Backware

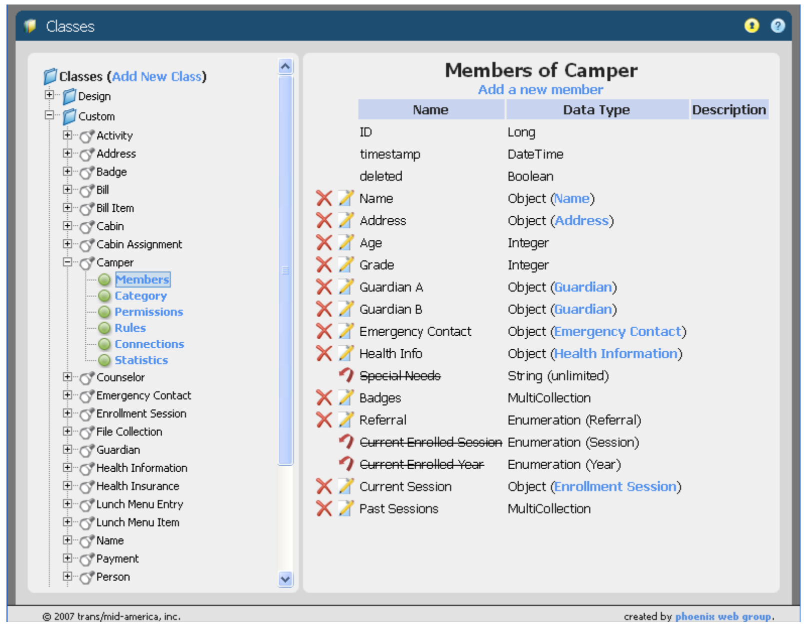
Data management with Simplicity, POWER, and *Performance*

The Building Blocks of Backware: Classes, Searches, Processes, Swappers

Backware has several tools to help you get your job done, but the system revolves around Classes, Searches, Processes, and Swappers as the major instruments.

The Classes Interface

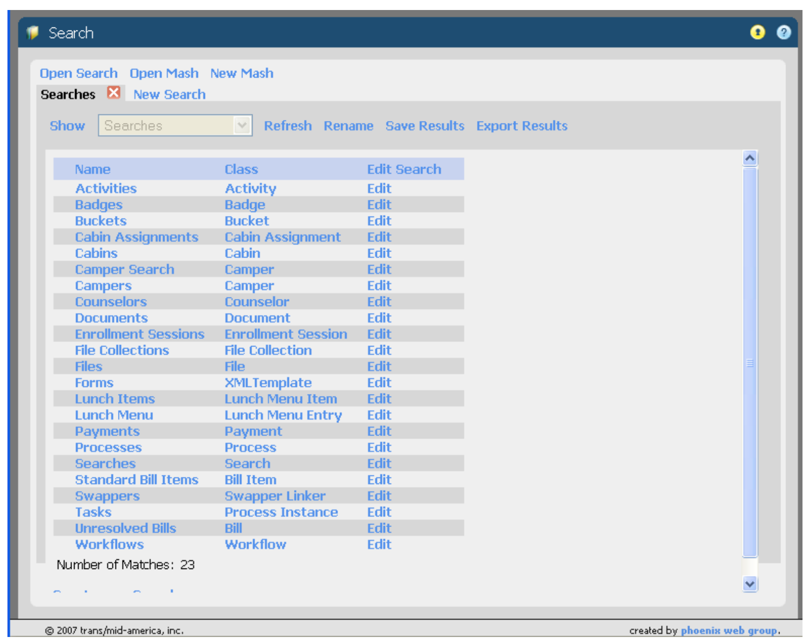
Basically, classes define the set of properties that all objects of its type will have. These properties are called members. However, each separate object will have its own values for those members. The Classes interface allows a class designer to create, modify, and delete the classes that a system will use. From here, classes can be started in a special design mode that prohibits them from being used while the class designer finishes his/her work. From there, the classes can be finalized and moved into production. In both stages, members can be added, removed, or changed, but some changes are restricted in the production mode, and removals are also only superficial so that no data is actually lost.



The Classes interface also allows designers to determine how members can be filled. We call this feature "Class Rules," and the rules too, can be modified in both stages of class design. The interface for working with class rules is completely graphical, but the result is actually turned into compiled code behind the scenes. That means property validation is speedy for the user.

The Searches Interface

The major hurdle to overcome when approaching Searches is that it does not work like your favorite Internet search engine. Those engines already know what you want to search: the Web. Before you can search in Backware, though, it must know what you want to search. Specifically, it needs to know what class of objects you want to search in order to get the set of results you are looking for. It is for that reason that Backware allows the creation of multiple search objects, each one based on a class in the system, which is the class of result objects. And because each search object can be based on a different class, it must be told which properties to use as criteria and which properties to display in the results.

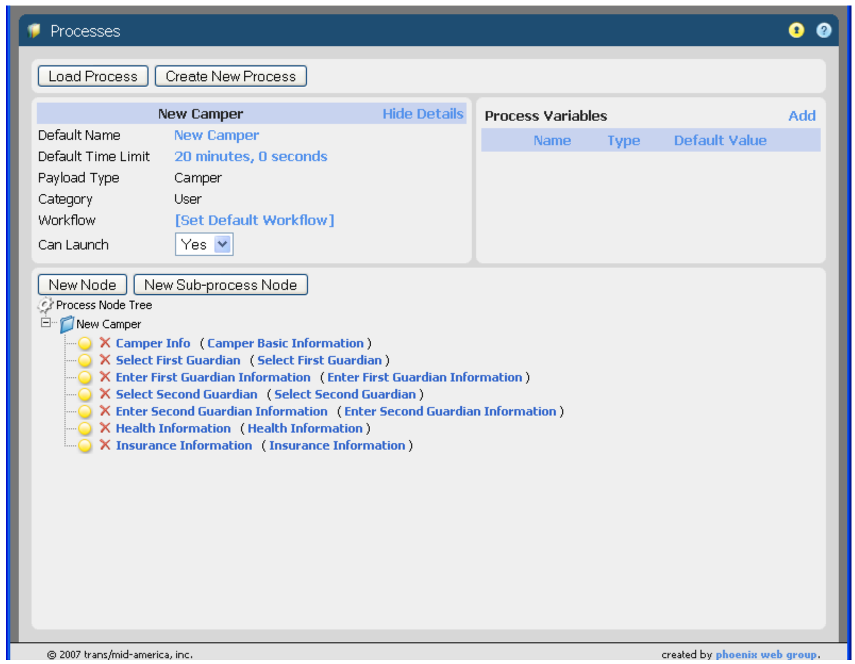


Once a search object is created, it can be used by any user who has permissions to do so. Search objects can be continually reused because they only provide the framework for searching. The result set is never captured or saved in a search object.

Searches form the very heart of a Backware application because they not only provide a way to view the exact data a user is looking for, but ways to interact with those objects. A search object can associate actions with columns in the results which may allow a user to edit an object's properties, send it to another user, view an address on a map, or just about anything else a search designer would want the user to be able to do. So much can be done with this interface that it could be the only tool a Backware user may need to know for their job.

The Processes Interface

Processes give Backware another level of flexibility and power. They can be used to manipulate objects, guide a user through a task, or automate repetitive or periodic jobs. Like searches, processes are really frameworks that are created and then repeatedly reused to accomplish their tasks. Using a process is as simple as following the steps of the process, which may include entering information in forms, sending or receiving communication, or even passing it along to someone else for their contribution. Yes, processes can be multi-user and multi-group! In addition, they may be assigned workflows that actually enforce the changing of hands, so each part of the process can be completed by a qualified individual or member of a task team.

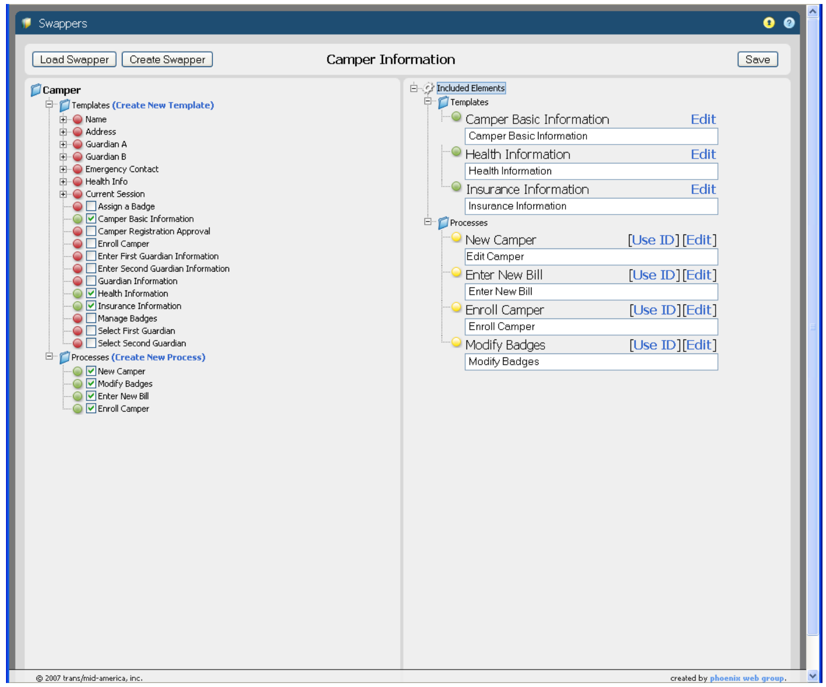


From the design perspective, processes are made for rapid development and execution. Each process consists of a series of nodes which may or may not include a form to show to the user and/or code to be executed. We use the power of the Microsoft .NET framework to dynamically compile the designer's code into assemblies to be executed upon use. Further, we provide a powerful Application Programming Interface (API) for designers to quickly write their code and be on their way. There is no need to rewrite or copy code either. Processes can include one another so that large processes can be built from many smaller ones, such as entering address or billing information. With this model, updating a process with future changes is a breeze.

Each process is built with one specific class in mind and, when executed, will always pass one object of that class along each step. However, using the forms, sub-processes, and designer code, this one object can interact with any other object in the system, as long as the user has permission to do so, of course.

The Swappers Interface

With searches, a user can view a common set of properties for a set of objects and interact with them. With processes, a user can have guided interaction with an object. There is still a gap, however. Suppose the user wants to see or modify multiple aspects of the object or other objects connected to it. For that, we created swappers. The name comes from the paneled interface which swaps informational forms as the user requests them.



Forms are created by a system designer to show or allow modification of object properties. Then, the forms a designer wants available for a user are selected and inserted into a swapper. Any processes that the designer wants the user to have access to are also inserted. Then, a user launches an object within a swapper and uses those forms and processes to quickly access or change the object's data.

Swappers are flexible enough to access other object data based on the primary object. Suppose you are looking at our golden delicious apple, which happens to be in a crate of apples. The swapper provides a link to a Crate form, which shows information about its arrival date and weight. The form is automatically updated with information about the crate that our golden delicious is in because it is the primary object of our swapper. The same swapper could be used to view information about our red delicious and its crate, which might have a different weight and arrival date.

Technologies Used In Backware

Server-Side – ASP.NET 2.0

The server-side Backware application runs on Microsoft's ASP.NET Framework 2.0, which means stability, great features, and managed code. Unlike interpreted web languages like PHP or ASP classic, ASP.NET applications can be pre-compiled for speedy execution, while still providing the same level of flexibility. Because the code is memory-managed, the application is extremely hardened against crashes.