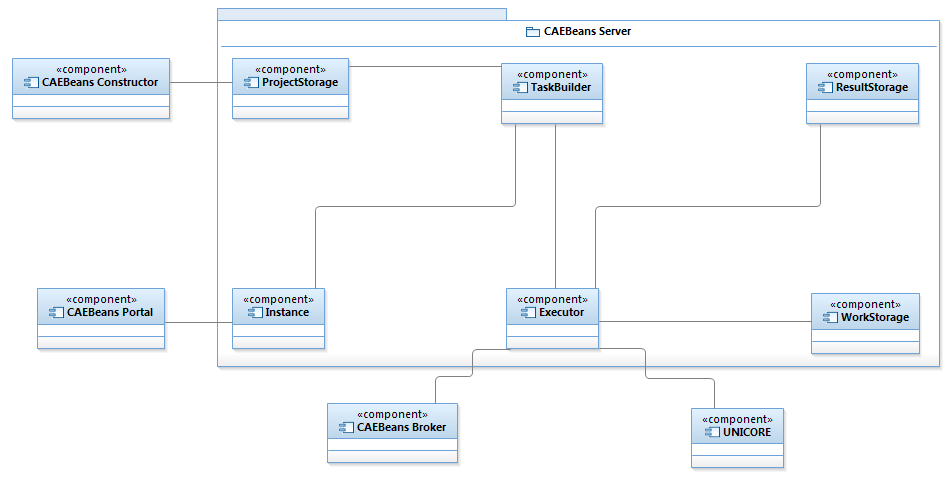
Студент: Иванов Иван Иванович, ВМИ-399

Руководитель: Петров Петр Петрович

Тема Разработка сервиса исполнения распределенных виртуальных испытательных стендов на базе платформы UNICORE.

***Структура и компоненты CAEBeans Server***



**Рис. 2. Компоненты CAEBeans Server**

CAEBeans Server состоит из шести основных компонентов:

1. *ProjectStorage* – компонент, ведущий учет загруженных проектов, по запросы выдает ссылки на файл описания проекта и файл проблемного слоя;
2. *TaskBuilder* – компонент, выполняющий преобразование описания проекта и проблемного слоя в комплексное описание задачи;
3. *WorkStorage* – компонент, хранящий промежуточные результаты выполнения задачи;
4. *ResultStorage* – компонент, занимающийся учетом и хранением результатов выполнения задач;
5. *Instance* – компонент, представляющий интерфейс CAEBeans Server в виде веб-сервиса;
6. *Executor* – компонент, занимающийся исполнением задачи согласно комплексного описания проекта, построенного компонентом *TaskBuilder*.

Для связи компонентов был использован фреймворк Guice, позволяющий компоненту избавиться от ссылок на объекты и взаимодействовать с другими компонентами только при помощи ссылок на интерфейсы, т. е. CAEBeans Server является слабосвязанной системой.

1. **Компонент ProjectStorage**

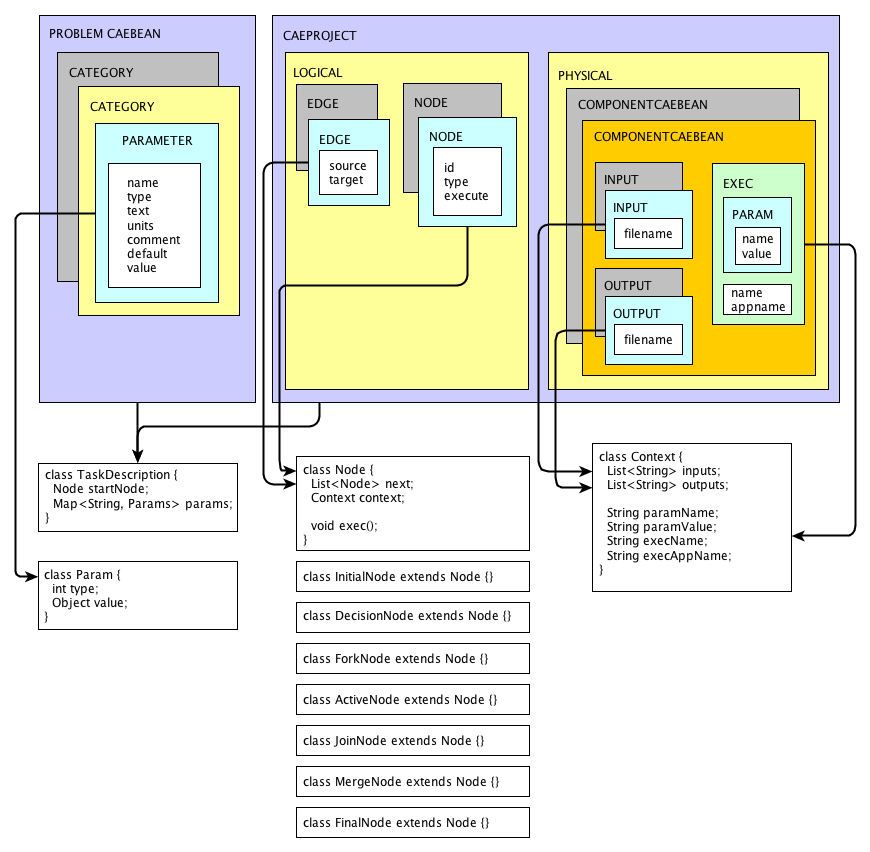
Основной задачей компонента является учет проектов, находящихся на сервере. Интерфейс компонента представлен двумя функциями:

1. *getProjectDescription* – возвращает указатель на файл, содержащий описание указанного проекта;
2. *getProjectDirectory* – возвращает указатель на директорию текущего проекта;

Каждый из хранимых проектов должен иметь уникальный идентификатор, совпадающий с именем директории, в которой находится описание этого проекта и все файлы, необходимые для его выполнения на конкретных входных данных.

1. **Компонент TaskBuilder**

Рисунок 3 отражает сущность преобразования описания проекта и проблемного слоя в комплексное описание задачи. Комплексное описание содержит список узлов задачи, каждый из который содержит ссылку на контекст (объект, содержащий текущую рабочую директорию для узла и задачи в целом, информацию из компонентного CAEBean, а также информацию о текущем проекте), список ссылок на следующие узлы (в случае начального узла — список из одного элемента, в случае конечного — пустой список), а также метод *exec*, инициирующий выполнение задачи, характерной для данного конкретного узла. Выполнение узлов производится при помощи запуска отложенных заданий для каждого узла из списка последующих.



**Рис. 3. Преобразование присылаемых данных в комплексное описание проекта**

1. **Компонент WorkStorage**

WorkStorage – компонент, отвечающий за управление директорией, в которой происходит исполнение задачи. Интерфейс компонента состоит из 8 методов:

1. *saveProblemCAEBean* – метод, сохраняющий присланный CAEBeans Portal проблемный слой задачи в файл, используемый затем при генерации входных файлов для задачи;
2. *getProblemCAEBean* – метод, возвращающий указатель на файл, содержащий проблемный слой задачи;
3. *createWorkingDirectory* – метод, создающий рабочую директорию задачи, имя директории совпадает с ID экземпляра, выполняющего обработку задачи (в рабочую директорию копируется все содержимое директории проекта);
4. *deleteWorkingDirectory* – метод, выполняющий удаление рабочей директории задачи, перед удалением происходит копирование результирующих файлов всех активных узлов в директорию результатов текущего задания;
5. *createNodeDirectory* – метод, создающий рабочую директорию для указанного узла, при этом в директорию копируются только те файлы, которые были указаны в компонентном CAEBean для этого узла;
6. *deleteNodeDirectory* – метод, удаляющий рабочую директорию указанного узла, при этом из директории узла в директорию задачи копируются файлы, которые были указаны как выходные в компонентном CAEBean для этого узла;
7. *getWorkingDirectory* – метод, возвращающий указатель на рабочую директорию указанного задания;
8. *getPreprocess* – метод, возвращающий абсолютный путь до JAR-пакета, содержащего классы для предварительной обработки данных конкретного узла.
9. **Компонент ResultStorage**

Компонент ResultStorage отвечает за создание и обслуживание директории, хранящей результаты выполнения того или иного задания. Интерфейс компонента состоит всего из одного метода:

1. *getResultDirectory* – метод возвращает ссылку на директорию, хранящую результаты выполнения задачи. Если директория не существует, то она будет создана.
2. **Компонент Instance**

Основой для функционирования CAEBeans сервера является *Java API for XML Web-Services* (JAX-WS), штатное средство для построения XML веб-сервисов платформы Java SE 6.

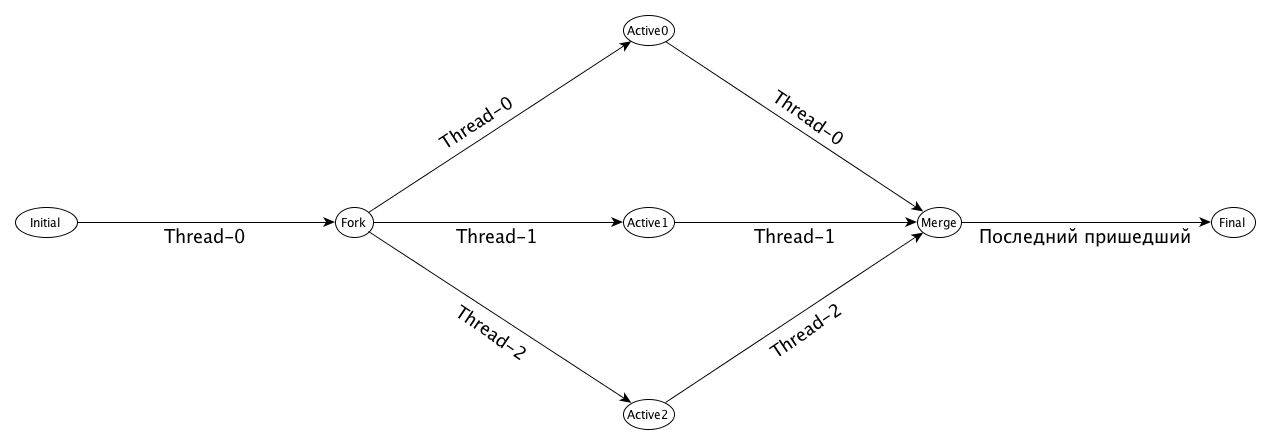
CAEBeans Server в терминах JAX-WS представляется в виде двух интерфейсов:

1. *CAEServer* – интерфейс, декларирующий операции, касающиеся сервера в общем; содержит одну операцию – создание экземпляра Instance, представляющего конкретную выполняющуюся задачу; операция возвращает документ, содержащий ID конкретного экземпляра;
2. *Instance* – интерфейс, декларирующий основные операции, касающиеся выполняющегося задания: запуск задания и получение его статуса. Для выполнения каждого из методов необходимо помимо специфических параметров передавать и ID конкретного экземпляра, возвращаемого CAEServer.
3. **Компонент Executor**

Executor – это компонент, занимающийся исполнением задачи в соответствии с комплексным описанием проекта, построенного компонентом TaskBuilder. На рисунке 4 виден процесс исполнения логического слоя задачи.

Изначально исполнение начинается с одного потока, поочередно исполняющего все узлы, которые встречаются ему на пути. При попадании в узел *Fork* происходит создание нового потока, который начинает исполнять параллельные ветви, основной же поток продолжает исполнение первой из веток. При попадании в узел *Merge* каждый поток может вести себя по-разному:

1. Если поток является последним из потоков, которые должны прийти в узел, он продолжает свое исполнение обычным образом;
2. Если же поток не является последним ожидаемым, его выполнение завершается.



**Рис. 4. Исполнение проекта в соответствии с его логическим слоем**

1. **Взаимодействие Executor с платформой UNICORE 6**

UNICORE 6 предусматривает API для построения пользовательских приложений, взаимодействующих с ядром UNICORE, кроме него существует собственная реализация интерфейса HiLA, который, однако, предоставляет существенно более узкий спектр возможностей, нежели API ядра UNICORE. Для взаимодействия с UNICORE/X были использованы фрагменты программного кода UNICORE Command line Client (UCC).

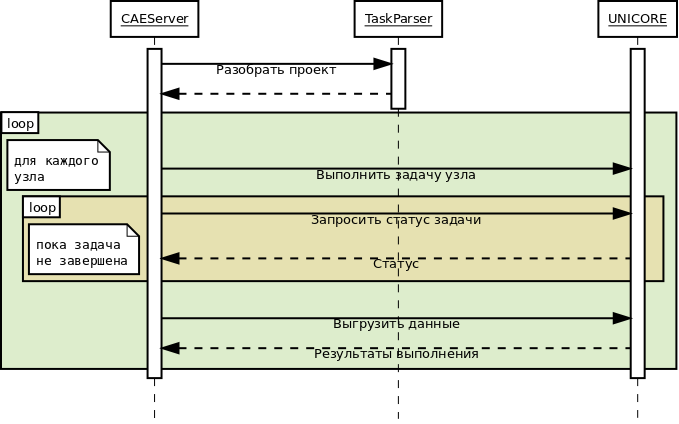


Рис. 5. Взаимодействие с UNICORE 6

Для каждого активного узла (*active node*) компонент executor производит последовательность следующих операций:

1. В директорию выполняющегося узла копируются необходимые файлы, подзадача ставится на выполнение в UNICORE.
2. В цикле проверяется статус задачи. Если задача завершена, происходит выгрузка всех необходимых файлов обратно в директорию задачи.