УДК 004.9, 004.51

ПЛАТФОРМА WALT ДЛЯ РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ

Кореньков Владимир Васильевич[1], Куняев Сергей Васильевич[2], Семашко Сергей Владимирович[3], Соколов Иван Александрович[4]

[1]Доктор технических наук, профессор;
Объединенный институт ядерных исследований,
141980, Россия, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: korenkov@jinr.ru.

[2]Кандидат технических наук;
Объединенный институт ядерных исследований,
141980, Россия, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: svk@jinr.ru.

[3]Инженер-программист 1 категории;
Объединенный институт ядерных исследований,
141980, Россия, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: semashko@jinr.ru.

[4]Стажёр-исследователь;
Объединенный институт ядерных исследований,
141980, Россия, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: isokolov@jinr.ru.

В работе представлена платформа «Web Application Lego Toolkit» (WALT). Она представляет собой шаблонно-ориентированную платформу, предназначенную для разработки web-приложений различной степени сложности. В отличие от многих других платформ, представляющих собой «волшебный чёрный ящик», его основная идея предоставление прозрачного, расширяемого и модифицируемого инструментария для решения типовых задач, которые постоянно возникают в процессе разработки. Платформа ориентирована на создание web-приложения любой сложности одним full-stack разработчиком или небольшой группой разработчиков. В статье представлена архитектура WALT и приведены примеры его использования в реальных проектах.

Ключевые слова: *web*-приложения, сервис, шаблоны, платформа, *java*, запрос.

Для цитирования: Платформа WALT для разработки web-приложений / В. В. Кореньков, С. В. Куняев, С. В. Семашко, И. А. Соколов // Системный анализ в науке и образовании: сетевое научное издание. 2021. № 4. С. 77–83. URL : http://sanse.ru/download/453.

PLATFORM WALT FOR WEB APPLICATION DEVELOPMENT

Korenkov Vladimir V.[1], Kuniaev Sergei V.[2], Semashko Sergei V.[3], Sokolov Ivan A.[4]

[1]Grand PhD in Engineering Sciences, professor;
Joint Institute for Nuclear Research,
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;
e-mail: korenkov@jinr.ru.

[2]PhD in Engineering Sciences;
Joint Institute for Nuclear Research,
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;
e-mail: svk@jinr.ru.

[3]Software engineer of the first category;
Joint Institute for Nuclear Research,
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;
e-mail: semashko@jinr.ru.

[4]*Intern researcher;*
Joint Institute for Nuclear Research,
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;
e-mail: isokolov@jinr.ru.

This paper describes the Web Application Lego Toolkit (WALT), which is a template-oriented platform designed for the development of web applications of various degrees of complexity. Unlike some other platforms, which represent a «magic black box», the main idea of WALT is to provide transparent, extensible, and modifiable tools for solving some specific problems that arise when developing web applications. WALT is designed to create a web application of any complexity by one full-stack developer or a small group of developers. The article depicts the WALT architecture and provides examples of its use.

Keywords: application, service, template, platform, java, request.

For citacion: Korenkov V. V., Kuniaev S. V., Semashko S. V., Sokolov I. A. Platform WALT for web application development. System Analysis in Science and Education, 2021;(4):77–83(In Russ). Available from: http://sanse.ru/download/453.

Введение

Существует множество различных платформ для разработки *web*-приложений: *Django, ASP.NET Core, Express, Angular* и т. д. [1, 2, 3, 4]. Они предоставляют разработчикам широкий спектр инструментов и библиотек, которые позволют создавать продвинутые *web*-приложения. Как правило, в крупных и средних *IT* компаниях, над проектом работает относительно большая команда разработчиков. При этом внутри команды есть разделение по характеру работы: дизайн и верстка, клиентская часть (*front-end*), серверная часть (*back-end)*, тестирование. Упомянутые платформы хорошо подходят для такой организации разработки и создания приложений.

Однако, в научной сфере и в нашей реальной жизни, очень часто разработкой *web*-приложений занимается один-два человека, которые, по сути, являются *full-stack* разработчиками. Кроме того, часто эти же люди параллельно занимаются работой над другими проектами и поддержкой уже введённых в эксплуатацию сервисов. Такой процесс разработки следует принципам гибкого *Agile* [5].

Представленная в статье платформа *WALT* хорошо подходит для быстрой разработки *web*-приложений разной степени сложности силами малой группы или одного разработчика. Кроме этого, она позволяет оперативно расширять и модифицировать функционал приложения в процессе его использования пользователями. Создаваемые с помощью платформы *web*-приложения отличаются высоким быстродействием и малым потреблением ресурсов. Платформа легка в освоении. Для ее использования разработчику достаточно: иметь базовые навыки верстки (*HTML + CSS*), уметь администрировать СУБД, владеть языком запросов *SQL*, изучить язык шаблонов *WALT* и ознакомиться с имеющимся набором сервисов *WALT*.

1. Описание платформы

Платформа реализована в виде библиотеки классов *JAVA* [6]. Однако, для решения типовых задач, она не требует от разработчика ее знания. *WALT* представляет собой шаблонно-ориентированную платформу, поэтому для использования шаблонов был разработан простой язык и соответствующий интерпретатор. Язык позволяет настраивать выводимый *HTML*-код, выполнять *SQL*-запросы к базе данных, а также изменять поведение обработки во время выполнения, в зависимости от конкретного пользовательского запроса и промежуточных результатов. Другими словами, шаблон говорит *Java*-программе «Что делать» (используя очень лаконичный язык) и она делает это. В *WALT* шаблоны определяют основную часть логики и поведения *web*-приложения. Более подробно о языке и шаблонах будет описано ниже.

WALT позволяет разработчику комбинировать функциональные возможности клиентской и серверной части в рамках одного модуля и перемещать функционал из *back-end* на *front-end* и наоборот. Он совместим с современными *front-end* инструментами, такими как *jQuery, Bootstrap* и т.д. Это

позволяет разработчику оперативно выбирать оптимальное распределение функционала непосредственно в процессе реализации *web*-приложения.

1.1. Архитектура WALT

Архитектура платформы *WALT* показана на рисунке 1. В ее основе лежит технология *JAVA*-сервлетов [7]. Для его работы необходим какой-нибудь контейнер сервлетов *(Servlet engine*), например, *Apache Tomcat* [8]. В ходе работы *web*-приложения *Servlet engine* получает *HTTP*-запрос от клиента и передает его сервлету *WALT*. В сервлете запрос проходит несколько общих начальных этапов обработки: создание и запуск отдельного потока для выполнения запроса, извлечение данных из запроса и формирование списка значений параметров, связь с базой данных, аутентификация пользователя и т.п. Пройдя общие этапы, на основе полученной из запроса информации, *WALT* выбирает конкретный модуль с шаблонами и запускает соответствующий сервис. Сервис извлекает необходимую информацию из шаблонов модуля и выполняет необходимые действия в соответствии с запросом клиента: обмен данными с базой данных, подготовка ответа клиенту, другие действия на стороне сервера. В завершение обработки запроса сервлет отправляет сформированный ответ клиенту.

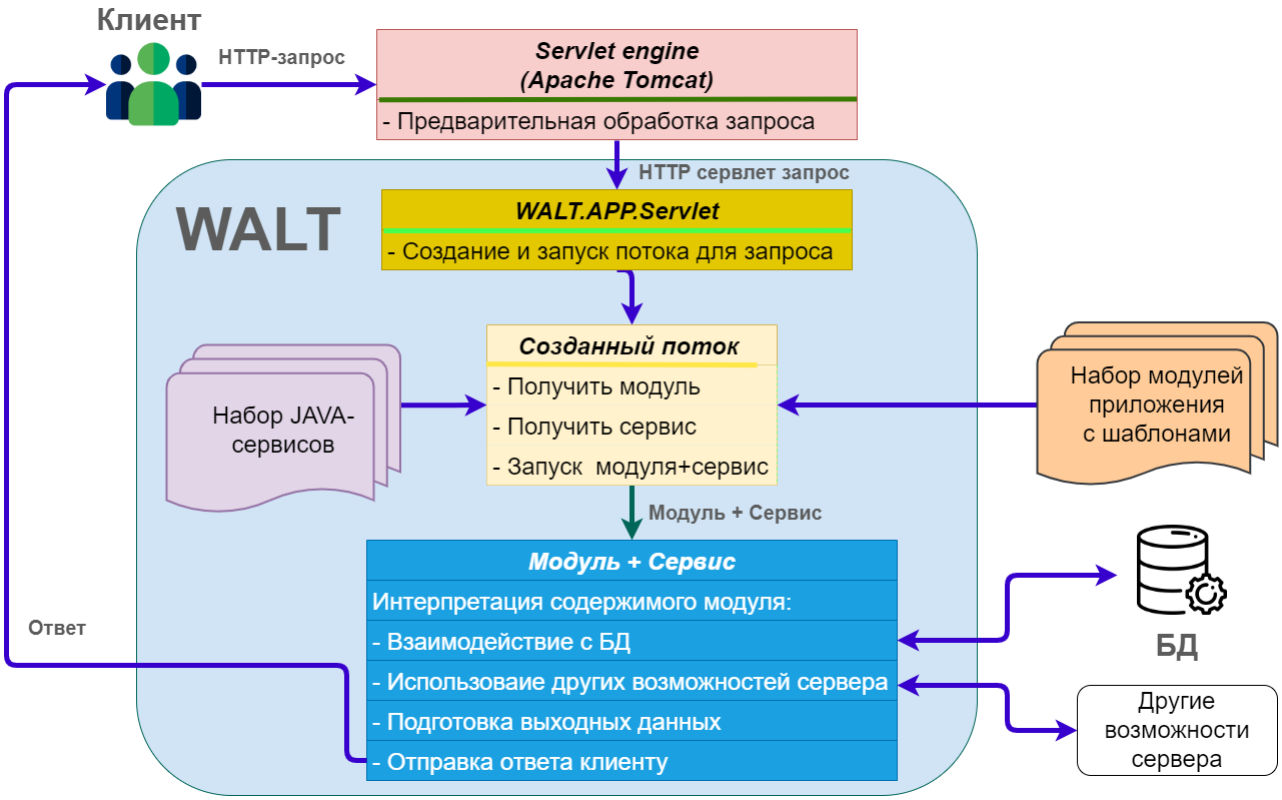

Рис. 1. Схема архитектуры WALT

1.2. Основные сервисы WALT

Для решения различных задач ядро *WALT* предоставляет разработчику 21 сервис. Сервис представляет собой класс написанный на *JAVA*, предназначенный для решения конкретного типа задач (работа с данными из базы данных, отправка писем, файловые операции и т.д.).

Базовым сервисом при разработке *web*-приложений является класс *dubna.walt.service.Service*. Он используется в качестве родительского класса для всех других сервисов. Кроме этого, *Service*, вместе с соответствующими модулями, может обрабатывать множество типовых задач: выполнение набора *SQL*-запросов к базе данных, подготовка выходного *HTML*-кода, выполнение некоторых файловых операций и т.д. При разработке наиболее часто используемыми сервисами являются:

- *TableServiceSimple, TableServiceSortable, TableServiceComplex, TableServiceSpecial* – различные виды таблично-ориентированных сервисов, которые могут выполнять *SQL*-запрос к базе данных

и обрабатывать полученный набор записей (как правило, выводить результаты клиенту, но также могут выполнять другие операции).

- *CrossTabService* – выполняет *SQL*-запрос и создает отчет с перекрестными вкладками аналогичный сводной таблице *EXCEL*.
- *CommandExecutor* – выполняет собственную команду на хосте и получает результат.
- *ServiceUploadFile* – выполняет загрузку файла на сервер.
- *ServiceBinaryData* – осуществляет отправку клиенту данных в бинарном виде для скачивания в виде файла или для какой-либо другой обработки.
- *SendMailService* – отправка *e-mail*.
- *ServiceImportData, ServiceCopyData* – осуществляет копирование некоторых данных из одной базы данных в другую.

Обычно базовый сервис и табличные сервисы позволяют решать не менее 95% задач. В редких случаях, когда существующие сервисы не могут решить какую-либо практическую задачу, или требуется модифицировать обработку данных/запроса, *WALT* позволяет расширить функциональность. Для этого необходимо реализовать новый сервис на основе существующего. Однако, от разработчика потребуются некоторые навыки программирования на *JAVA*.

1.3. Язык шаблонов WALT и директивы

Большая часть бизнес-логики содержится в шаблонах. Они группируются в модули. Модуль представляет собой простой текстовый файл, имеющий определённую структуру. Он состоит из секций, каждая из которых содержит фрагмент кода для решения некоторой части общей задачи. Использование секций позволяет следовать принципу «разделяй и властвуй», а также их можно использовать как внутри одного модуля, так и из других модулей, что позволяет вызывать один и тот же код в разных местах *web*-приложения.

На практике большинство задач в *web*-приложениях связаны с обработкой записей из базы данных. *WALT* позволяет решать такие задачи путем написания простого модуля с информацией о необходимых запросах к базе данных и о выводимом *HTML* (или любом другом) коде. Остальная часть работы выполняется основным сервисом *WALT*. Пример использования будет представлен в следующем разделе.

Важным базовым элементом синтаксиса шаблонов является конструкция «Подстановка параметр а». Она обозначается в коде как *#name#*, где *name* – имя параметра, значение которого мы хотим использовать в этом месте кода. В этом примере значение параметра с именем «*Age*» будет вставлено в код вместо #*age*#: *Age: #age#.*

Другой важный элемент синтаксиса «Условная проверка» – проверяет выполнение условия после двойного вопросительного знака, и, если результат равен *false,* то строка кода игнорируется и не выполняется. В этом примере строка «*Some_line_of code*» будет вставлена в код, если значение параметра «*ID*» больше 5 или значение параметра «*Name*» равно «*Ivan*» и значение параметра «*age*» меньше, чем 30: *Some_line_of_code* ??*ID*>5|*Name=Ivan*&*age*>30.

«Подстановка параметра» и «Условная проверка» позволяют управлять обработкой пользовательских запросов во время исполнения кода. Например, они могут быть использованы для настройки шаблона SQL-запроса в соответствии с запросом пользователя или для модификации выходного HTML кода (вывод определенных секций, применение тех или иных стилей и т.д.).

Ещё одним важным элементом синтаксиса является директива. Директива позволяет выполнить некоторое отдельное законченное действие на серверной части *web*-приложения. В шаблоне вызов директивы начинается со знака $. Всего в *WALT* реализовано 26 директив:

- $*INCLUDE* – вставить в код какой-то участок кода (секцию) из этого модуля или из другого. Позволяет повторно использовать код и делает его более компактным и читаемым.
- $*GET_DATA* – получить необходимые данные, выполнив какой-либо сценарий *SQL* или другими действиями (например, прочитать файл данных).

- $*CALL_SERVICE* – позволяет выполнить другой модуль *WALT* внутри родительского, что обеспечивает возможность разделить сложную задачу на несколько простых подзадач, а также повторно использовать код.
- $*JS*, $*JS_BEGIN*-$*JS_END* – предоставляет разработчику возможность использовать код *JavaScript* на стороне сервера. Часто избавляет разработчика от необходимости разрабатывать новый *JAVA*-сервис.
- $*READ_FILE*, $*COPY_FILE*, $*MOVE_FILE*, $*DELETE_FILE*, $*GET_FILE_SIZE* – организуют простые операции с файлами.
- $*GET_URL* – позволяет открыть указанный *URL* и получить ответ.
- А также: $*EXECUTE_LOOP*, $*IF*-$*ELSE* - $*ENDIF*, $*WAIT*, $*USE_DB*, $*CLOSE_DB*, $*STORE_PARAMETERS*-$*RESTORE_PARAMETERS*, $*GET_AUTH_URL*, $*BREAK*, $*PRINT*, $*LOG* и д.р.

Язык шаблонов *WALT* является емким и лаконичным, что позволяет создавать компактные модули для решения сложных задачи путем декомпозиции ее на простые части и повторного использования кода.

1.4. Пример использования

Код небольшого модуля и результат его работы представлены на рисунке 2. Представленный код выбирает данные о сотрудниках из базы данных и позволяет искать людей по началу фамилии. Код модуля содержит шаблон *SQL*-запроса к базе данных и выходного *HTML*-кода. Остальную часть работы по выборке данных и их выводу клиенту выполняет сервис *TableServiceSimple*. Таким образом, компактный код выполняет некоторую завершенную задачу.

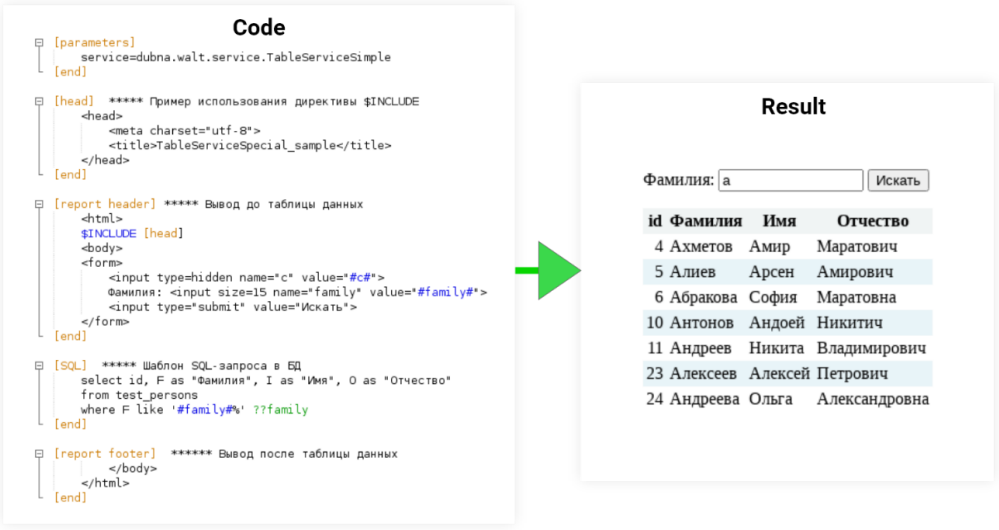

Рис. 2. Пример небольшого модуля и результат его работы

1.4. Корпоративные web-приложения ОИЯИ, разработанные с использованием WALT

WALT был применен для разработки ряда корпоративных *web*-приложений ОИЯИ различного уровня сложности. В таблице 1 приведён их перечень и ориентировочные трудозатраты на разработку первой рабочей версии в человеко-месяцах.

Табл. 1. Корпоративные web-приложения ОИЯИ, разработанные с использованием WALT

Приложение	**Назначение**	**Трудозатраты**
ADB2	Управленческий учет ОИЯИ	4
PIN	Информация о персонале в различных аспектах	6
EDMS «Documents	Электронное хранилище документов административной	6

DB»	деятельности	
NICA EVM	Структура проекта, рабочие планы, расходы, журнал затрат, отчеты и т. д.	4
EDMS «*Dubna*»	Система электронного документооборота ОИЯИ [10]	12
HR JINR	Информация о персонале в различных аспектах	4
Map JINR	Базовая карта полигонов ОИЯИ. Доступен для всех в режиме онлайн без ограничения	
Gateway	Универсальный шлюз для обмена данными между различными системами	2
ISSC	Система поддержки научной аттестации	3
CERN DB	Система обеспечения сотрудников ОИЯИ в ЦЕРНе: поездки, проживание, отчеты и т.д.	4
eco.jinr.ru	Структурированный список корпоративных приложений ОИЯИ с распределенным администрированием	0.5
EDMS «*Advance reports*»	Подготовка данных для бухгалтерской отчетности по командировкам	5
Checkpoint lists	Списки для доступа к площадкам ОИЯИ в режиме ограниченного доступа	0.25

Все эти приложения в настоящее время используются, некоторые из них продолжают развиваться. На разработку и запуск в тестовом режиме 1-й версии приложения уходило всего от одной недели до двенадцати человеко-месяцев, что подтверждает эффективность использования *WALT*. Кроме перечисленных приложений было разработано с десяток приложений за пределами ОИЯИ.

Заключение

Представленная платформа при очень малом своем объеме кода (~650 Кб) обладает рядом достоинств и способна облегчить разработку сложных web-приложений силами малой группы разработчиков. Благодаря лаконичному и емкому языку шаблонов, большому количеству доступных сервисов разработчик способен решать сложные задачи минимальным количеством кода в кратчайший срок. При возникновении специфичных задач *WALT* дает разработчику возможность расширить или модифицировать функционал. При этом *WALT* совместим с клиентскими средствами разработки типа *JQuery*, *AJAX*-технологией и т.д., что позволяет их использовать при разработке клиентской части приложения. Эффективность платформы подтверждает большое количество разработанных с её помощью web-приложений. В дальнейшем планируется развивать и улучшать платформу.

Список источников

1. Django: [Электронный ресурс]. URL: https://www.djangoproject.com/. (Дата обращения: 09.04.2022)
2. What is ASP.NET Core: [Электронный ресурс]. URL: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnetcore. (Дата обращения: 09.09.2021)
3. Express: [Электронный ресурс]. StrongLoop; IBM; OpenJS Foundation, 2017. URL: http://expressjs.com/.
4. Angular: The modern web developer’s platform: [Электронный ресурс]. URL: https://angular.io/. (Дата обращения 09.09.2021).
5. Manifesto for Agile Software Development: [Электронный ресурс] / K. Beck [и др.] // Agile Alliance. URL: https://agilemanifesto.org/.
6. Java / Oracle, 2022 . URLt: https://www.java.com/.
7. Java Servlet Technology: [Электронный ресурс]. URL: https://www.oracle.com/java/technologies/servlettechnology.html.
8. Apache Tomcat. URL: https://tomcat.apache.org/.

9. Scientific and organizational digital resources of the Joint Institute for Nuclear Research / V.F. Borisovsky [и др.] // Digital Libraries: Advanced Methods and Technologies, Proceedings of the RCDL 2008. P. 227. URL: http://rcdl2008.jinr.ru/pdf/277_283_paper33.pdf (Дата обращения: 12.04.2022)

10. Development and implementation of electronic document management system «EDMS Dubna» at JINR / I.N. Alexandrov [и др.] // CEUR Workshop Proceedings. 2016.Vol. 1787. P. 92

11. Нейросетевая реконструкция треков частиц для внутреннего CGEM-детектора эксперимента BESIII / Г.А. Ососков [и др.]// Компьютерные исследования и моделирование. 2020. Т. 12. №. 6. С. 1361–1381.

12. Goncharov P., Ososkov G., Baranov D. Particle track reconstruction with the TrackNETv2 // AIP Conference Proceedings. AIP Publishing LLC, 2019. Vol. 2163. No. 1. P. 040003.