) 으로 백 테스팅을 수행 |  |
| 데이터 시각화 | 개별 거래결과 조회 | 각 종목 (기업) 별 가상 거래내역을 본다 |  |
| 시뮬레이션 결과 도표 | 행해진 거래에 따른 수익률과 매매내역을 도표로 봄. |  |
| 수익률 그래프 | 상호 기업 또는 상호 시뮬레이션 간 수익률을 비교. |  |
| 몬테카를로 (시뮬레이션) | 파라미터 설정 | 스크립트 입력 | 시뮬레이션에 사용할 스크립트를 입력함 |  |
| 시뮬레이션 | 시뮬레이션 수행 | 입력된 스크립트로 몬테카를로 시뮬레이션을 수행 |  |
| 데이터 시각화 | 시뮬레이션 결과 도표 | 각 종목별 수익률 분포를 도표로 보여줌 |  |
| 개별 분포도 조회 | 각 종목별 상세 분포도를 그래프로 보여줌 |  |