

[0572] A mime type is a simple component that describes the content of a stream of data, array of bytes or a file. The mime type consists of the primary part and subpart, separated by a slash. The primary part groups data by datatype like “text” or “image”. The subtype specifies the format of the data exactly like “gif” or “jpeg”. File extensions can also be associated with mime types in order to aid in file processing.

[0573] An object pool is a component that stores interchangeable components of the same type. Processes can remove an object from the pool and use it. Each object is returned to its original state before being placed in the pool for use by another process. Using an object pool makes sense when it takes a lot of time to construct or destroy an object, but very little time to clean or reset the object.

[0574] An object cache stores infrequently changed objects. Processes can use the object while it remains in the cache. Using an object cache makes sense for objects that take a long time to construct and change infrequently. When the object must be changed, the cache can place a lock on the object to prevent access while the changes are made. Once the object has been updated, the lock can be removed.

[0575] Both object pools and object caches store objects in memory in order to reduce the amount of processing performed by the system. In order to balance memory use and processing power, strategies must be employed to control the amount of memory used. These strategies may be examined periodically to ensure that the correct one is being used and is configured optimally. Caches usually use a “least-frequently-used” strategy to eliminate objects that are not used often enough to warrant remaining in memory. Pools often implement a “preferred-level strategy” which slowly decreases the number of objects in the pool to a preferred number.

[0576] A resource store is an abstraction away from a specific type of binary storage. Rather than have file managers, database large object binary managers, encoded image managers, zip file managers, and many other forms of binary data access, these concepts are all grouped into a binary “resource” concept. Each type of resource store handles access to one type of binary resource. For example, a file resource store would provide a mechanism for reading and writing the data store in a file. A Blob resource store would provide a mechanism for reading and writing database binary large objects.

[0577] The sequence manager manages sequence components. Sequences are components that generate numbers according to some rule. A very simple sequence starts with 1 and then returns each subsequent integer (2, 3, 4, 5, etc.). Other sequences may be much more complex or even random. Sequences are useful for generating numbers for primary keys or other unique identities.

[0578] Threads are a mechanism for a computer to execute multiple instructions in a way that appears to be simultaneous. Each set of instructions is executed for a period of time and then swapped out for some other thread’s instructions. By swapping threads quickly, each makes progress toward completion and each appears to be running at the same time. Threads are grouped into thread pools. The thread pool manager manages these pools. By controlling the manner in which threads execute the instructions, a thread pool can throttle or accelerate the processing occurring within the thread pool.

[0579] XML is a popular data markup standard. The ability to convert between objects and XML is handled by the `xml_serializer` and `xml_taghandler` managers. The XML serializer manager manages XML serializer components that convert objects to XML. The XML taghandler manager converts XML back into objects. XML serializers are retrieved by the type of object that needs to be converted to XML. XML taghandlers are retrieved by the XML tag name being converted into an object.

[0580] For an object to be a manager, the developer must implement the manager interface. This allows for many other types of managers to be added to the system as the need for them arises. Therefore, the above list may not contain all possible managers and the above managers can be bound to different names to serve different purposes. For example an image manager might be bound to “campaign_images” to store the images related to a specific campaign. Or an XML serializer manager might be bound to “partner_xml” to write XML used only by a partner company.

[0581] Side effects are events that happen during code execution and continue to exist after execution has ended. For example a component that writes files may construct the directory structure prior to writing the file. Once the files are written, the directory structure remains. A component may create an index on a database table to make record selection faster. While not necessarily undesirable, side effects can make it more difficult to share a component if they alter the environment.

[0582] Security is required for any multi-user architecture. The component integration engine enforces security at the levels detailed below.

[0583] No amount of software security will protect a system not physically under lock and key. If a thief has the ability to access the computer hardware, all software techniques stand a much greater chance of being circumvented. Data can be accessed without going through the system’s security constraints. Passwords can be physically overwritten on the disk or password scanners can be applied to retrieve passwords. Only encrypted data pose a problem to someone with physical access to a secure system and even then the data may eventually be decrypted given enough time and processing power.

[0584] Sandbox security refers to security built into the programming language. The sandbox security policy controls which components are allowed to run and what actions they are allowed to take. The mechanism for how this is controlled is language dependant. One embodiment of the present invention might use Java’s sandbox security policy. Another embodiment of the present invention might use Net’s security policies. The CIE owner will standardize the sandbox security mechanisms of each platform into a security policy component type.

[0585] The authorization policy is similar to the sandbox policy except that it lists the privileges to execute code and access resources associated with each user. The sandbox policy decides whether or not code is trusted to execute at all. The authorization policy decides if the user is trusted to execute a specific piece of code.

[0586] In order to determine the user associated with the current execution of a task, the user must provide credentials verifying identity. Usually some part of these credentials is