Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.422

Ураев
Денис Алексеевич

Создание UI-фреймворка для использования в бизнес-ориентированных Web-приложениях

АВТОРЕФЕРАТ
на соискание академической степени
магистра информатики и вычислительной техники

по специальности 1-40 81 01 – Информатика и технологии разработки программного обеспечения

Научный руководитель
Хотеев Александр Леонидович
к.ф.-м.н., доцент

Минск 2017

ВВЕДЕНИЕ

В настоящее время довольно трудно представить себе современную цивилизацию без использования во всех сферах деятельности информационных технологий. Фактически информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных, и теперь уже большинство операций не может быть выполнено вручную, они требуют применения наиболее перспективных компьютерных технологий. Любые административные решения требуют четкой и точной оценки ситуации и возможных перспектив ее изменения. И, если раньше в оценке ситуации участвовало несколько десятков факторов, которые могли быть вычислены вручную, то теперь таких факторов сотни, а обоснованность принимаемых решений требуется большая, потому что и реакция на неправильные решения более серьезная. И, конечно, обойтись без автоматизации процессов в этом случае невозможно.

Процессы автоматизации затронули не только производственную, техническую и технологическую сферы деятельности человечества, но и информационное пространство; в частности издательства, библиотеки, музеи, информационные центры. Уже сложно представить жизнь общества без таких понятий как интернет, средства телекоммуникации и т.д.

Всё больше предприятий в мире автоматизирует свои бизнес-процессы. С развитием Web-технологий, все чаще выбор автоматизированных систем падаем на Web-приложения, где вся информация сможет храниться в "облаке". Наряду с этим, с увеличением мощности клиентских устройств, всё больше логики Web-приложений не ограничиваются простым набором библиотек. Поэтому от front-endразработки требуется современный подход в решении задач по автоматизации процессов.

В контексте данной работы я буду рассматривать процесс создания UI-фреймворка для бизнес-ориентированных Web-приложений. Также будет рассмотрен пример использования данного фреймворка в реализации пользовательского интерфейса в Web-приложении электронного документооборота.

КРАТКОЕ СОДЕРЖАНИЕ

Основной целью при разработке интерфейса для взаимодействия человека с машиной является создание UI, который позволит легко (интуитивно), эффективно, и с удовольствием (user friendly) добиваться желаемого результата во время работы с машиной. Идеальным результатом считается тот, при котором оператор (или пользователь) производит минимальные усилия для ввода, получая от машины желаемый вывод, при этом машина минимизирует вывод лишних данных.

Разработка UI – это создание интерфейса, который обеспечивает самый лучший, простой, приятный и не обременяющий способ взаимодействия пользователя с продуктом. Большую часть работы во время создания интерфейса составляет наблюдение за поведением пользователя, что позволяет принимать решения, основанные на собранных данных.

Фреймворк программной системы – это каркас программной системы (или подсистемы). Может включать: вспомогательные программы, библиотеки кода, язык сценариев и другое ПО, облегчающее разработку и объединение разных компонентов большого программного проекта. Обычно, объединение происходит за счёт использования единого API.

Одно из главных преимуществ при использовании "каркасных" приложений – "стандартность" структуры приложения. "Каркасы" стали популярны с появлением графических интерфейсов пользователя, которые имели тенденцию к реализации стандартной структуры для приложений. С их использованием стало гораздо проще создавать средства для автоматического создания графических интерфейсов, так как структура внутренней реализации кода приложения стала известна заранее. Для обеспечения каркаса, обычно используются техники объектно-ориентированного программирования (например, части приложения могут наследоваться от базовых классов фреймворка).

В архитектуре данного фреймворка был предусмотрен ООП подход в создании компонентов. На протяжении долго времени в программировании применялся процедурный подход, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Гораздо позже появилось объектно-ориентированное программирование, которое позволяет группировать функции и данные в единой сущности – "объекте".

При объектно-ориентированном подходе мы описываем поведение системы на уровне объектов, которые имеют определённые свойства и методы, имеют между собой связи и взаимодействуют друг с другом. В JavaScript существует два способа организации кода в ООП стиле: ООП в

функциональном стиле и ООП в прототипном стиле. В моей работе будет использоваться прототипное наследование, так как оно более прозрачное при реализации, а следовательно порог входа в создаваемый фреймворк будет ниже.

Реализуется наследование через неявную (внутреннюю) ссылку одного объекта на другой, который называется его прототипом и в спецификации обозначается [[prototype]]. Это свойство обычно скрыто от программиста.

Также существует свойство с похожим названием prototype (без квадратных скобок) – оно вспомогательное и указывает, откуда брать прототип при создании объекта.

Для отрисовки компонентов используется система шаблонизации. Основа данной системы заключается в том, что каждый не абстрактный компонент имеет свой собственный шаблон, записанный строкой в html формате. При написании приложения, каждый компонент, кроме корневого, имеет "родителя". И каждый компонент может иметь дочерние компоненты. Основываясь на этой иерархии, фреймворк отрисовывает шаблон каждого компонента на правильном месте. Такой подход упрощает создание интерфейса, так как интерфейс разбивается на множество мелких компонентов, каждый из которых отвечает за свою небольшую функцию.

Связь между компонентами системы происходит через события. Данный фреймворк является событийно-ориентированным, т.к. это позволяет сохранить слабую связанность между компонентами.

Архитектура, управляемая событиями (англ. event-driven architecture, EDA) является шаблоном архитектуры программного обеспечения, позволяющим создание, определение, потребление и реакцию на события. Событие можно определить как «существенное изменение состояния».

Этот архитектурный шаблон может применяться при разработке и реализации приложений и систем, передающих события среди слабосвязанных программных компонентов и служб. Система, управляемая событиями, обычно содержит источники событий (или агентов) и потребителей событий (или стоков). Стоки ответственны за ответную реакцию, как только событие возникло. Реакция может полностью или не полностью создаваться стоком. К примеру, сток может отвечать лишь за фильтрацию, преобразование и доставку события другому компоненту, либо он может создать собственную реакцию на это событие. Первая категория стоков может основываться на традиционных компонентах, таких как промежуточное программное обеспечение для обмена сообщениями, а вторая категория стоков (формирующая собственную реакцию в процессе работы) может потребовать наличия более подходящей платформы выполнения транзакций.

Создание приложений и систем в рамках архитектуры, управляемой событиями, позволяет им быть сконструированными способом,

способствующим лучшей интерактивности, поскольку системы, управляемые событиями, по структуре более ориентированы на непредсказуемые и асинхронные окружения.

Также хочется упомянуть о наличии во фреймворке вспомогательных классов для работы с различными типами данных и структурами. Например утилиты упрощают работы со *строками, объектом браузера, кэшем браузера, куками, CSS свойствами, датами, DOM объектами, объектами функций, картинками, JSON форматом, числами, регулярными выражениям, XML форматом*.

На основе фреймворка было создано приложение для электронного документооборота. Документы, поступающие в организацию, проходят: первичную обработку, предварительное рассмотрение, регистрацию, рассмотрение руководством, передачу на исполнение. На каждом из этапов пользователи могут внести свои изменения или комментарии к тексту.

У документов имеются различные состояния, такие как "в процессе редактирования", "одобренный", "закрытый" и др. На каждом из состояний есть возможность выполнить определённый набор действий.

Набор действий, которые может выполнить пользователь, также регулируется в зависимости от роли пользователя. В системе предусмотрена система управления пользователями, при которой документ может быть разделён между людьми и группами с определёнными наборами прав.

Также в системе предусмотрена система отслеживания истории редактирования документа и система версионирования документов.

СПИСОК ОПУБЛИКОВАННЫХ РАБОТ

Издательство: Журнал «Наука, образование и культура»

Публикация от 29 июня 2017 г № 06(21).

Организационно-экономические и правовые проблемы организации электронной коммерции

ЗАКЛЮЧЕНИЕ

В рамках данного диссертации на соискание академической степени магистра информатики и вычислительной техники был создан JavaScript фреймворк для бизнес-ориентированных Web-приложения. Также было создано приложение для автоматизации документооборота, с использованием данного фреймворка, чтобы продемонстрировать его состоятельность. В ходе разработки были выполнены все пункты задания для диссертации:

- создание объектно-ориентированного подхода (реализация инкапсуляции, наследования и полиморфизма);
- создание набора готовых компонентов;
- компиляция и сборка фреймворка.

Был разработан фреймворк, который позволяет повысить качество кода создаваемых приложений за счет написания структурированного кода, за счет наличия возможности простой поддержки и масштабируемости проектов. Также за счет грамотно продуманной архитектуры приложения и разбиения кода на компоненты, удалось уменьшить время, необходимое для написания новых компонентов системы. Также увеличивается качество проектов за счет снижения количества ошибок.

Результатом применения данного фреймворка является повышение конкурентоспособности написанных приложений.

Таким образом, поставленные цели и задачи диссертации достигнуты. Также учитываются все достоинства и недостатки существующих аналогов. Разработанное программное обеспечение соответствует сформулированным требованиям и успешно прошло опытную эксплуатацию.