ABSTRACT

Various technologies related to an enterprise computing platform are presented. A new level of software development can be achieved by avoiding the coding process. An enterprise computing platform having a variety of frameworks can be configured to operate in a variety of business domains. Features such as parallel computing, distributed computing, logical documents, document transformation, space visualization, data security, and others can be accomplished via configuration rather than coding. Considerable efficiency improvements in the software development process can be realized.

FIG. 1

We claim: We claim:

1. One or more computer-readable devices comprising computer-executable instructions for performing a method comprising:

in an extensible view architecture framework, receiving a request to interact with a document;

responsive to receiving the request, processing the request according to configuration information stored for the extensible view architecture framework;

collecting content of one or more logical document elements from one or more locations within a data source;

constructing a logical view of the document based on the one or more logical document elements; and—

providing access to the logical view for consumption by a view provider;

wherein the logical view is of a format for consumption by a plurality of different view provider types.

2. The one or more computer-readable devices of claim 1 wherein:

the logical view is encapsulated in one or more programming objects of an object class derived from a domain-independent base class.

3. The one or more computer-readable devices of claim 1 wherein:

the extensible view architecture framework supports entities other than documents; and

the entities other than documents and the documents inherit from a base entity class.

4. The one or more computer-readable devices of claim 1 wherein:

processing the request according to configuration information comprises directing a method call to a programming object of an object class specified in the configuration information.

5. The one or more computer-readable devices of claim 4 wherein:

the configuration information supports references to domain-specific object classes that derive from domain-independent object classes.

6. The one or more computer-readable devices of claim 1 wherein:

the document is represented by a class derived from a domain-independent entity class; and

the class derived from the domain-independent entity class contains domain-specific callable methods for processing domain-specific aspects of the document.

7. The one or more computer-readable devices of claim 1 wherein:

the extensible view architecture framework supports location independence.

8. The one or more computer-readable devices of claim 1 further comprising:

representing the document internally as a logical document comprising one or more logical document elements, wherein the content of the one or more logical document elements are stored as source documents in one or more locations within a data source.

9. The one or more computer-readable devices of claim 1 further comprising:

representing a logical view of the document, wherein the logical view is based on the logical document.

10. The one or more computer-readable devices of claim 9 further comprising:

instantiating a view manager as a view manager programming object; and

managing the logical view with the view manager.

11. The one or more computer-readable devices of claim 10 wherein:

the view manager programming object is of an object class derived from a domain-independent object class.

12. The one or more computer-readable devices of claim 11 wherein:

the domain-independent object class comprises:

a callable method creating a programming object representing the logical view.

13. The one or more computer-readable devices of claim 9 further comprising:

instantiating the logical view as a logical view programming object;

wherein the logical view programming object is of an object class derived from a domain-independent object class.

14. The one or more computer-readable devices of claim 13 wherein:

the domain-independent object class comprises:

a callable method drawing the logical view; and

a callable method capturing the logical view.

15. The one or more computer-readable devices of claim 9 wherein:

the logical view is of a view-provider-independent form.

16. The one or more computer-readable devices of claim 15 wherein providing access comprises:

providing a reference to the logical document to a view provider for translation from a logical view to a rendered view.

17. The one or more computer-readable devices of claim 16 further comprising:

presenting the rendered view, wherein the presenting visually presents a presentation of the document.

18. The one or more computer-readable devices of claim 1 wherein:

collecting content comprises accessing a serialized version of content of the document in a database;

drawing content of one or more logical view elements from the serialized version of the content of the document; and

constructing the logical view comprises representing the content of the one or more logical view elements in memory as the logical view.

devices of claim 9

19. The one or more computer-readable devices of claim 8 further comprising of claim 9

storing hierarchical relationships within the document; and

outputting the hierarchical relationships to a view provider for presentation by the view provider.

20. The one or more computer-readable devices of claim 1 further comprising:

storing rules for document transformation for a plurality of entities; and

in an enterprise automation framework, applying the rules for respective of the entities, the applying generating a plurality of transformed documents comprising the document.

21. The one or more computer-readable devices of claim 20 further comprising:

storing availability and capacity information for a plurality of servers; and

based on the availability and capacity information for the plurality of servers, in an enterprise computing framework, sending a job to a server;

wherein the job performs the generating for the document;

based on the availability and capacity information for the plurality of servers, in an enterprise computing framework, sending an other job to an other server; and

wherein the job performs the generating for an other document.

22. The one or more computer-readable devices of claim 21 further comprising:

in an enterprise resource and services management framework, storing configuration information for a plurality of other frameworks.

23. A method implemented at least in part by one or more computing devices, the method comprising:

receiving a request for a view of a document;

responsive to the request, accessing a logical document comprising one or more logical document elements of the document, wherein accessing the logical document comprises calling at least one method on at least one programming object representing the logical document, wherein the at least one programming object is configured to provide content of the document; and

further responsive to the request, providing the content of the logical document as represented by the at least one programming object representing the logical document.

24. One or more computer-readable devices comprising computer-executable instructions for performing the method of claim 23.

25. The method of claim 23 further comprising:

respresenting one logical view out of a plurality of logical views with a programming object representing the logical view;

providing access to the programming object representing the logical view;

receiving a method call on the programming object representing the logical view to provide the logical view for display as a rendered view; and

responsive to the method call, providing content in a view-provider-independent format for display as the rendered view.

26. The method of claim 23 wherein:

the logical document is represented by a domain-independent entity object class representing an entity.

27. The method of claim 26 wherein:

the domain-independent entity object class comprises the following attributes:

identifier;

meta type;

one or more serialization attributes; and

one or more security attributes.

28. The method of claim 27 wherein:

the meta type specifies whether that the entity is a document.

29. The method of claim 27 wherein:

the meta type supports specifying that the entity is a document, a view, a business object, and a use case.

30. The method of claim 23 wherein:

the logical document is represented by an object class having domain-specific features; and

the object class having domain-specific features is derived from a domain-independent entity object class.

31. The method of claim 23 wherein:

the logical document is represented by a more domain-specific object class having more domain-specific features, and the object class is derived from a lesser domain-specific object class having lesser domain-specific features; and

the lesser domain-specific object class is derived from a domain-independent entity object class.

32. The method of claim 23 wherein:

the document is a document type out of a plurality of document types; and

features are provided for the logical document by a configurable framework according to configuration information stored as associated with the document type.

33. The method of claim 32 wherein the features comprise:

whether a document is a compound document.

34. The method of claim 32 wherein the features comprise:

encryption.

35. A computer-implemented method of processing a document of a document type, the method comprising:

in a computer system comprising a processor, representing a logical view of the document for consumption by a view provider out of a plurality of view providers, wherein the logical view of the document is represented via an entity object or a class derived from the entity object;

representing a logical document, wherein the logical document comprises one or more logical document elements taken from scattered locations in a data source, and the logical document filters data from the scattered locations in the data source, wherein the logical document is represented via the entity object or a class derived from the entity object;

storing configuration information for the document type, wherein the configuration information comprises a meta type indicating that the document is a document;

receiving a request to view the document;

responsive to the request, calling a method on an object class for providing content of the document, wherein the object class is derived from a domain-independent object class for providing content of an entity; and

sending the content to the view provider for rendering as a physical view of the document.

36. A computer-implemented method comprising:

in a computer-system, receiving an indication from a user of one or more reference implementation layers;

responsive to receiving the indication from the user, accessing one or more reference implementations corresponding to the one or more reference implementation layers; and

configuring one or more domain-independent frameworks according to the one or more reference implementations;

wherein at least one of the one or more reference implementation layers is an intermediate reference implementation layer.

37. An enterprise computing platform stored on one or more computer-readable media and comprising:

an enterprise automation framework configured to transform one or more documents to a plurality of transformed documents according to business rules, wherein the enterprise automation framework is configured to send a request to an enterprise computing framework;

the enterprise computing framework, wherein the enterprise computing framework is configured to break down the request into a plurality of jobs and send the plurality of jobs to different servers based on server availability and capacity; and

an extensible view architecture framework configured to provide user interface interaction with the plurality of transformed documents.

38. The enterprise computing platform of claim 37 wherein:

the extensible view architecture framework is domain independent; and

the enterprise computing platform further comprises at least one domain-specific domain mapper configured to extend domain-independent functionality of the extensible view architecture framework to domain-specific functionality.

39. The enterprise computing platform of claim 38 wherein:

the at least one domain-specific domain mapper extends retail user interface functionality.

40. The enterprise computing platform of claim 37 wherein:

the extensible view architecture framework is further configured to store a logical document, wherein the logical document comprises one or more logical document elements comprising content drawn from a plurality of source documents stored in a data source.

41. The enterprise computing platform of claim 37 further comprising:

a common extensible pattern architecture framework, which comprises the extensible view architecture framework and a smart proxy.

42. The enterprise computing platform of claim 37 further comprising:

a hierarchical visualization framework configured to support user interface manipulation of hierarchically-arranged entities.

43. The enterprise computing platform of claim 42 wherein:

the hierarchical visualization framework implements spatial hierarchical visualization.

44. One or more computer-readable devices comprising computer-executable instructions for performing a method comprising:

receiving a request to generate a plurality of transformed documents according to business rules associated with a plurality of respective business locations;

breaking the request down into a plurality of jobs according to a policy description;

based on periodic monitoring of availability and capacity of a plurality of possible servers, sending the plurality of jobs to respective of the plurality of possible servers, wherein the plurality of jobs collectively generate the plurality of transformed documents;

managing the plurality of jobs for failure recovery and job dependencies;

storing content of the plurality of transformed documents in a data source as one or more source documents;

receiving a request from a view provider for a view of a document out of the plurality of transformed documents;

responsive to the request for the view, generating a logical document representing the document, wherein the logical document comprises a plurality of logical document elements drawn from the one or more source documents;

further responsive to the request for the view, generating a logical view based on the logical document and comprising content from one or more of the plurality of logical document elements; and

providing the logical view to a view provider for presentation as a rendered view.

45. The one or more computer-readable devices of claim 44 wherein:

processing the logical view with a programming object of an object class derived from a domain-independent entity object class, wherein the object class derived from the domain-independent entity object class comprises domain-specific attributes or callable methods.

46. A method comprising:

in a computer system, receiving a request to perform processing in an enterprise automation framework customized by configuration information;

from the configuration information, reading policy description information for the request;

breaking the request into a plurality of jobs according to the policy description information;

assigning the plurality of jobs to respective servers;

sending the plurality of jobs to the respective servers for processing;

receiving results of the processing from the respective servers; and

indicating a status of the request.

47. One or more computer-readable devices comprising computer-executable instructions causing a computer system to perform the method of claim 46.

48. The method of claim 46 wherein:

the policy description information is of a format according to a XML schema;

the XML schema specifies one or more tags for parallel processing; and

the XML schema specifies one or more tags for specifying retries on a per-job basis.

49. The method of claim 46 further comprising:

accepting the policy description information from a graphical user interface.

50. The method of claim 46 wherein:

breaking the request into a plurality of jobs comprises breaking the request in accordance with the policy description information.

51. The method of claim 46 further comprising:

receiving at least a portion of the policy description information from a programming object.

52. The method of claim 46 wherein:

the policy description information comprises configuration information specifying how to process a job error.

53. The method of claim 46 wherein:

the policy description information comprises configuration information specifying how many times a job is to be retried.

54. The method of claim 53 wherein sending the plurality of jobs to respective servers comprises:

resending a job that has failed responsive to a determination that the configuration information indicates that retries remain.

55. The method of claim 46 wherein:

the policy description information comprises configuration information specifies at least one job concurrency relationship; and

sending the plurality of jobs respects the at least one job concurrency relationship.

56. The method of claim 46 wherein:

the policy description information comprises configuration information specifying a job bundle comprising a plurality of jobs.

57. The method of claim 46 further comprising:

receiving a request to execute a job; and

responsive to receiving the request, invoking a thread execution handler provided as part of the configuration information.

58. The method of claim 46 further comprising:

monitoring workload at the respective servers; wherein

assigning the plurality of jobs to respective servers comprises evaluating the workload and assigning the plurality of jobs accordingly.

claim 59 cc One or more computer-readable storage media comprising computer-executable claim 59
instructions causing a computer to perform a method comprising:

receving a request to execute a bundle of jobs;

receiving configuration information for the bundle of jobs, wherein the configuration information comprises a reference to a programming object;

responsive to detecting the reference to the programming object, sending a request to the programming object for policy description information;

receiving from the programming object, policy description information specifying per-job retry information, and one or more job concurrency relationships;

assigning the jobs to respective different servers for execution, respecting the one or more job concurrency relationships;

detecting a failure of a job;

responsive to detecting failure, retrying the job in accordance with the per-job retry information;

detecting successful completion of the bundle of jobs;

responsive to detecting successful completion of the bundle of jobs, indicating success of the request.

60. An enterprise computing framework stored on one or more computer-readable media and comprising:

a request handler configured to receive a request;

a request workflow manager configured to break the request down into a plurality of jobs according to a policy description;

an enterprise resource config configured to manage details of servers executing in a network, wherein the details comprise availability of a server and capacity of the server;

a scheduler configured to schedule the plurality of jobs according to job dependencies specified in the policy description and server availability as indicated by the enterprise resource config;

a process monitor configured to monitor whether jobs complete successfully and identify a failed job, and orchestrate a retry request for the failed job based on the policy description; and

a recovery manager configured to receive the retry request from the process monitor and implement a retry for the failed job, wherein a number of allowable retries is specified in the policy description, and the enterprise computing framework respects the number of allowable retries specified.

61. The enterprise computing framework of claim 60 further comprising:

a job workflow manager;

a thread workflow manager; and

a response manager.

62. The enterprise computing framework of claim 60 wherein:

the policy description accommodates a different number of retries for different jobs.

63. The enterprise computing framework of claim 60 wherein:

the policy description accommodates dependencies between jobs; and

the enterprise computing framework executes the jobs according to the dependencies.

64. A computer-implemented method comprising:

in a computer system, receiving a request to generate a plurality of transformed documents, wherein the request specifies a plurality of destinations;

receiving a source document representing a standardized business entity;

repeatedly transforming the source document according to the request and business rules, the transforming generating the plurality of transformed documents; and

repeatedly storing the plurality of transformed documents as associated with respective destinations.

65. The computer-implemented method of claim 64 wherein:

the source document represents a standardized business resource; and

the plurality of transformed documents represent customized versions of the standardized business resource, wherein the customized versions represent a plurality of respective destination business resources; and

transforming the source document comprises customizing the source document according to rules associated with respective of the destination business resources.

66. The computer-implemented method of claim 65 wherein:

the source document represents a standardized planogram for a retail store; and

the plurality of transformed documents represent customized versions of the standardized planogram, wherein the customized versions represent a plurality of respective fixtures.

67. The computer-implemented method of claim 64 wherein storing the plurality of transformed documents comprises:

storing the plurality of transformed documents as logical documents in a database.

68. The computer-implemented method of claim 64 wherein repeatedly transforming the source document comprises:

transforming the source document at two different servers having separately executing local transformation engines.

69. The computer-implemented method of claim 64 further comprising:

receiving a request to view a transformed document out of the plurality of transformed documents; and

responsive to the request, presenting the transformed document in a user interface provided by a space visualization framework.

70. The computer-implemented method of claim 64 wherein:

repeatedly transforming the source document comprises enforcing transaction processing discipline on database operations.

71. The computer-implemented method of claim 64 wherein the request is received by an enterprise automation framework, them method further comprising:

logging activities of the enterprise automation framework; and

based on the logged activities, providing an audit trail for display to a user.

72. A software development kit comprising:

an extensible view architecture framework encoded on one or more computer-readable media;

one or more reference implementations encoded on the one or more computer-readable media.

73. A computer-implemented method of developing software comprising:

receiving a request to build a new software application;

presenting a user interface accepting an indication of a business domain;

receiving an indication of the business domain;

responsive to receiving an indication of the business domain, selecting a pre-coded reference implementation for the business domain; and

customizing for domain-independent framework for the business domain, wherein the customizing comprises merging the domain-independent framework with the pre-coded reference implementation for the business domain, wherein the merging comprises indicating object classes derived from domain-independent object classes of the domain-independent framework.

Dated: 12th day of August, 2011

Anuj Kulshreshtha
Patent agent no. 1352
Infosys Limited

following detailed description of disclosed embodiments, which proceeds with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 is a block diagram of an exemplary system implementing the enterprise computing platform technologies described herein.

FIG. 2 is a flowchart of an exemplary method of implementing the enterprise computing platform technologies described herein.

FIG. 3 is a block diagram of an exemplary system implementing the enterprise computing platform technologies described herein for use by external applications.

FIG. 4 is a flowchart of an exemplary method of implementing the enterprise computing platform technologies described herein via platform configuration information.

FIG. 5 is a block diagram of an exemplary system for developing an application using the enterprise computing platform technologies described herein.

FIG. 6 is a flowchart of an exemplary method of developing an application using the enterprise computing platform technologies described herein.

FIG. 7 is a block diagram of an exemplary extensible entity class system.

FIG. 8 is a flowchart of an exemplary method of providing functionality according to base functionality extended by domain-specific functionality.

FIG. 9 is a block diagram of an exemplary domain mapper system.

FIG. 10 is a flowchart of an exemplary method of implementing domain mapping.

FIG. 11 is a screen shot of an exemplary user interface for choosing reference implementation layers.

FIG. 12 is a flowchart of an exemplary method of implementing reference implementation layers.

FIG. 13 is a block diagram of an exemplary extensible view architecture framework a system.

FIG. 14 is a flowchart of an exemplary method of providing a view of a document.

FIG. 15 is a block diagram of an exemplary extensible view architecture framework providing a plurality of views of a logical document.

FIG. 16 is a flowchart of an exemplary method of providing a view of a logical document.

FIGS. 17A-B are block diagrams of an exemplary smart proxy system.

FIG. 18 is a block diagram of an exemplary enterprise automation framework system.

FIG. 19 is a flowchart of an exemplary method of transforming documents.

FIG. 20 is a block diagram of an exemplary enterprise computing framework system.

FIG. 21 is a flowchart of an exemplary method of fulfilling a request via execution of a plurality of jobs at a plurality of servers.

FIG. 22 is a block diagram of an exemplary enterprise computing framework system with detail concerning managers.

FIG. 23 is a block diagram of exemplary policy description information.

FIG. 24 is a flowchart of an exemplary method of processing policy description information during job processing.

FIG. 25 is a block diagram of an exemplary system supporting object references in policy description information.

FIG. 26 is a flowchart of an exemplary method of processing object references in policy description information during job processing.

FIG. 27 is a block diagram of an exemplary enterprise management framework system.

FIG. 28 is a block diagram of an exemplary visualization framework system.

FIG. 29 is a flowchart of an exemplary method of fulfilling a request to display a document.

FIG. 30 is a block diagram of an exemplary document hierarchy.

FIG. 31 is a screen shot of an exemplary user interface for displaying a document in a document hierarchy.

FIG. 32 is a flowchart of an exemplary method of navigating to a different level in a document hierarchy while a document in the hierarchy is being displayed.

FIG. 33 is a flowchart of an exemplary method of processing documents using a plurality of the frameworks.