РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 419 864** (13) **C2**

(51) МПК
G06F 17/30 (2006.01)
G06F 15/16 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2007137005/08, 09.03.2006**

(24) Дата начала отсчета срока действия патента:
09.03.2006

Приоритет(ы):
(30) Конвенционный приоритет:
08.04.2005 US 11/102,039

(43) Дата публикации заявки: **10.04.2009** Бюл. № 10

(45) Опубликовано: **27.05.2011** Бюл. № **15**

(56) Список документов, цитированных в отчете о поиске: **RU 2148856 C1, 10.05.2000. US 5539909 A, 23.07.1996. US 2002/0046303 A1, 18.04.2002. US 20000645048, 23.08.2000.**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **05.10.2007**

(86) Заявка PCT:
US 2006/008275 (09.03.2006)

(87) Публикация заявки PCT:
WO 2006/110237 (19.10.2006)

Адрес для переписки:
129090, Москва, ул.Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
ПЕРЕЙРА Хорхе (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

RU 2419864 C2

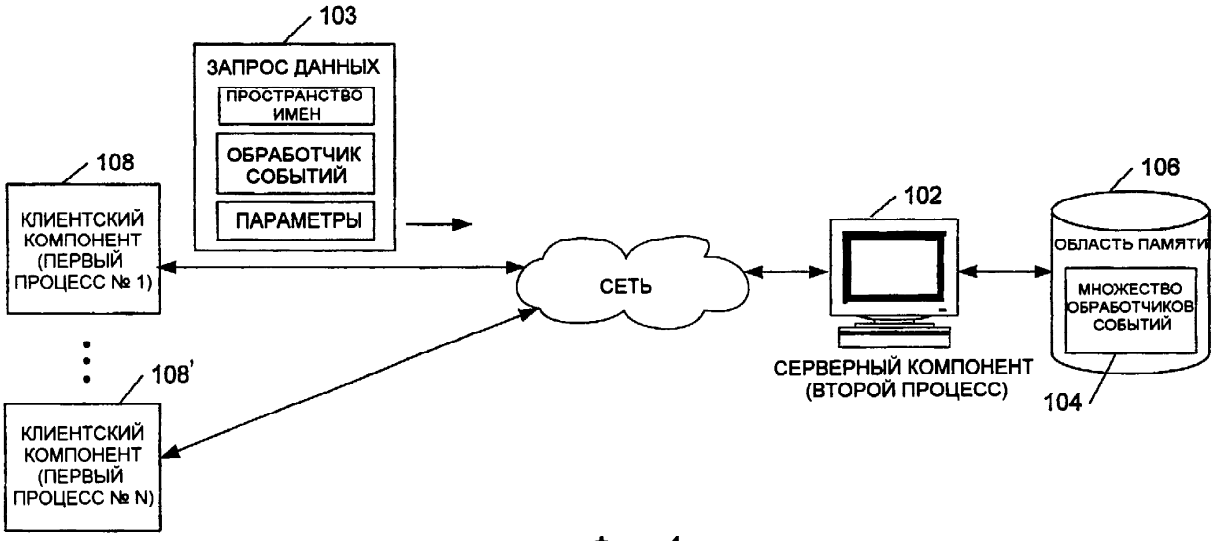
(54) СИСТЕМА И СПОСОБ ДЛЯ ФОРМИРОВАНИЯ И ПЕРЕДАЧИ ЗАПРОШЕННЫХ ДАННЫХ МЕЖДУ СЕТЕВЫМИ ПРИКЛАДНЫМИ ПРОГРАММАМИ

(57) Реферат:

Изобретение относится к запросам данных в форме схемы, имеющей пространство имен, события в пространстве имен и параметры для этих событий. Техническим результатом является увеличение надежности и быстродействия систем связи клиент-сервер. Система включает: считываемый компьютером носитель для хранения: множества обработчиков событий и структуру данных, представляющую форматированный запрос данных; и распределенное клиентское приложение, развернутое серверным компонентом, идентификации значения пространства имен, выбора значения обработчика событий, заполнения структуры данных, передачи заполненной структуры данных к серверному компоненту, серверный компонент, сконфигурированный так, чтобы выполнять выполняемые компьютером команды для: приема заполненной структуры данных, выбора, как функции принятой

RU 2419864 C2

структуры данных, обработчика событий из множества обработчиков событий, выполнения выбранного обработчика событий, передачи сформированных данных результата к распределенному клиентскому приложению. 3 н. и 15 з.п., ф-лы, 5 ил., 4 табл.



Фиг.1

RU 2419864 C2

RU 2419864 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 419 864** (13) **C2**

(51) Int. Cl.
G06F 17/30 (2006.01)
G06F 15/16 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2007137005/08, 09.03.2006**

(24) Effective date for property rights:
09.03.2006

Priority:
(30) Priority:
08.04.2005 US 11/102,039

(43) Application published: **10.04.2009** Bull. 10

(45) Date of publication: **27.05.2011 Bull. 15**

(85) Commencement of national phase: **05.10.2007**

(86) PCT application:
US 2006/008275 (09.03.2006)

(87) PCT publication:
WO 2006/110237 (19.10.2006)

Mail address:
129090, Moskva, ul.B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
PEREJRA Khorkhe (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

RU 2419864 C2

RU 2419864 C2

(54) SYSTEM AND METHOD OF GENERATING AND TRANSMITTING REQUESTED DATA BETWEEN NETWORK APPLICATION PROGRAMMES

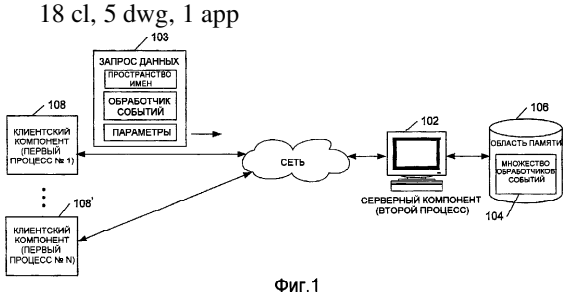
(57) Abstract:

FIELD: information technology.

SUBSTANCE: system includes: a computer-readable data medium; a plurality of event handlers and a data structure which is a formatted data request; and a distributed client application rolled out by a server computer, identification of name space values, selection of event handler values, populating the data structure, transmitting the populated data structure to the server component, the server component being configure to execute computer-executable instructions for: receiving the populated data structure, selecting, as a function of the received data structure, an event handler from multiple event handlers, execution of the selected event handler, transmitting the formed result data to the distributed client application.

EFFECT: high reliability and operation speed of client-server communication systems.

18 cl, 5 dwg, 1 app



Фиг.1

Область техники

Варианты осуществления настоящего изобретения относятся к области компьютерных обменов. В частности, варианты осуществления настоящего изобретения относятся к запросам данных в форме схемы, имеющей пространство имен, события в пространстве имен и параметры для этих событий.

Предшествующий уровень техники

Некоторые развертываемые сервером доступные через всемирную сеть (web) почтовые приложения предоставляют пользователям доступ к их центрально расположенным почтовым ящикам. Эти приложения также обеспечивают возможность посылать и принимать электронную почту (e-mail), назначать расписание и выполнять другие задачи персонального управления информацией (PIM) из сети, такой как Интернет. Некоторые из этих приложений выполняются в браузере и являются сопутствующими другим приложениям PIM. Такие клиентские web приложения требуют большого объема трафика клиент-сервер, чтобы быть способными реализовать функциональные возможности PIM. Например, когда пользователь обновляет список электронной почты в ящике для приема сообщений для проверки новой электронной почты, web-запрос посылают серверу, чтобы извлечь новый список электронной почты в ящик для приема сообщений.

Для некоторых клиентских приложений трафик клиент-сервер является смесью распространяемых в Web расширений авторизации и управления версиями (Web Distributed Authoring and Versioning) (WebDAV) к протоколу передачи гипертекстовых файлов (HTTP) и других различных схем (например, на основании упрощенных пар "имя - значения" в запросах POST HTTP). Использование множества протоколов для доступа к данным в этих известных системах усложняет реализацию и способствует малой надежности при эксплуатации. Эти системы предшествующего уровня техники также используют непоследовательную и недостаточную схему для запросов. Формат имя-значение, такой как используемый в некоторых известных системах, является очень ограниченным и легко не поддерживает данные со строгим контролем типов или более сложные структуры или массивы. Кроме того, так как формат имя-значение требует (символа) конца строки в качестве разделителя для пар имя-значение, все данные для конкретной пары имя-значение должны быть закодированы без символа конца строки перед передачей и затем декодированы после приема. Эти дополнительные этапы в некоторых существующих системах замедляют обработку и увеличивают сложность. Еще одна другая проблема с некоторыми существующими клиентскими web-приложениями состоит в том, что код на стороне сервера, обеспечивающий различные функциональные возможности клиентскому приложению, является хаотическим и дезорганизованным. Добавление новых обработчиков запросов к коду на стороне сервера существующих систем является затруднительным.

Соответственно, требуется усовершенствованная система для связи клиент-сервер, чтобы разрешить один или более из этих и других недостатков.

Сущность изобретения

Варианты осуществления изобретения включают в себя протокол и схему для формирования и обмена запрошенных данных между сетевыми прикладными программами. В варианте осуществления изобретение дополнительно включает в себя архитектуру и набор интерфейсов прикладного программирования (API), доступных разработчикам обработчиков запросов, которые формируют запрошенные данные. API дают возможность разработчикам создавать сделанные на заказ обработчики запросов.

В варианте осуществления клиент - сервер по изобретению клиент согласно изобретению посылает запросы данных к серверу согласно хорошо определенной схеме изобретения. Запрос идентифицирует пространство имен обработчика запросов, обработчик событий в пределах пространства имен обработчиков запросов, и один или более параметров обработчика событий. Изобретение включает в себя инфраструктуру обработчиков событий, сгруппированных в пространство имен обработчиков запросов в сервере, который доставляет ответы клиенту во множестве форматов, как запрошено каждым клиентом. Инфраструктура сервера производит синтаксический анализ запроса, чтобы идентифицировать конкретный обработчик событий в конкретном пространстве имен и выполнить код обработчика, ассоциированный с идентифицированным обработчиком событий, как функцию параметров запроса.

Изобретение является эффективным, но легким и масштабируемым. Использование непротиворечивой схемы для запросов данных упрощает взаимодействие между прикладными программами и способствует улучшенному удобству сопровождения обработчиков запросов. Схема изобретения поддерживает сложные структуры данных. Изобретение поддерживает добавление обработчиков запросов без изменений в инфраструктуре для создания и доставки запрошенных данных.

В соответствии с одним аспектом изобретения система включает в себя один или более считываемых компьютером носителей для сохранения множества обработчиков событий, причем каждый имеет пространство имен, ассоциированное с ним. Каждый из множества обработчиков событий дополнительно имеет один или более параметров, ассоциированных с ним. Считываемые компьютером носители хранят структуру данных, которая представляет отформатированный запрос данных. Структура данных включает в себя поле пространства имен, хранящее значение пространства имен, представляющее пространство имен, поле обработчика событий, хранящее значение обработчика событий, соответствующего одному из множества обработчиков событий. Один из множества обработчиков событий ассоциирован с пространством имен, представленным значением пространства имен в поле пространства имен. Структура данных также включает в себя поле параметров, хранящее значение параметра, соответствующего одному из параметров. Этот один из параметров ассоциирован с обработчиком событий, представленным значением обработчика событий в поле обработчика событий. Система также включает в себя первый процесс, развернутый вторым процессом. Первый процесс конфигурирован так, чтобы выполнять выполняемые компьютером команды для приема запроса данных от пользователя или процесса, идентификации значения пространства имен, ассоциированного с принятым запросом, выбора значения обработчика событий как функции принятого запроса и идентифицированного значения пространства имен, и определения значения параметра как функции принятого запроса и выбранного значения обработчика событий. Первый процесс далее заполняет структуру данных посредством сохранения идентифицированного значения пространства имен в поле пространства имен, выбранного значения обработчика событий в поле обработчика событий, и определенного значения параметра в поле параметров. Первый процесс также передает заполненную структуру данных ко второму процессу. Второй процесс конфигурирован так, чтобы выполнять выполняемые компьютером команды для приема заполненной структуры данных от первого процесса; выбора, как функции принятой структуры данных, обработчика событий из множества обработчиков событий, сохраненных на считываемом компьютером носителе, выполнение

выбранного обработчика событий, чтобы сформировать данные результата; и передавать сформированные данные результата к первому процессу.

В соответствии с другим аспектом изобретения реализованный на компьютере способ формирует и доставляет данные от одного процесса к другому процессу. Осуществляемый компьютером способ включает в себя прием первым процессом запроса данных от пользователя и идентификацию пространства имен, ассоциированных с принятым запросом. Способ также включает в себя определение, посредством первого процесса, обработчика событий, ассоциированного с идентифицированным пространством имен как функции принятого запроса. Способ дополнительно включает в себя заполнение первым процессом параметра, ассоциированного с определенным обработчиком событий, как функции принятого запроса и создание отформатированного запроса с идентифицированным пространством имен, определенным обработчиком событий и заполненным параметром. Способ также включает в себя передачу из первого процесса созданного отформатированного запроса ко второму процессу. Способ также включает в себя прием вторым процессом отформатированного запроса от первого процесса и выбор, как функцию принятого отформатированного запроса, обработчика событий из множества обработчиков событий, сохраненных в области памяти. Реализованный компьютером способ также включает в себя выполнение вторым процессом выбранного обработчика событий, чтобы сформировать данные результата, и передачу сформированных данных результата от второго процесса к первому процессу.

В соответствии с другим аспектом изобретения один или более считываемых компьютером носителей имеют выполняемые компьютером компоненты для создания и передачи запрошенных данных в сетевой среде. Эти компоненты включают в себя развернутый сервером клиентский компонент для приема запроса данных от пользователя; идентификации пространства имен, ассоциированного с принятым запросом; определения обработчика событий, ассоциированного с идентифицированным пространством имен, как функции принятого запроса; заполнения параметра, ассоциированного с определенным обработчиком событий, как функции принятого запроса; создания отформатированного запроса с идентифицированным пространством имен, определенным обработчиком событий и заполненным параметром; и передачу созданного отформатированного запроса к серверному компоненту. Компоненты также включают в себя серверный компонент для приема переданного запроса от развернутого сервером клиентского компонента; выбор, как функции принятого запроса, обработчика событий из множества обработчиков событий, сохраненных в области памяти; выполнение выбранного обработчика событий, чтобы сформировать данные результата; и передачу сформированных данных результата к развернутому сервером клиентскому компоненту.

Альтернативно, изобретение может содержать различные другие способы и устройства.

Другие признаки должны быть частично очевидны и частично указаны ниже.

Краткое описание чертежей

Фиг. 1 является примерной блок-схемой, иллюстрирующей серверный компонент, обменивающийся с одним или более клиентскими компонентами через сеть.

Фиг. 2 является примерной последовательностью операций, иллюстрирующей работу клиентского и серверного процессов для передачи запрошенных данных.

Фиг. 3 является примерной блок-схемой, иллюстрирующей модули в серверном компоненте.

Фиг. 4 является примерной блок-схемой, иллюстрирующей примерный набор классов обработчика событий.

Фиг. 5 является блок-схемой, иллюстрирующей один пример подходящей среды вычислительной системы, в которой может быть реализовано изобретение.

Приложение A описывает примерные события и параметры в пространстве имен электронной почты согласно изобретению.

Соответствующие ссылочные символы указывают соответствующие части на чертежах.

Подробное описание изобретения

В варианте осуществления изобретение включает в себя серверное приложение, компонент, процесс, или подобное, доставляющее данные к развернутому сервером клиентскому приложению, компоненту, процессу, или подобному в ответ на запрос данных от клиентского приложения, например, как проиллюстрировано на фиг. 1. На серверном компоненте 102 код или подпрограмма для ответа на запрос данных, такого как запрос 103 данных, разделяется на обработчики событий (например, множество обработчиков 104 событий) или подобные. Обработчики 104 событий логически организованы в одном варианте осуществления в пространство имен. В частности, изобретение включает в себя построение запроса, такого как запрос 103 данных, который содержит информацию о пространстве имен и обработчике событий для выполнения, а также параметры для этого обработчика. Изобретение дополнительно включает в себя инфраструктуру в сервере 102, которая позволяет разработчикам реализовать обработчики запросов, которые доставляют ответы на запросы данных.

Один пример изобретения вовлекает представление календаря в распределенном приложении управления персональной информацией. Представление календаря показывает список назначений на данный день. Сервер 102 обновляет (восстанавливает) это представление в календаре посредством запроса базы данных, чтобы извлечь назначения в заданном временном диапазоне. Извлеченные назначения затем выдаются в клиенте (например, клиентском компоненте 108) в правых позициях в зависимости от начального времени и продолжительности каждого из назначений. В предшествующих системах такая операция восстановления достигалась клиентским компонентом 108, запрашивающим данные, которые должны быть возвращены в формате расширяемого языка разметки (XML), используя распространяемые в Web расширения авторизации и управления версиями (WebDAV) для протокола передачи гипертекстовых файлов (HTTP), используя преобразование расширяемого определения схемы (XSD), чтобы сформировать HTML из XML данных, и представить HTML пользователю. Такие системы являются сложными и трудными для поддержания.

Однако, согласно изобретению обновление календаря достигается клиентским компонентом 108, посылающим форматированный запрос данных (например, запрос 103 данных) и принимающим поток кода (например, JavaScript), определяющего календарные данные. Календарные данные частично воспроизводятся (визуализируются) в сервере 102. Клиентский компонент 108 выполняет код и формирует HTML-разметку, используя сценарий. Нагрузка на клиентский компонент 108 уменьшается, так как большинство данных календаря и информация отображения предварительно вычисляются в сервере 102. В другом примере изобретения сервер 102 обновляет представление почты приложения PIM

посредством возврата HTML, который был визуализирован в сервере 102. Изобретение не ограничено возвращением данных в любом конкретном формате. Скорее, изобретение возвращает данные в любой форме, продиктованной клиентским компонентом 108, такой как HTML, XML, текст, полезные данные JavaScript, или подобной.

Обращаясь снова к фиг. 1, примерная блок-схема иллюстрирует серверный компонент, процесс, или подобное, такое как серверный компонент 102, обменивающийся данными через сеть (например, Интернет) с одним или более клиентскими компонентами 108, процессами, или подобными, такими как клиентский компонент №1 - клиентский компонент № N. Система согласно фиг. 1 включает в себя один или более считываемых компьютером носителей, таких как область 106 памяти. Область 106 памяти хранит множество обработчиков 104 событий, каждый имеющий пространство имен, ассоциированное с ним. Каждый из множества обработчиков 104 событий дополнительно имеет один или более параметров, ассоциированных с ним. Структура данных, представляющая отформатированный запрос 103 данных, также сохранена на считываемом компьютером носителе. На фиг. 1 запрос данных показан посылаемым от клиентского компонента 108 к серверному компоненту 102. Структура данных, представляющая запрос 103 данных, включает в себя поле пространства имен, хранящее значение пространства имен, представляющее это пространство имен. Структура данных дополнительно включает в себя поле обработчика событий, хранящее значение обработчика событий, соответствующее одному из множества обработчиков 104 событий, сохраненных в области 106 памяти, которая ассоциирована с пространством имен, представленным значением пространства имен в поле пространства имен. Структура данных дополнительно включает в себя поле параметров, хранящее по меньшей мере одно значение параметра, соответствующее параметру, ассоциированному с обработчиком событий, представленным значением обработчика событий в поле обработчика событий.

В примере согласно фиг. 1 клиентский компонент 108 является первым процессом, а серверный компонент 102 является вторым процессом. Первый процесс является распределенным клиентским приложением или, иначе, развернутым посредством второго процесса, чтобы обеспечить некоторые или все функциональные возможности серверного компонента 102. Например, первый процесс может быть развернутым сервером клиентским приложением, таким как развернутый сервером клиент управления персональной информацией, и серверный компонент 102 может быть сервером управления персональной информацией. Однако изобретение не ограничено отношениями клиент-сервер между первым процессом и вторым процессом. Например, отношения между первым процессом и вторым процессом могут быть одноранговыми.

В одном варианте осуществления первый процесс конфигурирован так, чтобы выполнять выполняемые компьютером команды для запрашивания данных от второго процесса. В частности, первый процесс принимает запрос данных от пользователя или процесса. Запрос может включать в себя, например, запрос проверки недавно принятой электронной почты, запрос модифицировать отображение календаря, или любой другой запрос данных. Первый процесс идентифицирует значение пространства имен, ассоциированное с принятым запросом, и выбирает значение обработчика событий как функцию принятого запроса и идентифицированного значения пространства имен. Выбранный обработчик событий является конкретным для определенного запроса от пользователя и является одним из

множества обработчиков 104 событий, ассоциированных с идентифицированным значением пространства имен. Выбранный обработчик событий имеет один или более параметров. Первый процесс определяет значение параметра для одного или более параметров выбранного обработчика событий. Значение параметра определяется, используя данные из принятого запроса. Значение параметра может включать в себя данные любого типа и может быть воплощено в любом формате, распознаваемым вторым процессом (например, на расширяемом языке разметки). Первый процесс заполняет структуру данных, представляющую запрос 103 данных, посредством сохранения идентифицированного значения пространства имен в поле пространства имен, выбранное значение обработчика событий - в поле обработчика событий, и определенное значение параметра - в поле параметров. Первый процесс передает заполненную структуру данных (например, запрос 103 данных) ко второму процессу.

В одном варианте осуществления заполненная структура данных включает в себя ряд пар имя-значение. Например, поле пространства имен и значение пространства имен являются парой имя-значение, поле обработчика событий и значение обработчика событий являются парой имя-значение, и поле параметра и значение параметра являются парой имя-значение.

Второй процесс конфигурирован так, чтобы выполнять выполняемые компьютером команды для формирования и доставки запрошенных данных к первому процессу. В одном варианте осуществления второй процесс принимает заполненную структуру данных (например, запрос 103 данных) от первого процесса и выбирает обработчик событий из множества обработчиков 104 событий, сохраненных в области 106 памяти. Выбранный обработчик событий соответствует обработчику событий, указанному в принятом запросе 103 данных. Второй процесс выполняет выбранный обработчик событий, используя параметры из запроса 103 данных, принятого от первого процесса, для формирования данных результата. Второй процесс передает сформированные данные результата к первому процессу.

В одном варианте осуществления второй процесс выбирает подходящий обработчик событий, обращаясь к базе данных конфигураций, заполненной информацией относительно доступных пространств имен, обработчиков событий (например, множестве обработчиков 104 событий) и параметров. Второй процесс включает в себя, например, компонент регистрации для построения базы данных конфигурации до приема запроса 103 данных от первого процесса. Например, множество обработчиков 104 событий может быть организовано в базе данных конфигурации по пространству имен.

Изобретение включает в себя средство для сохранения множества обработчиков 104 событий, средство для выполнения первого процесса и средство для выполнения второго процесса. Аппаратное и программное обеспечение, такое как структуры данных, интерфейс пользователя, прикладная программа, интерфейс прикладного программирования (API), выполняемые компьютером команды, программно-аппаратные средства, и т.п. (такие как иллюстрируются на чертежах) составляют средства для сохранения множества обработчиков 104 событий, средство для выполнения первого процесса и средство для выполнения второго процесса.

В одном варианте осуществления изобретение реализуется, используя HTTP-запросы с глаголом POST (отправить почтовое сообщение) или GET (получить). Запрос включает в себя следующую информацию: пространство имен (например, обозначенный в строке запроса URL строковый параметр 'ns'), название события (например, обозначенное в строке запроса URL как параметр 'ev'), и один или более

параметров запроса. Параметры представлены или в XML для запросов POST или как пары имя-значение - для запросов GET.

Примерный запрос GET (например, запрос 103 данных) согласно изобретению является следующим:

URL: /mail/ev.owa?ns=CalendarView&ev=Refresh&d=l 1142004T00:00:00Z

В этом примере клиент (например, клиентский компонент 108) зарашивает сервер (например, серверный компонент 102), чтобы выполнить обработчик "Refresh" в пространстве имен "CalendarView". Клиентский компонент 108 передает параметр даты типа, названной "d" с значением 11/14/2004 в 12 часов. Серверный компонент 102 возвращает полезные данные разметки или JavaScript для этого представления, в зависимости от реализации.

Примерный запрос POST (например, запрос 103 данных) согласно изобретению является следующим:

URL: /mailserver/ev.owa?ns=Messaging&ev=SaveMessage

BODY:

<params>

<subject>Testemail</subject><to><item>userA@pageA.net</item><item>userB@pageA.net</item><to>

<body>Test something < /body > < /params >

В этом примере клиентский компонент 108 просит серверный компонент 102 выполнить обработчик 'SaveMessage' событий в пространстве имен 'Messaging'. Параметрами события являются subject (строка), к (массив строк) и body (строка). Ответом для этого события является результат HTTP, указывающий успех или ошибку. В этом случае никакая возвращенная разметка не является необходимой. Параметры могут быть в форме XML или формате любого другого языка, понятного серверу.

Обращаясь к фиг. 2, примерная последовательность операций иллюстрирует работу клиента и процессы сервера для передачи запрошенных данных. Последовательность операций согласно фиг. 2 отражает реализуемый компьютером способ создания и доставки данных от одного процесса к другому процессу. Способ включает в себя прием первым процессом, таким как клиентский компонент 108 на фиг. 1, запроса данных от пользователя на этапе 202 и идентификацию пространства имен, ассоциированного с принятым запросом, на этапе 204. Способ дополнительно включает в себя определение обработчика событий, ассоциированного с идентифицированным пространством имен, как функцию (в ответ на) принятого запроса на этапе 206 и заполнение параметра, ассоциированного с определенным обработчиком событий, как функцию принятого запроса на этапе 208. Способ также включает в себя создание форматированного запроса (например, запрос 103 данных на фиг. 1) с идентифицированным пространством имен, определенным обработчиком событий и заполненным параметром на этапе 210. Согласно способу передают от первого процесса ко второму процессу (например, серверному компоненту 102) созданный отформатированный запрос на этапе 212.

Способ согласно изобретению включает в себя прием вторым процессом отформатированного запроса от первого процесса на этапе 214. Согласно способу выбирают пространство имен на этапе 215 на основании принятого запроса. Способ также включает в себя выбор вторым процессом, как функции принятого отформатированного запроса, обработчика событий из множества обработчиков событий (например, множества обработчиков 104 событий на фиг. 1), сохраненных на одном или более считываемых компьютером носителях (например, области 106

памяти на фиг. 1) на этапе 216. Согласно способу синтаксически анализируют и проверяют правильность параметров в принятом запросе на этапе 217. Способ дополнительно включает в себя выполнение выбранного обработчика событий, чтобы сформировать данные результата на этапе 218 и передачу от второго процесса к первому процессу сформированных данных результата к первому процессу на этапе 220. Первый процесс принимает данные результата и, в одном примере, отображает данные пользователю.

Разработчик или другой пользователь могут создавать определяемые пользователем пространства имен, обработчик событий и/или параметры. Способ согласно изобретению дополнительно включает в себя прием первым процессом от пользователя определяемого пользователем пространства имен, определяемого пользователем обработчика событий, ассоциированного с определяемым пользователем пространством имен, и/или определяемого пользователем параметра определяемого пользователем обработчика событий. Первый процесс регистрирует принятое определяемое пользователем пространство имен, определяемый пользователем обработчик событий и/или определяемый пользователем параметр во втором процессе. Второй процесс регистрирует принятое определяемое пользователем пространство имен, определяемый пользователем обработчик событий и определяемый пользователем параметр без изменения инфраструктуры второго процесса. То есть код, осуществляющий второй процесс (например, серверный компонент), не изменяется из-за регистрации. Инфраструктура второго процесса обращается к зарегистрированному определяемому пользователем пространству имен, обработчику событий, или параметру тем же самым способом, каким второй процесс обращается к другим пространствам имен, обработчикам события и значениям параметра.

В одном варианте осуществления один или более считываемых компьютером носителей имеют выполняемые компьютером команды для выполнения реализуемого компьютером способа, проиллюстрированного на фиг. 2.

Обращаясь к фиг. 3, примерная блок-схема иллюстрирует модули в серверном компоненте. Модули вызываются в пределах потока 302 запросов, порожденного серверным компонентом. В одном варианте осуществления обработчиками событий являются обработчики HTTP, осуществляющие интерфейс IHttpHandler. Изобретение вызывает метод ProcessRequest в этом интерфейсе, который позволяет классу обработчиков выполнять код, чтобы обработать запрос. Фабрика обработчиков HTTP является классом, который осуществляет интерфейс IHttpHandlerFactory. Такие объекты используются, чтобы создать экземпляры объектов обработчика HTTP, используемого для обработки запросов. В примере на фиг. 3 показывается класс фабрики обработчика HTTP по имени EventHandlerFactory 304. Изобретение (например, HTTP приложение 306) инициализирует объект (например, экземпляр 308 EventHandlerBase) этого класса и вызывает метод "GetHandler" интерфейса IHttpHandlerFactory, который создает экземпляр подходящего класса обработчика HTTP (например, подклассы EventHandlerBase) на основании пространства имен запроса. HTTP приложение 306 и HTTP приложение 306 EventHandlerBase обращаются к HTTP-контексту 314.

Объект EventContext 312 является структурой данных, которая содержит информацию о запросе (например, пространстве имен, имени события и параметрах запроса). EventHandlerRegistry 310 является классом, который служит в качестве архива информации о пространствах имен и их обработчиков событий, реализованных

приложением, а также сделанных на заказ поддерживаемых структурах данных и перечислениях. Приложение регистрирует класс EventHandlerBase, используя этот архив 310 при запуске приложения. Изобретение сканирует класс (например, используя отражение), чтобы определить пространство имен, события в пространстве имен и параметры событий. Эта информация сохраняется и используется позже в течение обработки запроса и синтаксического анализа параметров.

Примерная последовательность операций выполнения запроса согласно изобретению проиллюстрирована на фиг. 3. Согласно изобретению принимают запрос данных в соответствии с изобретением, создают экземпляр класса EventHandlerFactory и вызывают (метод) GetHandler в отношении созданного экземпляра. Реализация GetHandler для EventHandlerFactory просматривает строку запроса, чтобы вычислить названия пространства имен и события, и исследует EventHandlerRegistry 310 на класс обработчика, соответствующего названиям пространства имен и события в строке запроса. Если согласно изобретению находят запрошенный обработчик событий, изобретение создает экземпляр этого обработчика событий. EventContext 312 также создается. Изобретение вызывает ProcessRequest в отношении обработчика событий. ProcessRequest проводит синтаксический анализ параметров запроса, используя класс синтаксического анализатора, специфический для самого запроса (например, является ли запрос глаголом GET или глаголом POST). Класс синтаксических анализов анализирует параметры в запросе и использует EventHandlerRegistry 310, чтобы сделать любые преобразования типов и гарантировать, что схема является корректной (например, гарантировать, что все параметры, требуемые этим обработчиком, установлены). Параметры могут быть сохранены в коллекции в контексте в этот момент. Согласно изобретению выполняют код обработчика событий, ассоциированного с обработчиком событий (например, выполняют метод в классе обработчика). Этот код обращается к параметрам в контексте, выполняет обработку и записывает ответ на запрашивающий объект в любом формате, указанном запрашивающим объектом.

Со ссылками на фиг. 4 примерная блок-схема иллюстрирует типовой набор классов обработчика событий, наследуемых от EventHandlerBase 402. EventHandlerBase 402 реализует интерфейс 404 IHttpHandler. CalendarEventHandler 406, MessagingEventHandler 408 и MessageViewEventHandler 410 наследуются из EventHandlerBase 402. Разработчики или другие пользователи могут создавать созданный на заказ обработчик событий, такой как показан на фиг. 4, посредством создания пространства имен, идентифицируя событие в пространстве имен и определяя параметры для события. Определяемое пользователем пространство имен, обработчик событий и/или параметр могут быть выбранными пользователем (например, новый обработчик событий в пределах существующего пространства имен) или разработаны пользователем (например, новое пространство имен). В одном варианте осуществления определяемый пользователем обработчик событий реализован как динамически загружаемая библиотека.

Разработчики или другие пользователи могут определять схему пространства имен, используя атрибуты так, как показано ниже.

```
[EventNamespace ("Namespase1")]
internal classc TestEventHandler: EventHandlerBase
{
[Event("Event1")]
[EventParameter ("Param1", typeof (string))]
```

```
[EventParameter ("Param2", typeof (int))]
public void Event1 ()
{
string param1=
(string)Context.GetParamValue("Param1");
int param2=(int)Context.GetParamValue ("Param2");
Writer.Write("{0}+{1}={2}", param1, param2, param1 +param2).
}
}
```

В этом примере атрибут EventNamespace определяет название (имя) пространства имен на соединении, атрибут Event определяет название события на соединении, и атрибут EventParameter определяет имя параметра, тип и является ли параметр необязательным или нет. Чтобы использовать этот обработчик, разработчик обработчика записывает класс, который наследуется от EventHandlerBase 402, записывает события как методы, которые не имеют каких-либо параметров и возвращают результат void (пусто), определяет схему, используя соответствующие атрибуты, описанные выше, и регистрирует класс в серверном компоненте.

Структуры данных по изобретению позволяют разработчикам определять сложные типы, которые могут использоваться в качестве параметров. Примерная структура данных показывается ниже.

```
[EventStruct ("r")]
internal classc RecipientWellltem
{
[EventField ("dn")]
public string DisplayName;
[EventField ("em", true, "oof@pageA.net")]//true= необязательное поле
public string Email;
[EventField ("t")]
public int Type;
internal RecipientWelirtem ()
{
DisplayName=" ";
Email=" ";
Type = 0;
}
}
```

Примерная структура данных, указанная выше, закодирована так, как показано ниже.

```
< r dn = "display name " em = "email address" t = "1" / >
```

Когда серверный компонент согласно изобретению принимает запрос, который соответствует конкретной структуре данных, изобретение создает экземпляр этого конкретного класса структуры данных и заполняет его данными из запроса.

Примерная рабочая среда

Фиг. 5 иллюстрирует один пример вычислительного устройства общего назначения в форме компьютера 130. В одном варианте осуществления изобретения компьютер, такой как компьютер 130, подходит для использования в других чертежах, проиллюстрированных и описанных в настоящем описании. Компьютер 130 имеет один или более процессоров или устройств 132 обработки и системную память 134. В

иллюстрированном варианте осуществления системная шина 136 подсоединяет различные элементы системы, включая системную память 134, к процессорам 132. Шина 136 представляет одну или более из любого из нескольких типов шинных структур, включающих в себя шину памяти или контроллер памяти, шину периферийных устройств, ускоренный графический порт и процессорную или локальную шину, используя любую из разнообразия шинных архитектур. В качестве примера и не ограничения, такая архитектура включает в себя шину архитектуры, соответствующей промышленному стандарту (ISA), шину микроканальной архитектуры (MCA), расширенную шину ISA (EISA), локальную шину Ассоциации по стандартам в области видеоэлектроники (VESA) и шину соединения периферийных компонентов (PCI), также известную как шина Mezzanine.

Компьютер 130 в обычном варианте имеет по меньшей мере некоторую форму считываемых компьютером носителей. Считываемые компьютерные носители, которые включают в себя и энергозависимые и энергонезависимые носители, сменные и несменные носители, могут быть любой доступной средой, к которой можно обращаться компьютером 130. В качестве примера и не ограничения, считываемые компьютерные носители содержат компьютерные запоминающие устройства и коммуникационные носители. Компьютерные запоминающие устройства включают в себя энергозависимые и энергонезависимые, сменные и несменные носители, осуществленные любым способом или технологией для хранения информации, такой как считываемых компьютером команд, структур данных, программных модулей или других данных. Например, компьютерные запоминающие устройства включают в себя ОЗУ, ПЗУ, СППЗУ, флэш-память или память по другой технологии, CD-ROM, цифровые универсальные диски (DVD) или другую оптическую память на дисках, магнитных кассетах, магнитной ленте, память на магнитном диске или другие магнитные запоминающие устройства, или любую другую среду, которая может использоваться, чтобы хранить желательную информацию, и к которой можно обращаться компьютером 130. Коммуникационные носители обычно воплощают считываемые компьютером команды, структуры данных, программные модули, или другие данные в модулируемом сигнале данных, таком как несущая, или другом транспортном механизме, и включают в себя любые носители доставки информации. Специалисты знакомы с модулируемым сигналом данных, который имеет одну или более из его набора характеристик, установленную или измененную таким образом, чтобы кодировать информацию в сигнале. Проводные носители, такие как проводные сети или непосредственное проводное соединение, и беспроводные носители, такие как акустический, РЧ, инфракрасное излучение, и другие беспроводные носители, являются примерами коммуникационных носителей. Комбинации любого из вышеупомянутых также включено в объем понятия считываемых компьютером носителей.

Системная память 134 включает в себя компьютерные запоминающие устройства в форме сменной и/или несменной, энергозависимой и/или энергонезависимой памяти. В иллюстрированном варианте осуществления системная память 134 включает в себя постоянное запоминающее устройство (ПЗУ) 138 и оперативное запоминающее устройство (ОЗУ) 140. Базовая система ввода/вывода 142 (BIOS), содержащая основные подпрограммы, которые помогают передавать информацию между элементами в компьютере 130, например, во время запуска, обычно сохраняется в ПЗУ 138. ОЗУ 140 обычно содержит данные и/или программные модули, которые являются немедленно доступными для и/или над которыми в настоящее время

работают посредством процессора 132. В качестве примера и не ограничения фиг. 5 иллюстрирует операционную систему 144, прикладные программы 146, другие программные модули 148 и программные данные 150.

Компьютер 130 может также включать в себя другие сменные / несменные, энергозависимые / энергонезависимые компьютерные запоминающие устройства. Например, фиг. 5 иллюстрирует накопитель 154 на жестком диске, который считывает с или записывает на несменный энергонезависимый магнитный носитель. Фиг. 5 также изображает накопитель 156 на магнитном диске, который считывает с или записывает на сменный энергонезависимый магнитный диск 158, и накопитель 160 на оптических дисках, который считывает с или записывает на сменный энергонезависимый оптический диск 162, такой как CD-ROM или другие оптические носители. Другие сменные / несменные, энергозависимые / энергонезависимые компьютерные запоминающие устройства, которые могут использоваться в примерной среде, включают в себя, но не ограничиваются ими, кассеты магнитной ленты, платы флэш-памяти, цифровые универсальные диски, цифровую видеоленту, твердотельное ОЗУ, твердотельное ПЗУ и т.п. Накопитель 154 на жестком диске и накопитель 156 на магнитном диске и накопитель 160 на оптических дисках обычно подсоединяются к системной шине 136 посредством интерфейса энергонезависимых ЗУ, такого как интерфейс 166.

Накопители или другие устройства памяти большой емкости и их ассоциированные компьютерные запоминающие устройства, описанные выше и проиллюстрированные на фиг. 5, обеспечивают сохранение считываемых компьютером команд, структур данных, программных модулей и других данных для компьютера 130. На фиг. 5, например, накопитель 154 на жестком диске иллюстрируется как сохраняющий операционную систему 170, прикладные программы 172, другие программные модули 174 и программные данные 176. Следует заметить, что эти компоненты могут или быть одинаковыми или отличными от операционной системы 144, прикладных программ 146, других программных модулей 148 и программных данных 150. Операционной системе 170, прикладным программам 172, другим программным модулям 174 и программным данным 176 здесь даны отличающиеся номера, чтобы иллюстрировать, что, как минимум, они являются различными копиями.

Пользователь может вводить команды и информацию в компьютер 130 посредством устройств ввода или устройств выбора пользовательского интерфейса, таких как клавиатура 180 и устройство 182 управления позицией (например, мышь, шаровой указатель, перо, сенсорная площадка). Другие устройства ввода (не показаны) могут включать в себя микрофон, джойстик, игровую клавиатуру, спутниковую антенну, сканер или подобные. Эти и другие устройства ввода соединены с процессором 132 через пользовательский интерфейс 184 устройств ввода, который подсоединен к системной шине 136, но может быть соединен другим интерфейсом и шинными структурами, такими как параллельный порт, игровой порт, или универсальная последовательная шина (USB). Монитор 188 или другой тип устройства отображения также соединен с системной шиной 136 через интерфейс, такой как видеоинтерфейс 190. В дополнение к монитору 188 компьютеры часто включают в себя другие периферийные устройства вывода (не показаны), например, принтер и громкоговорители, которые могут быть соединены через интерфейс периферийных устройств вывода (не показаны).

Компьютер 130 может работать в сетевой среде, используя логические соединения с одним или более удаленными компьютерами, такими как удаленный компьютер 194.

Удаленный компьютер 194 может быть персональным компьютером, сервером, маршрутизатором, сетевым персональным компьютером, одноранговым устройством или другим общим сетевым узлом, и обычно включает в себя многие или все элементы, описанные выше относительно компьютера 130. Логические соединения, изображенные на фиг. 5, включают в себя локальную сеть (ЛС, LAN) 196 и глобальную сеть (ГС, WAN) 198, но могут также включать в себя другие сети. LAN 136 и/или WAN 138 может быть проводной сетью, беспроводной сетью, их комбинацией и так далее. Такие сетевые среды являются обычными в офисах, компьютерных сетях в масштабах предприятия, интранет и глобальных компьютерных сетях (например, Интернет).

При использовании в локальной сетевой среде компьютер 130 соединен с LAN 196 через сетевой интерфейс или адаптер 186. Когда используется в глобальной сети, компьютер 130 обычно включает в себя модем 178 или другие средства для установления связи по WAN 198, например, Интернет. Модем 178, который может быть внутренним или внешним, связан с системной шиной 136 через интерфейс 184 ввода пользователя или другой соответствующий механизм. В сетевой среде программные модули, изображенные относительно компьютера 130 или его частей, могут быть сохранены в удаленном запоминающем устройстве (не показано). В качестве примера, а не ограничения, фиг. 5 иллюстрирует удаленные прикладные программы 192 как постоянно находящиеся на ЗУ. Показанные сетевые соединения являются примерными, и могут использоваться другие средства установления линии связи между компьютерами.

Обычно процессоры данных компьютера 130 запрограммированы посредством команд, сохраненных в разное время в различных считываемых компьютером компьютерных носителях данных. Программы и операционные системы обычно распространяются, например, на гибких дисках или CD-ROM. С них они устанавливаются или загружаются во вторичную память на компьютере. При выполнении они загружаются, по меньшей мере частично, в первичную электронную память компьютера. Изобретение, описанное здесь, включает в себя эти и другие различные типы считываемых компьютером носителей данных, когда такие носители содержат команды или программы для осуществления этапов, описанных ниже совместно с микропроцессором или другим процессором. Изобретение также включает в себя сам компьютер, когда он запрограммирован согласно способам и методике, описанной здесь.

С целью иллюстрации программы и другие выполняемые программные компоненты, такие как операционная система, проиллюстрированы здесь как дискретные блоки. Понятно, однако, что такие программы и компоненты постоянно находятся в различное время в различных компонентах памяти компьютера, и выполняются процессорами обработки данных компьютера.

Хотя описано со ссылками на примерную вычислительную среду системы, включающую в себя компьютер 130, изобретение работоспособно с многочисленными другими средами или конфигурациями вычислительных систем общего назначения или специализированных. Среда вычислительной системы не предназначена, чтобы предложить какое-либо ограничение относительно объема использования или функциональных возможностей изобретения. Кроме того, среда вычислительной системы не должна интерпретироваться как имеющая какую-либо зависимость или требование, относящееся к любому из или комбинации компонентов, иллюстрированных в примерной среде. Примеры известных вычислительных систем,

сред и/или конфигураций, которые могут быть подходящими для использования с изобретением, включают в себя, но не ограничиваются ими, персональные компьютеры, серверные компьютеры, карманные или портативные компьютерные устройства, мультипроцессорные системы, основанные на микропроцессорах системы, телевизионные приставки, программируемую бытовую электронику, мобильные телефоны, сетевые персональные компьютеры, мини компьютеры, универсальные компьютеры, распределенные вычислительные среды, которые включают в себя любую из вышеупомянутых систем или устройств, и т.п.

Изобретение может быть описано в общем контексте выполняемых компьютером команд, таких как программные модули, выполняемые одним или более компьютером или другими устройствами. Обычно программные модули включают в себя, но не ограничиваются ими, подпрограммы, программы, объекты, компоненты и структуры данных, которые выполняют конкретные задачи или реализуют конкретные абстрактные типы данных. Изобретение может также быть осуществлено в распределенных вычислительных средах, где задачи выполняются удаленными устройствами обработки, которые соединены через коммуникационную систему. В распределенной вычислительной среде программные модули могут быть расположены и в локальных и в удаленных компьютерных запоминающих носителях, включая в себя запоминающие устройства памяти.

Интерфейс в контексте программной архитектуры включает в себя модуль программного обеспечения, компонент, часть кода, или другую последовательность выполняемых компьютером команд. Интерфейс включает в себя, например, первый модуль, обращающийся ко второму модулю, чтобы выполнить вычислительные задачи от имени первого модуля. Первый и второй модули включают в себя, в одном примере, интерфейсы прикладного программирования (API), такие как те, что обеспечиваются операционными системами, интерфейсами модели компонентных объектов (COM) (например, для однорангового обмена приложений), и интерфейсами формата обмена метаданными расширяемого языка разметки (XMI) (например, для обмена между Web-услугами).

Интерфейс может быть сильно связанной синхронной реализацией, например, в примерах Java 2 Platform Enterprise Edition (J2EE), COM, или распределенной COM (DCOM). Альтернативно или в дополнение, интерфейс может быть слабосвязанной, асинхронной реализацией, таким как в Web-услуге (например, используя простой протокол доступа к объектам). Вообще, интерфейс включает в себя любую комбинацию следующих характеристик: с сильной связью, со слабой связью, синхронный и асинхронный. Дополнительно, интерфейс может соответствовать стандартному протоколу, частному протоколу, или любой комбинации стандартных и частных протоколов.

Интерфейсы, описанные здесь, все могут быть частью единственного интерфейса или могут быть осуществлены как отдельные интерфейсы или любой их комбинацией. Интерфейсы могут выполняться локально или удаленно, чтобы обеспечивать функциональные возможности. Кроме того, интерфейсы могут включать в себя дополнительные или меньшее количество функциональных возможностей, чем проиллюстрировано или описано в настоящем описании.

Во время работы компьютер 130 выполняет выполняемые компьютером команды, такие как те, что иллюстрируются на фиг. 2, чтобы сформировать и передать запрошенные данные между сетевыми прикладными программами.

Приложение A перечисляет примерные пространства имен, обработчики событий и

параметры согласно изобретению. В Приложении А тип обработчика DatePickerEventHandler включает в себя события для средства выбора даты. Средство выбора даты является компонентом пользовательского интерфейса, который позволяет пользователю выбирать даты в нескольких контекстах в приложении. Событие RenderMonth воспроизводит фрагмент HTML, соответствующий месяцу. Когда пользователь выполняет щелчок, чтобы переместиться к январю 2005, клиентский компонент изобретения посылает событие RenderMonth к серверу, который принимает входные параметры (например, 'm' для месяца, который является датой январь 2005) и воспроизводит фрагмент HTML для этого месяца. Воспроизведенный фрагмент HTML посылают назад клиенту.

Тип обработчика MessageListViewEventHandler включает в себя события для представления списка сообщений, которое показывает сообщения, например, в ящике для приема сообщений. Событие Delete (Удалить) выполняется, когда пользователь выбирает несколько сообщений для стирания. Идентификатор папки представлен атрибутом 'fId'. Массивы идентификаторов сообщений для удаления представлены атрибутом 'id'.

RecipientWellEventHandler является обработчиком событий для контейнеров получателя. Примером события в этом типе обработчика является 'resolve' (разрешение). 'Resolve' принимает поля to, cc, и bcc в сообщении электронной почты в качестве параметров, которые являются массивами строк. Событие Resolve производит синтаксический анализ и производит разрешение этих строк. Например, если пользователь напечатает "BobS", запрос данных посылается к серверу с единственным элементом в массиве, соответствующим параметру "to": "BobS". Ответом является HTML-фрагмент с "Bob Smith" (Боб Смит) в HTML с адресом электронной почты Боба Смита и, возможно, другими данными, присоединенными к нему.

Подробное объяснение метода GetViewPayload в обработчике события представления календаря описано ниже. Параметры для этого метода включают в себя атрибуты 'nvs', 'fId', 'days', 'vt' и 'dir'. Этот обработчик событий вызывается с клиента, когда представление календаря должно быть обновлено. Представление календаря отображает полосу времени слева дисплея с приращением в полчаса, и каждый день отображается в столбце.

Когда представление календаря должно быть обновлено, возбуждается событие GetViewPayload. Это событие загружает данные из выходного буфера (например, делает запрос в календаре пользователя в почтовом ящике) и выполняет код (например, JavaScript), который описывает эти данные. Выполняемый код называется полезными данными. Полезные данные посылают клиенту, который перерисовывает представление, используя данные полезных данных. Примерные полезные данные включает в себя время начала/конца каждого назначения, субъект, местоположение, состояние свободен/занят и так далее. Также, координаты каждого прямоугольника вычисляют в сервере и посылают как часть этих полезных данных, чтобы упростить клиентский дополнительный сценарий.

Параметры для метода GetViewPayload включают в себя атрибуты 'nvs', 'fId', 'days', 'vt' и 'dir'. Атрибутом 'nvs' является атрибутом "нет состояния представления", который, если равен значению "истина", указывает, что не имеется необходимости сохранять состояние представления. Состоянием представления является набор параметров, который определяет представление (например, количество просматриваемых дней и тип представления). Типом представления может быть

еженедельно или ежедневно. Если параметр 'nvs' равен значению "ложь", состояние представления сохраняется до следующего раза, когда пользователь войдет в систему. Атрибут 'fId' является атрибутом "идентификатор папки", который соответствует идентификатору папки календаря для просмотра. Атрибут 'days' включает в себя массивы дат для просмотра. Атрибут символа 'vt' соответствует "типу представления" ежедневно или еженедельно. Атрибут 'dir' является необязательно указанным "направлением" навигационных стрелок. Установка этого атрибута означает, что пользователь нажал навигационные стрелки, и сервер должен вычислить следующий набор дат для просмотра на основании атрибута 'days'. Например, если атрибут 'days' равен 2-ое, 3-е и 4-е марта и атрибут 'dir' установлен равным +1, сервер добавит 7 дней к каждому из них (например, многодневные представления продвигаются с приращениями 7 дней). Если атрибут 'days' равен только 2 марта (например, представление единственного дня), нажатие на правую стрелку приводит к перемещению к 3 марта, так что сервер только добавляет 1 день.

Примерный запрос к методу GetViewPayload показан ниже.

POST/mailclien1/ev.owa?oeh=1&ns=CalendarView&ev=GetViewPayload

HTTP /1.1

Body:

<params><fId>LgAAAABjczgvWhhRQKW7OM2Ok4GDAQD4If+Jk6V9SaZ9+ rfNa9e8AAAAABedAAAB</fId><days><item>2004-12-15T00:00:00</item> <item>2004-12 - 16T00:00:00 </item>< item > 2004-12-17T00: 00:00 </ item > </days><vt> 1 </vt > <nvs> 1 </nvs> < / params >

Часть примерного ответа на вышеупомянутый запрос показывается ниже.

a_cERow=1; a_sFTtl="Calendar:December 2004"; a_iNRP=1; a_rgCal=new Array(nc("RgAAAABjctheWhhRQKW7OM2Ok4GDBwD4If+Jk6V9Saz9+rf Na9e8AAAAABedAAD4If+Jk6V9Saz9+rfNa9e8AAAAADTfAAAI","12-15-2004 17:30:00 U "5" 12-25-200418:00:00 U ","Strategy Meeting","Conf Room 1",2,1, Bob Smith",0,10,0,""), nc("CAjGsFAnk8AARgAAAABjczgtheh RQKW7OM2Ok4GDBwD4If+Jk6V9Saz9+ rfNa9e8AAAAAB[beta]dAAD4If+Jk6V9Saz9+rfNa9e8AAAAACNqAAAJ "," 12-15-2004 21: 00:00 U "," 12-15-2004 21:30:00U ","Out of Office","",2,0,"",1,3,1,0,""), nc("RgAAAABjczgvWhhRQKW7 OM2Ok4GDBwD4If+Jk6V9Saz9+ rfNa9e8AAAAABedAAD4If+Jk6V9Saz9+rfNa9e8AAAAADThAAAF, " 12-16-2004 00:00:00 U "," 12- 16-2004 01:00:00 U'V'Portfolio Development Meeting "," Conf Room 2 ", 2,1, "JoeA.", 1,1,0,0,"")); a_rgDay=new Array(); vart;t=newCalDay("l5", "Wednesday",l5,l2,2004); av(t,05l9,l,l,0);av(t,0,24,0.5,4,1);av(t,0.5,26,0.5,1,2);av(t,0,32,1,2,3);av(t,0,34,1,2,4); a_rgDay[0]=t;t=newCaIDay("16","Thursday",16,12,2004);av(t,0,20,1,1,5);av(t,0,21,1,2,6); av(t,0,23,1,2,7);av(t,0,25,1,1,8);av(t,0,28,0.5,6,10);av(t,0.5,28,0.5,4,9);av(t,0.5,33,0.5,3,11); a_rgDay[1]=t;t=newCalDay("17","Friday",17,12,2004);av(t,0,20,0.5,4,13);av(t,0.5,20,0.5,2,12); av(t,0.5,23,0.5,3,14);av(t,0;25,0.5,1,15);av(t,0,28,0.5,2,17);av(t,0.5,29,0,5,348); avf(t,26,22,3);a_rgDay[2]=t;a_rgEvr=new Array (ne(2,0,1,16,2,0))

Порядок выполнения или производительность методов, проиллюстрированных и описанных здесь, не является существенным, если иначе не определено. То есть, элементы методов могут быть выполнены в любом порядке, если иначе не определено, и что эти методы могут включать в себя большее или меньшее количество элементов, чем раскрыто здесь. Например, рассматривается, что исполнение или выполнение конкретного элемента прежде, одновременно с, или после другого элемента, находится в пределах изобретения.

При представлении элементов настоящего изобретения или вариантов его

осуществления, артикли "a", "an", "the" и термин "упомянутый" предназначены, чтобы подразумевать, что имеется один или более элементов. Термины "содержащий", "включающий" и "имеющий" означают включение и подразумевают, что могут существовать дополнительные элементы, другие чем перечисленные элементы.

Ввиду вышеизложенного следует заметить, что достигнуты несколько задач изобретения и другие выгодные результаты достигнуты.

Поскольку различные изменения могут быть сделаны в вышеупомянутых конструкциях, продуктах и способах без отрыва от объема изобретения, подразумевается, что все содержание, содержащееся в вышеупомянутом описании и показанное на сопроводительных чертежах, должно интерпретироваться как иллюстративное, а не в смысле ограничения.

ПРИЛОЖЕНИЕ А

Примерные события и параметры в пространстве имен электронной почты (например, Mail.Clients) для развернутой сервером прикладной программы электронной почты показываются в таблицах ниже.

Таблица Al
Примерные Обработчики событий и параметры для обработчика событий Средство выбора даты.

Тип Обработчика	'Mail.Clients.DatePickerEventHandler'
Имя обработчика событий	'RenderMonth'. Глагол: 'Get'
Имя параметра	'uF', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'm', тип: 'System.DateTime', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'GetFreeBusy'. Глагол: 'Get'
Имя параметра	'Month', тип: 'System.DateTime', isArray: 'False', isOptional: 'False', isStruct: 'False'

Таблица A2
Примерные обработчики событий и параметры для обработчика событий представления списка сообщений.

Тип обработчика	'Mail.Clients.MessageListViewEventHandler'
Имя обработчика событий	'Delete'. Глагол: 'Post'
Имя параметра	'fId', тип: 'MaILObjects-IuniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'id', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'PermanentDelete'. Глагол: 'Post'
Имя параметра	'id', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct 'False'
Имя обработчика событий	'MarkAsRead'. Глагол: 'Post'
Имя параметра	'id', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'MarkAsUnread'. Глагол: 'Post'
Имя параметра	'id', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False',
	isStruct: 'False'
Имя обработчика событий	'PersistWidth'. Глагол: 'Post'
Имя параметра	'fId', тип: 'MailObjects-IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'w', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'PersistHeight'. Глагол: 'Post'
Имя параметра	'fId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'h', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'PersistReadingPane'. Глагол: 'Post'
Имя параметра	's', тип: 'Mail.Clients.Common.Reading anePosition', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'fId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'Refresh'. Глагол: 'Get'
Имя параметра	'sId', тип: 'Mail.Objects.IUniqueItemId',
	isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO', тип: 'Mail.Objects.SortOrder', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sC', тип: 'Mail.Clients.Controls.ColumnId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'mL', тип: 'System.Boolean', isArray: 'False', isOptional: 'False', isStruct: 'False'

Имя параметра	'sR', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct 'False'
Имя обработчика событий	'SeekNextItem'. Глагол: 'Get'
Имя параметра	'sR', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'mL', тип: 'SystemJBoolean', isArray: 'False', isOptional: 'False', isStruct 'False'
Имя параметра	'sId', тип: 'Mail.Objects.Iuniqueltemld', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'mId', тип: 'Mail.Objects. IUniqueltemld ', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sC', тип: 'Mail.CIients.Controls.Columnld', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO, тип: 'Mail.Objects.SortOrder', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'SeekPreviousItem'. Глагол: 'Get'
Имя параметра	'mL', тип: 'System.Boolean', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sId', тип: 'Mail.Objects.IuniqueltemId', isArray: 'False', isOptional: 'False', isStruct 'False'
Имя параметра	'mId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct 'False'
Имя параметра	'sR', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sC',тип: Mail.Clients.Controls.Colimxtild', IsArray:
	'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO', тип: 'Mail.Objects.SortOrder', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'ToggleMultiVsSingleLine'. Глагол: 'Get'
Имя параметра	'sId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'mL', тип: 'System. Booleari, isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sR', тип: 'System.Int32', isAxray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sC', тип: 'Mail.Clients.Controls. Colunmld1, isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO', тип: 'Mail.Objects. SortOrder1, isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'FirstPage'. Глагол: 'Get'
Имя параметра	'sId', тип: 'Mail.Objects. IUmqueItemId ', isArray: 'False', isOptional: 'False',
	isStruct: 'False'
Имя параметра	'sR', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sC', тип: 'Mail.CHents.Controls. ColumnId ', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO', тип: 'Mail.Objects. SortOrder ', iisAiray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'mL', тип: 'System.Boolean', isArray: 'False', isOptional: 'True', isStruct: 'False'
Имя обработчика событий	'PreviousPage'. Глагол: 'Get'
Имя параметра	'sR', тип: 'System.Int32', isAjray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sId', тип: 'Mail.Objects.IuniqueltemId', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'sC', тип: 'Mail.Clients.Controls.ColumnId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO', тип: 'MaiLObjects.SortOrder',
	isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'mL', тип: ' SystemJBooleari, isArray: 'False', isOptional: 'False', isStract: 'False'
Имя обработчика событий	'NextPage'. Глагол: 'Get'
Имя параметра	'sC', тип: 'Mail.CUents.Controls. Columnld ', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'sO', тип: 'MailObjects. SortOrder ', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'mL', тип: 'System.Boolean, isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'sId', тип: 'Mail.ObjectsjnJniqueltemld', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'sR', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'LastPage'. Глагол: 'Get'
Имя параметра	'sId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False',
	isStract: 'False'
Имя параметра	'mL', тип: 'System.Boolean', isArray: 'False', isOptional: 'False', isStract 'False'
Имя параметра	'sId', тип: 'System.Int32',isArray: 'False', isOptional: 'False', isStract 'False'
Имя параметра	'sC', тип: 'Mail.Clients.Controls.ColumnId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'sO', тип: 'Mail.Objects.SortOrder', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя обработчика событий	'Sort'. Глагол: 'Get'
Имя параметра	'sC', тип: 'Mail.Clients.Controls.ColumnId', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'sO', тип: 'Mail.Objects.SortOrder', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'mL', тип: 'System.Boolean', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'sR', тип: 'System.Int32', isArray: 'False', isOptional: ' True', isStract: 'False'
Имя параметра	'sld', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя обработчика событий	'MsgToPeople'. Глагол: 'Post'
Имя параметра	'id', тип: 'Mail.Objects. IUniqueItemId', isArray: 'False', isOptional: 'False', isStract: 'False'

Имя обработчика событий	'Move'. Глагол: 'Post'
Имя параметра	'DestId', тип: 'Mail.Objects.IUmqueItemld', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'id', тип: Mail.Objects.IUniqueItemId', isArray: 'True', isOptional: 'False', isStract: 'False'
Имя обработчика событий	'Copy'. Глагол: 'Post'
Имя параметра	'DestId', тип: 'Mail.Objects. IUmqueItemld ', isArray: 'False', isOptional: 'False', isStract: 'False'
Имя параметра	'id', тип: 'Mail.Objects.IUniqueItemId', isArray: 'True', isOptional: 'False',
	isStract: 'False'

Таблица А3

Примерные обработчики событий и параметры для обработчика событий контейнера получателя

Тип обработчика	'Mail.Clients.RecipientWellEventHandler'
Имя обработчика событий	'Resolve'. Глагол: 'Post'
Имя параметра	'Bcc', тип: 'System.String', isArray: 'True',isOptional: 'True', isStruct: 'False'
Имя параметра	'Cc', тип: 'System.String', isArray: ' True1, isOptional: 'True', isStruct: 'False'
Имя параметра	'To', тип: 'System.String', isArray: 'True', isOptional: 'True', isStruct: 'False'
Имя параметра	'Recips', тип: 'Mail.Clients.Common. RecipientInfoAC', isArray: 'False', isOptional: 'True', isStruct: 'True'
Имя обработчика событий	'ResolveOneRecipient'. Глагол: 'Post'
Имя параметра	'n', тип: 'System.String', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'GetRecipProps. Глагол: 'Post'
Имя параметра	'id', тип: 'System.String', isArray: 'False', isOptional: 'False', isStruct: 'False'

Имя параметра	'rs', тип: 'Mail.Clients.Common.Recipient Source', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'GetCache'. Глагол: 'Get'
Имя обработчика событий	'DeleteFromCache'. Глагол: 'Post'
Имя параметра	'Em',тип: 'System.String', isArray: 'False', isOptional: 'False', isStruct: 'False'

Таблица А4

Примерные обработчики событий и параметры для обработчика событий представления календаря.

Тип Обработчика	'Mail.Clients.CalendarViewEventHandler'
Имя обработчика событий	'PersistWidth'. Глагол: 'Post'
Имя параметра	'w', тип: 'System.Int32', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'fId', тип: 'Mail.ObjectsJUniqueItemld ', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'PersistHeight'. Глагол: 'Post'
Имя параметра	'fld', тип: 'Mail.Objects. IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'h', тип: 'System. B t32 '., isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'PersistReadingPane'. Глагол: 'Post'
Имя параметра	'IsMD', тип: 'System.Boolean', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	's', тип: 'Mail.Clients.Common. ReadmgPanePosition'; isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'fId', тип: 'Mail.Objects. HJniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя обработчика событий	'GetViewPayload'. Глагол: 'Post'
Имя параметра	'nvs', тип: 'System.Boolean', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'fId', тип: 'Mail.Objects. IUniqueItemld', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'days', тип: 'System.DateTime', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'vt', тип: 'Mail.Clients.Controls.Calendar
	ViewType', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'dir', тип: 'System.Int32', isArray: 'False', isOptional: 'True', isStruct: 'False'
Имя обработчика событий	'Move'. Глагол: 'Post'
Имя параметра	'et',тип:'System.DateTime',isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'fId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'days'. тип: 'System.DateTime', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'nvs', тип: 'System.Boolean', isArray: 'False', isOptional: 'True', isStruct: 'False'
Имя параметра	'id', тип: 'Mail.Objects.IUmqueItemld', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'st', тип: 'System.DateTime', isArray: 'False', isOptional: 'False', isStruct: 'False'

Имя параметра	'vt', тип: 'Mail.Clients.Controls.Calendar ViewType', isArray: 'False', isOptional:
	'False', isStruct: 'False'
Имя обработчика событий	'Delete. Глагол: 'Post'
Имя параметра	'nvs', тип: 'System.Boolean', isArray: 'False', isOptional: 'True', isStruct: 'False'
Имя параметра	'vt',тип: 'Mail.Clients.Controls.CalendarViewType', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'id', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'fId', тип: 'Mail.Objects.IUniqueItemId', isArray: 'False', isOptional: 'False', isStruct: 'False'
Имя параметра	'days', тип: 'System.DateTime', isArray: 'True',isOptional: 'True', isStruct: 'False'

Формула изобретения

1. Система передачи, содержащая:

один или более считываемых компьютером носителей для хранения:

множества обработчиков событий, причем каждый имеет пространство имен, ассоциированное с ним, каждый из упомянутого множества обработчиков событий дополнительно имеет один или более параметров, ассоциированных с ним; и

структуру данных, представляющую форматированный запрос данных, причем упомянутая структура данных включает в себя поле пространства имен, хранящее значение пространства имен, представляющее это пространство имен, поле обработчика событий, хранящее значение обработчика событий, соответствующее одному из множества обработчиков событий, причем упомянутый один из множества обработчиков событий ассоциирован с пространством имен, представленным значением пространства имен в поле пространства имен, и поле параметра, хранящее значение параметра, соответствующего одному из этих параметров, при этом упомянутый один из параметров ассоциирован с обработчиком событий, представленным значением обработчика событий в поле обработчика событий; и

распределенное клиентское приложение, развернутое серверным компонентом, причем упомянутое распределенное клиентское приложение сконфигурировано так, чтобы выполнять выполняемые компьютером команды для:

приема запроса данных от пользователя или процесса;

идентификации значения пространства имен, ассоциированного с принятым запросом;

выбора значения обработчика событий как функции принятого запроса и идентифицированного значения пространства имен;

определения значения параметра как функции принятого запроса и выбранного значения обработчика событий;

заполнения структуры данных посредством сохранения значения идентифицированного пространства имен в поле пространства имен, выбранного значения обработчика событий - в поле обработчика событий, и определенного значения параметра - в поле параметров; при этом заполнение содержит сохранение поля пространства имен и значения пространства имен как пара имя-значение, поля обработчика событий и значения обработчика событий как пара имя-значение, и поля параметра и значения параметра как пара имя-значение, и

передачи заполненной структуры данных к серверному компоненту; и

серверный компонент, сконфигурированный так, чтобы выполнять выполняемые компьютером команды для:

приема заполненной структуры данных от распределенного клиентского приложения;

выбора, как функции принятой структуры данных, обработчика событий из множества обработчиков событий, сохраненных на считываемых компьютером

носителях;

выполнения выбранного обработчика событий, чтобы сформировать данные результата, причем серверный компонент воспроизводит часть запрошенных данных, упомянутые сформированные данные результата представляют оставшуюся часть запрошенных данных, которые еще должны быть обработаны; и

передачи сформированных данных результата к распределенному клиентскому приложению, причем распределенное клиентское приложение выполняет сформированные данные результата так, что серверный компонент частично воспроизводит запрошенные данные, и распределенное клиентское приложение воспроизводит оставшуюся часть запрошенных данных.

2. Система по п.1, в которой считываемые компьютером носители и распределенное клиентское приложение ассоциированы с развернутым сервером клиентом управления персональной информацией.

3. Система по п.1, в которой считываемый компьютером носитель и серверный компонент ассоциированы с сервером управления персональной информацией.

4. Система по п.1, в которой значение параметра содержит данные, отформатированные согласно расширяемому языку разметки.

5. Система по п.1, в которой распределенное клиентское приложение обеспечивает по меньшей мере часть функциональных возможностей серверного приложения.

6. Система по п.1, в которой серверный компонент дополнительно сконфигурирован, чтобы выполнить выполняемые компьютером команды для построения базы данных конфигурации с множеством обработчиков событий, сохраненных на считываемом компьютером носителе.

7. Система по п.1, дополнительно содержащая средство для сохранения множества обработчиков событий.

8. Система по п.1, дополнительно содержащая средство для выполнения распределенного клиентского приложения и средство для выполнения серверного компонента.

9. Система по п.1, в которой серверный компонент дополнительно сконфигурирован, чтобы выполнять выполняемые компьютером команды для:

проверка достоверности значения параметра; и

преобразования значения параметра к типу данных, ассоциированному к ним.

10. Реализованный компьютером способ создания и доставки данных от распределенного клиентского приложения к серверному, от распределенного клиентского приложения к серверному компоненту, причем упомянутый реализованный компьютером способ содержит этапы:

прием распределенным клиентским приложением запроса данных от пользователя;

идентификацию распределенным клиентским приложением пространства имен, ассоциированного с принятым запросом, причем упомянутое идентифицированное пространство имен сохраняет значение пространства имен в поле пространства имен в структуре данных, упомянутое значение пространства имен представляет идентифицированное пространство имен;

определение распределенным клиентским приложением обработчика событий, ассоциированного с идентифицированным пространством имен, как функции принятого запроса, причем упомянутый обработчик событий сохраняет значение обработчика событий в поле обработчика событий в упомянутой структуре данных, упомянутое значение обработчика событий соответствует этому обработчику событий, упомянутый один из множества обработчиков событий ассоциирован с

пространством имен, представленным значением пространства имен в поле пространства имен;

заполнение распределенным клиентским приложением параметра, ассоциированного с определенным обработчиком событий, как функции принятого запроса, причем заполненный параметр хранит значение параметра в поле параметра в упомянутой структуре данных, упомянутое значение параметра соответствует этому параметру, упомянутый один из параметров ассоциирован с обработчиком событий, представленным значением обработчика событий в поле обработчика событий;

создание распределенным клиентским приложением форматированного запроса с идентифицированным пространством имен, определенным обработчиком событий и заполненным параметром;

при этом форматированный запрос данных содержит пространство имен и значение пространства имен в качестве пары имя-значение, обработчик событий и значение обработчика событий в качестве пары имя-значение, и параметр и значение параметра в качестве пары имя-значение,

передачу от распределенного клиентского приложения созданного форматированного запроса к серверному компоненту;

прием серверным компонентом форматированного запроса от распределенного клиентского приложения;

выбор серверным компонентом как функции принятого форматированного запроса обработчика событий из множества обработчиков событий, сохраненных в области памяти;

выполнение серверным компонентом выбранного обработчика событий, чтобы сформировать данные результата, при этом серверный компонент воспроизводит часть запрошенных данных, упомянутые сформированные данные результата представляют оставшуюся часть запрошенных данных, которые еще должны быть обработаны; и

передача от серверного компонента сформированных данных результата к распределенному клиентскому приложению, при этом упомянутые сформированные данные результата включают в себя поток кода, причем распределенное клиентское приложение выполняет сформированные данные результата так, что серверный компонент частично воспроизводит запрошенные данные, и распределенное клиентское приложение воспроизводит оставшуюся часть запрошенных данных.

11. Реализованный компьютером способ по п.10, дополнительно содержащий прием распределенным клиентским приложением переданных данных результата.

12. Реализованный компьютером способ по п.10, дополнительно содержащий этапы:

прием распределенным клиентским приложением от пользователя определяемого пользователем пространства имен, определяемого пользователем обработчика событий, ассоциированного с определяемым пользователем пространством имен, и определяемого пользователем параметра определяемого пользователем обработчика событий; и

регистрацию распределенным клиентским приложением в серверном компоненте принятого определяемого пользователем пространства имен, определяемого пользователем обработчика событий и определяемого пользователем параметра.

13. Реализованный компьютером способ по п.12, в котором серверный компонент регистрирует принятое определяемое пользователем пространство имен, определяемый пользователем обработчик событий и определяемый пользователем

параметр без изменения инфраструктуры серверного компонента.

14. Считываемый компьютером носитель, хранящий выполняемые компьютером компоненты для формирования и передачи запрошенных данных в сетевой среде, причем упомянутые компоненты содержат:

развернутый сервером клиентский компонент для:

приема запроса данных от пользователя;

идентификации пространства имен, ассоциированного с принятым запросом, причем упомянутое идентифицированное пространство имен сохраняет значение пространства имен в поле пространства имен в структуре данных, упомянутое значение пространства имен представляет идентифицированное пространство имен;

определения обработчика событий, ассоциированного с идентифицированным пространством имен, как функции принятого запроса, причем упомянутый обработчик событий сохраняет значение обработчика событий в поле обработчика событий в упомянутой структуре данных, упомянутое значение обработчика событий соответствует этому обработчику событий, упомянутый один из множества обработчиков событий ассоциирован с пространством имен, представленным значением пространства имен в поле пространства имен;

заполнения параметра, ассоциированного с определенным обработчиком событий, как функции принятого запроса, причем заполненный параметр хранит значение параметра в поле параметра в упомянутой структуре данных, упомянутое значение параметра соответствует этому параметру, упомянутый один из параметров ассоциирован с обработчиком событий, представленным значением обработчика событий в поле обработчика событий;

создания форматированного запроса с идентифицированным пространством имен, определенным обработчиком событий и заполненным параметром,

при этом форматированный запрос данных содержит пространство имен и значение пространства имен в качестве пары имя-значение, обработчик событий и значение обработчика событий в качестве пары имя-значение, и параметр и значение параметра в качестве пары имя-значение; и

передачи созданного форматированного запроса к серверному компоненту; и

серверный компонент для:

приема переданного запроса от развернутого сервером клиентского компонента;

выбора, как функции принятого запроса, обработчика событий из множества обработчиков событий, сохраненных в области памяти;

выполнения выбранного обработчика событий, чтобы сформировать данные результата, при этом серверный компонент воспроизводит часть запрошенных данных, упомянутые сформированные данные результата представляют оставшуюся часть запрошенных данных, которые еще должны быть обработаны; и

передачи сформированных данных результата к развернутому сервером клиентскому компоненту, при этом развернутый сервером клиентский компонент выполняет сформированные данные результата так, что серверный компонент частично воспроизводит запрошенные данные, и развернутый сервером клиентский компонент воспроизводит оставшуюся часть запрошенных данных.

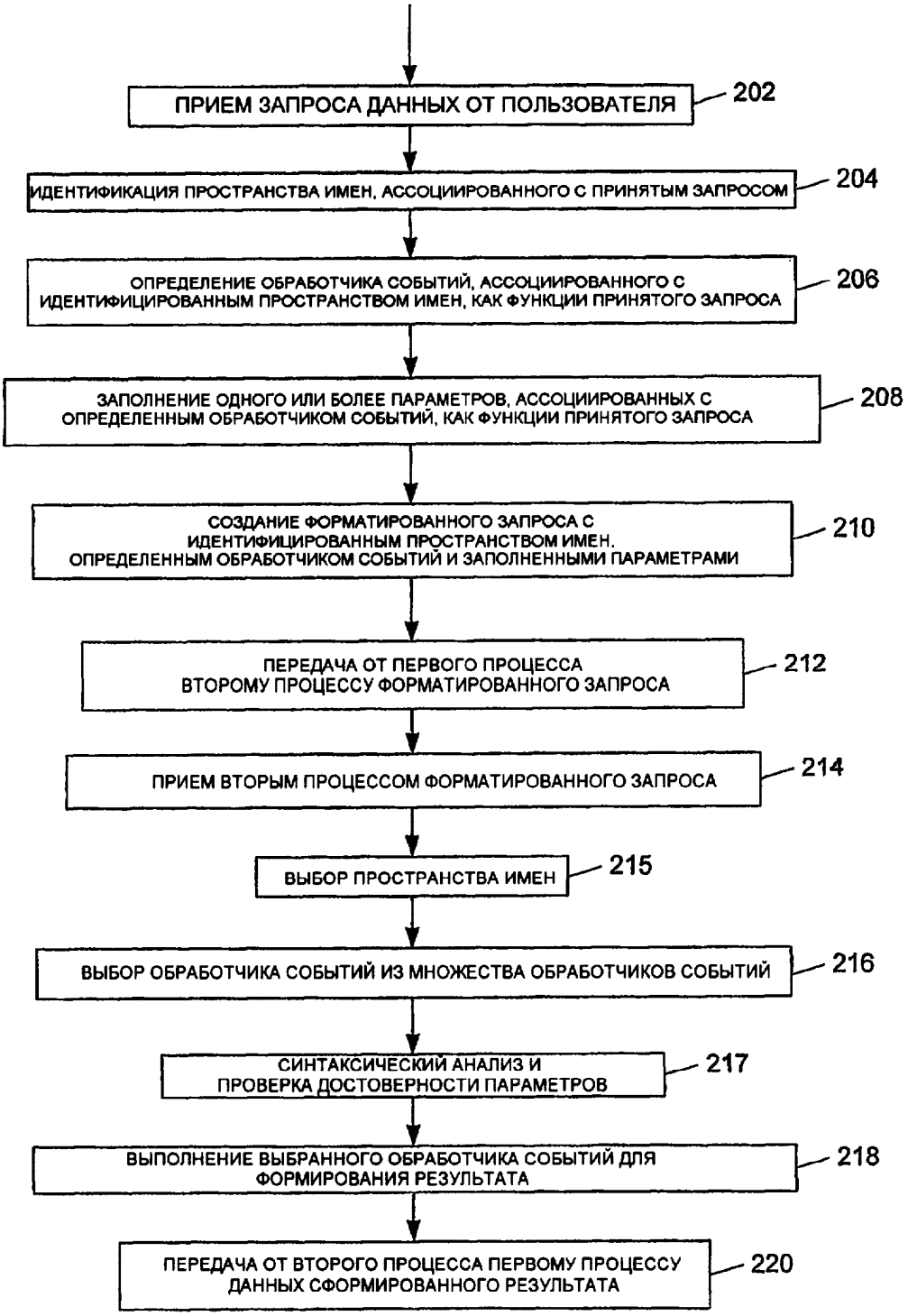
15. Считываемый компьютером носитель по п.14, дополнительно содержащий компонент регистрации для построения базы данных конфигурации с множеством обработчиков событий, сохраненных в области памяти.

16. Считываемый компьютером носитель по п.15, в котором серверный компонент выбирает, как функцию принятого запроса, обработчик событий из базы данных

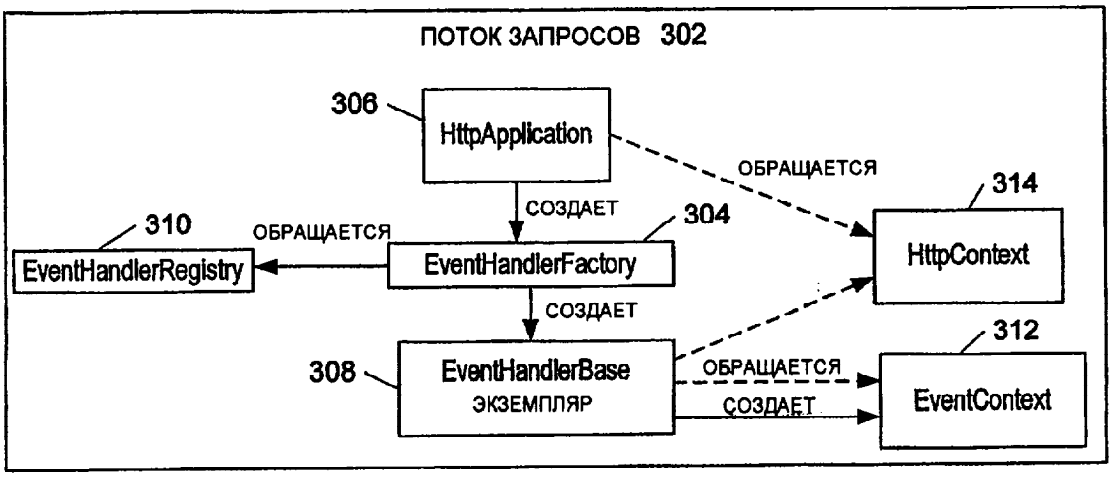
конфигурации.

17. Считываемый компьютером носитель по п.14, в котором развернутый сервером клиентский компонент содержит клиент управления персональной информацией, обменивающийся по сети с серверным компонентом.

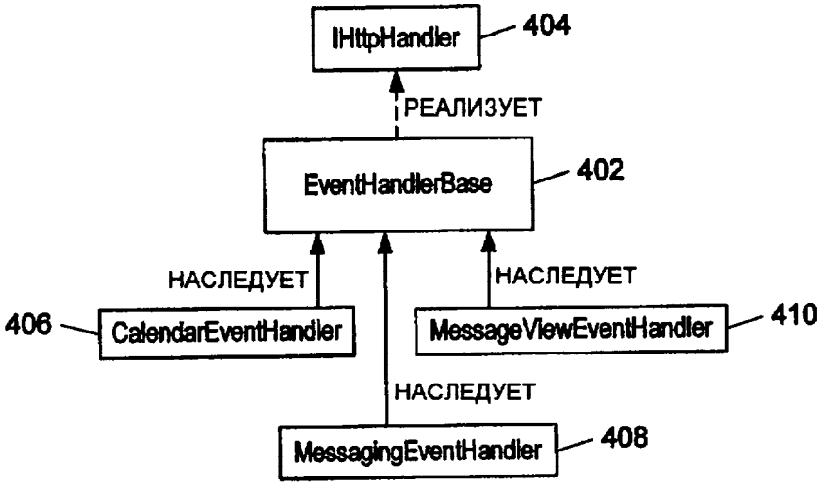
18. Считываемый компьютером носитель по п.14, в котором серверный компонент содержит сервер управления персональной информацией.


ПРИЕМ ЗАПРОСА ДАННЫХ ОТ ПОЛЬЗОВАТЕЛЯ
202
ИДЕНТИФИКАЦИЯ ПРОСТРАНСТВА ИМЕН, АССОЦИИРОВАННОГО С ПРИНЯТЫМ ЗАПРОСОМ
204
ОПРЕДЕЛЕНИЕ ОБРАБОТЧИКА СОБЫТИЙ, АССОЦИИРОВАННОГО С ИДЕНТИФИЦИРОВАННЫМ ПРОСТРАНСТВОМ ИМЕН, КАК ФУНКЦИИ ПРИНЯТОГО ЗАПРОСА
206
ЗАПОЛНЕНИЕ ОДНОГО ИЛИ БОЛЕЕ ПАРАМЕТРОВ, АССОЦИИРОВАННЫХ С ОПРЕДЕЛЕННЫМ ОБРАБОТЧИКОМ СОБЫТИЙ, КАК ФУНКЦИИ ПРИНЯТОГО ЗАПРОСА
208
СОЗДАНИЕ ФОРМАТИРОВАННОГО ЗАПРОСА С ИДЕНТИФИЦИРОВАННЫМ ПРОСТРАНСТВОМ ИМЕН, ОПРЕДЕЛЕННЫМ ОБРАБОТЧИКОМ СОБЫТИЙ И ЗАПОЛНЕННЫМИ ПАРАМЕТРАМИ
210
ПЕРЕДАЧА ОТ ПЕРВОГО ПРОЦЕССА ВТОРОМУ ПРОЦЕССУ ФОРМАТИРОВАННОГО ЗАПРОСА
212
ПРИЕМ ВТОРЫМ ПРОЦЕССОМ ФОРМАТИРОВАННОГО ЗАПРОСА
214
ВЫБОР ПРОСТРАНСТВА ИМЕН
215
ВЫБОР ОБРАБОТЧИКА СОБЫТИЙ ИЗ МНОЖЕСТВА ОБРАБОТЧИКОВ СОБЫТИЙ
216
СИНТАКСИЧЕСКИЙ АНАЛИЗ И ПРОВЕРКА ДОСТОВЕРНОСТИ ПАРАМЕТРОВ
217
ВЫПОЛНЕНИЕ ВЫБРАННОГО ОБРАБОТЧИКА СОБЫТИЙ ДЛЯ ФОРМИРОВАНИЯ РЕЗУЛЬТАТА
218
ПЕРЕДАЧА ОТ ВТОРОГО ПРОЦЕССА ПЕРВОМУ ПРОЦЕССУ ДАННЫХ СФОРМИРОВАННОГО РЕЗУЛЬТАТА
220

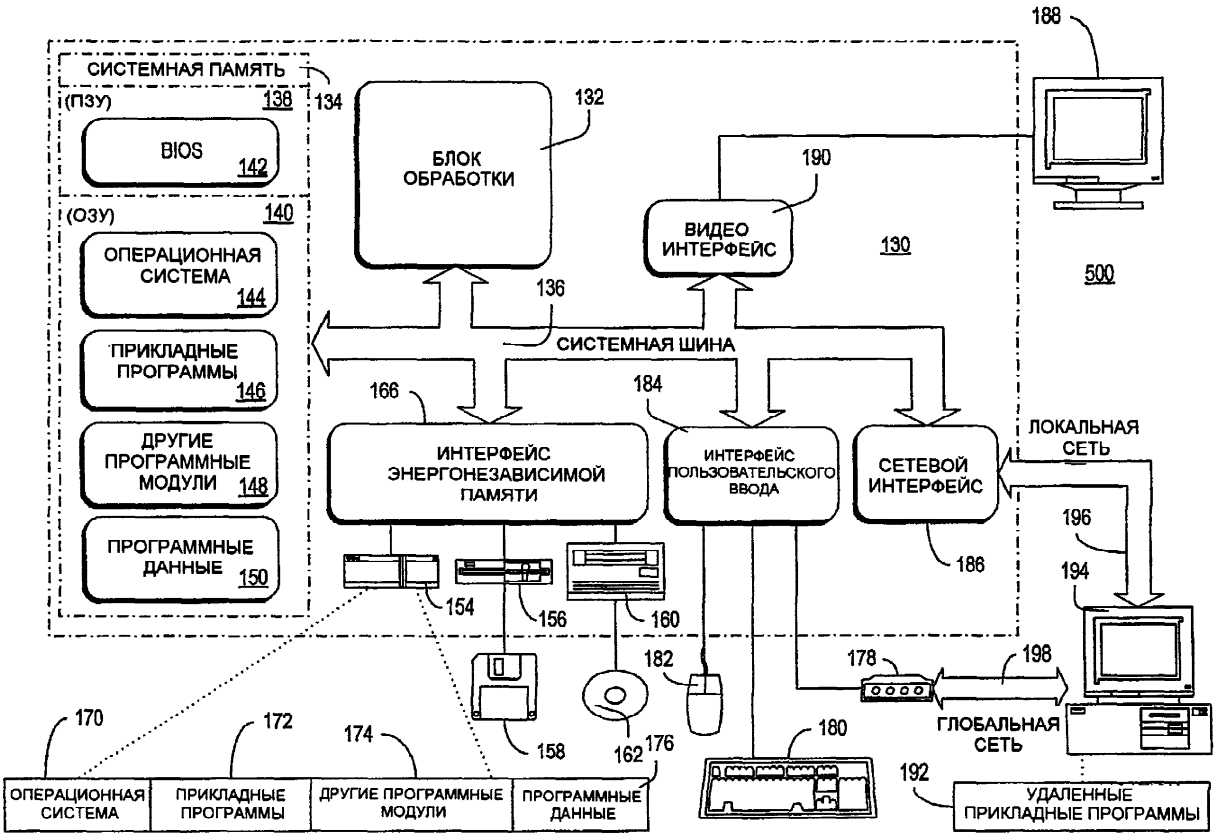
Фиг.2


ПОТОК ЗАПРОСОВ 302
306
HttpApplication
ОБРАЩАЕТСЯ
314
СОЗДАЕТ
310
ОБРАЩАЕТСЯ
304
EventHandlerRegistry
EventHandlerFactory
HttpContext
СОЗДАЕТ
312
308
EventHandlerBase
ЭКЗЕМПЛЯР
ОБРАЩАЕТСЯ
СОЗДАЕТ
EventContext

Фиг.3


IHttpHandler
404
РЕАЛИЗУЕТ
EventHandlerBase
402
НАСЛЕДУЕТ
НАСЛЕДУЕТ
406
CalendarEventHandler
MessageViewEventHandler
410
НАСЛЕДУЕТ
MessagingEventHandler
408

Фиг.4


188
СИСТЕМНАЯ ПАМЯТЬ
(ПЗУ) 138
134
BIOS 142
(ОЗУ) 140
ОПЕРАЦИОННАЯ СИСТЕМА 144
ПРИКЛАДНЫЕ ПРОГРАММЫ 146
ДРУГИЕ ПРОГРАММНЫЕ МОДУЛИ 148
ПРОГРАММНЫЕ ДАННЫЕ 150
БЛОК ОБРАБОТКИ
132
190
ВИДЕО ИНТЕРФЕЙС
130
500
136
СИСТЕМНАЯ ШИНА
166
184
ИНТЕРФЕЙС ЭНЕРГОНЕЗАВИСИМОЙ ПАМЯТИ
ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА
СЕТЕВОЙ ИНТЕРФЕЙС
ЛОКАЛЬНАЯ СЕТЬ
196
194
186
154
156
160
182
178
198
ГЛОБАЛЬНАЯ СЕТЬ
170
172
174
158
162
176
180
192
ОПЕРАЦИОННАЯ СИСТЕМА
ПРИКЛАДНЫЕ ПРОГРАММЫ
ДРУГИЕ ПРОГРАММНЫЕ МОДУЛИ
ПРОГРАММНЫЕ ДАННЫЕ
УДАЛЕННЫЕ ПРИКЛАДНЫЕ ПРОГРАММЫ

Фиг.5