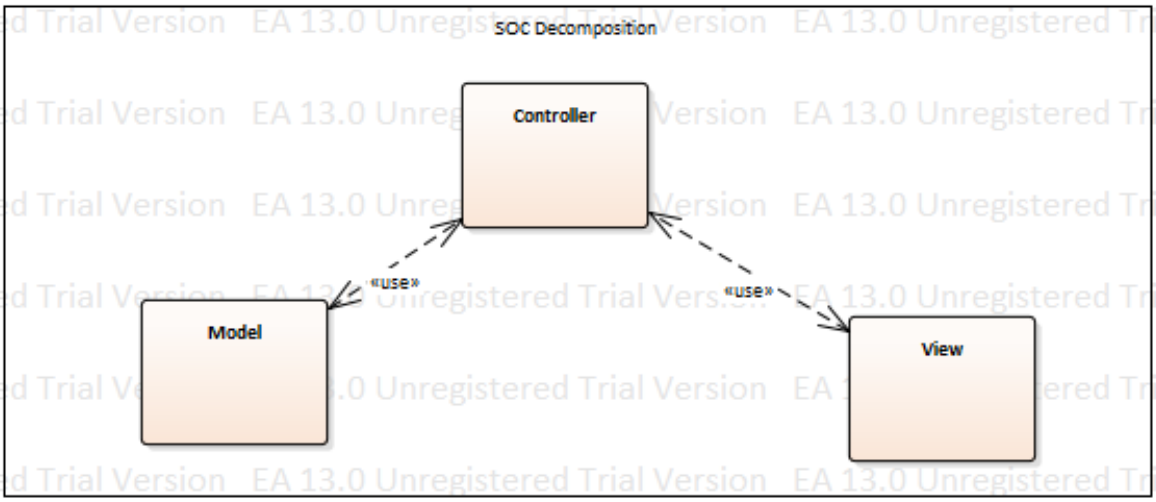
Web MVC Framework main features

Introduction

Web MVC Framework offers to developers a complete set of functionalities for rapid development of data intensive web applications. Generally, it provides services for system decomposition that developers can do at different levels when coding a complex web application. Firstly it entire implements services for realizing the *MVC design pattern* decomposition. However, this is not the only feature provided by the Framework for acting application decomposition. The list below shows its main characteristics by detailing these aspects and many others that Framework includes.

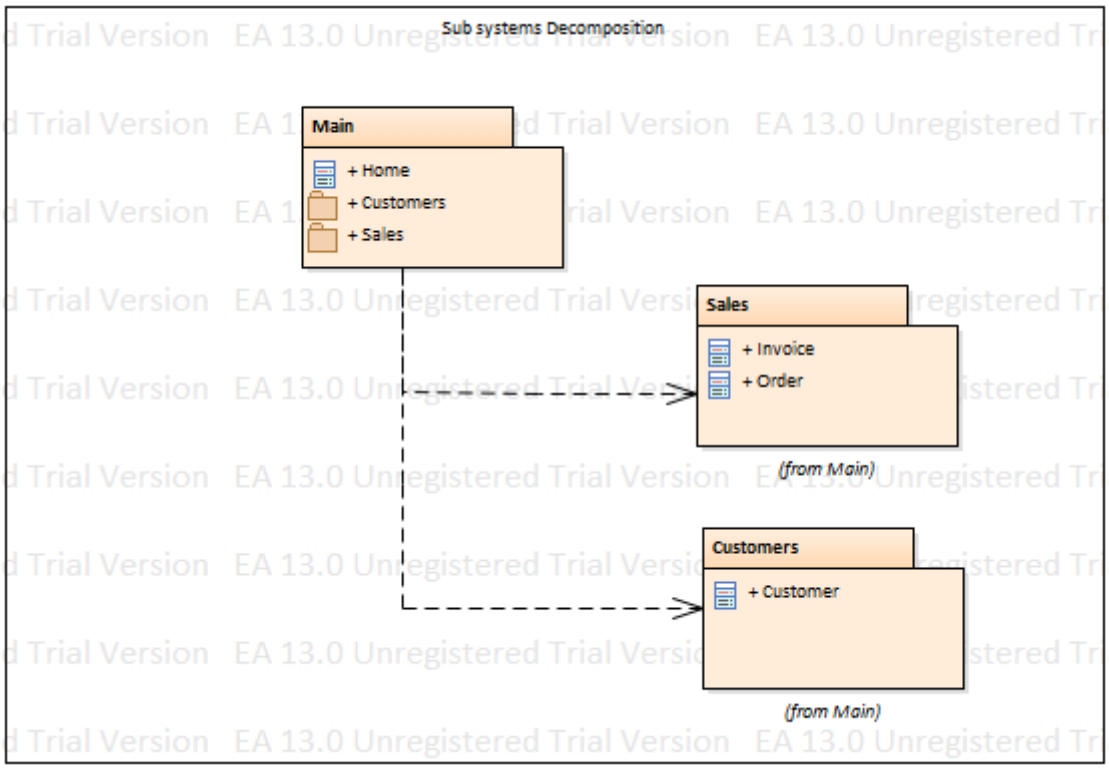
MVC Design Pattern

Framework implements the ***MVC design pattern*** allowing developers to decompose complex logic into different layers. This is a specialization of the well know concept of the ***Separation of Concern,*** alias ***SOC,*** which thinks to an application like a big set of functionalities intercommunicating each other's but having different system purposes. SOC organizes application by identifying its main purposes and then by grouping all its functions under these ones. MVC, which is a SOC implementation, classifies purposes under three high-level application tiers: data, presentation and application logic by providing, respectively, Model, View and Controller intercommunicating logic layers. Therefore, by implementing a web application and using MVC pattern developers must decomposing and grouping its classes under this layers and must manage communications among them.
Framework offers all base classes for building Model, View and Controller layers of a web application and for simplifying data communications. Developers can quickly create the application MVC layers just by extending Framework classes. Framework will provide the necessary service to the class's instantiation and intercommunication.
However, the MVC decomposition is not mandatory when using the Framework. In fact, it also supports classic PHP development approach, where developers can write code into a single file by using booth OOP or Functional paradigm. This aspect facilitate the integration of the Framework together with any existing application that use a different development technique.



Auto loading, Namespaces and Sub Systems

Framework, depending from application's complexity, lets developers to organize and decompose application's classes by using namespaces and sub systems. Sub systems allows designing the application's architecture with a high level of decomposition, well known as ***Systems Decomposition***. System Decomposition looks at business logic decompositions of an application. It differs from the SOC decomposition, which is principally oriented to the functions concern like, for example, file system functions, memory management, database, GUI functions and so on.



The mechanism of classes' auto loader, in conjunction with the namespaces, also simplify the objects' creation of the framework and application's classes, also if they are located into different sub systems, without the need of their explicit files' inclusion and also eliminating naming conflicts among sub systems. Frameworks uses case sensitive 1 to 1 mapping for:

- namespace ⇔ sub system : a namespace named *XYZ* identifies a sub system named *XYZ* with its own variables scope
- sub system ⇔ web server folder : sub system named *XYZ* is placed into a folder with the same name
- class name ⇔ file : A class name *MyClass* is placed into a file *MyClass.php* which is located into a sub system folder

For example:

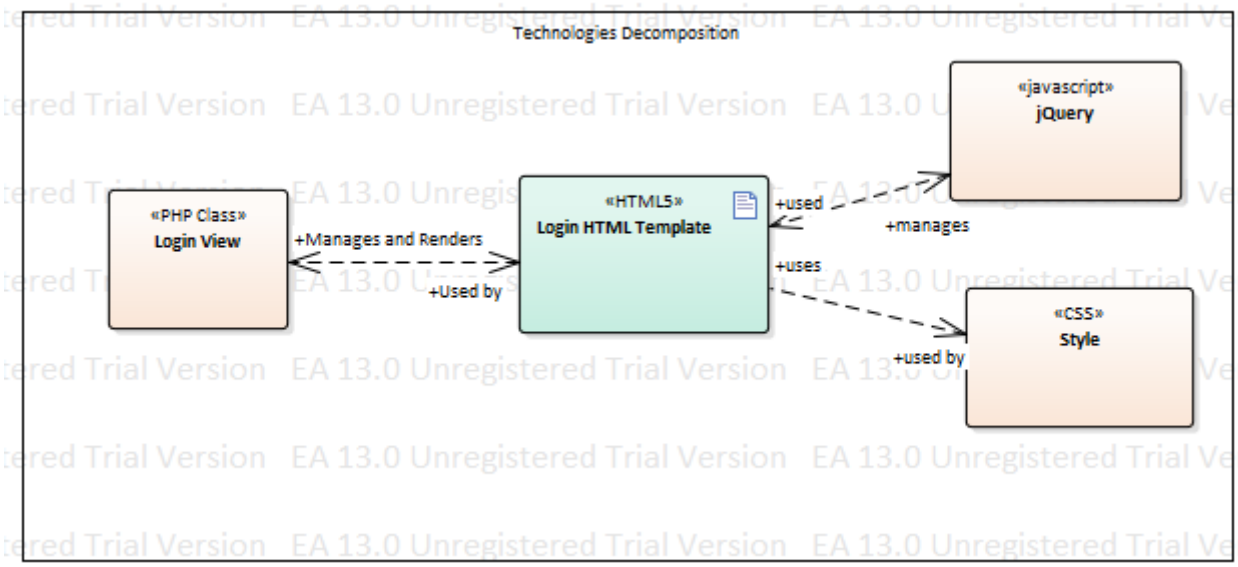
- namespace controllers\MySubSystem ⇔ web_root/controllers/MySubSystem
- use controllers\MySubSystem \MyClass⇔ web_root/controllers/MySubSystem/MyClass.php

Template Engine for the code separation between HTML design and PHP logic

Templating functionalities of the Framework permits to decuple PHP source code used for the presentation logic from HTML/CSS source code needed for the GUI design. Both pieces of code are under the responsibility of the View layer and must coexist inside it. By using the Framework, will be possible to avoid the mixture of different programming languages when implementing the View. Developers can build a View by putting the GUI static design into an external and standard HTML file, which acts as a template, optionally containing some special elements: placeholders and blocks. A placeholder is just a simple string between braces, for example, `{Placeholder}` while a block is common HTML code between special HTMLs comments like:

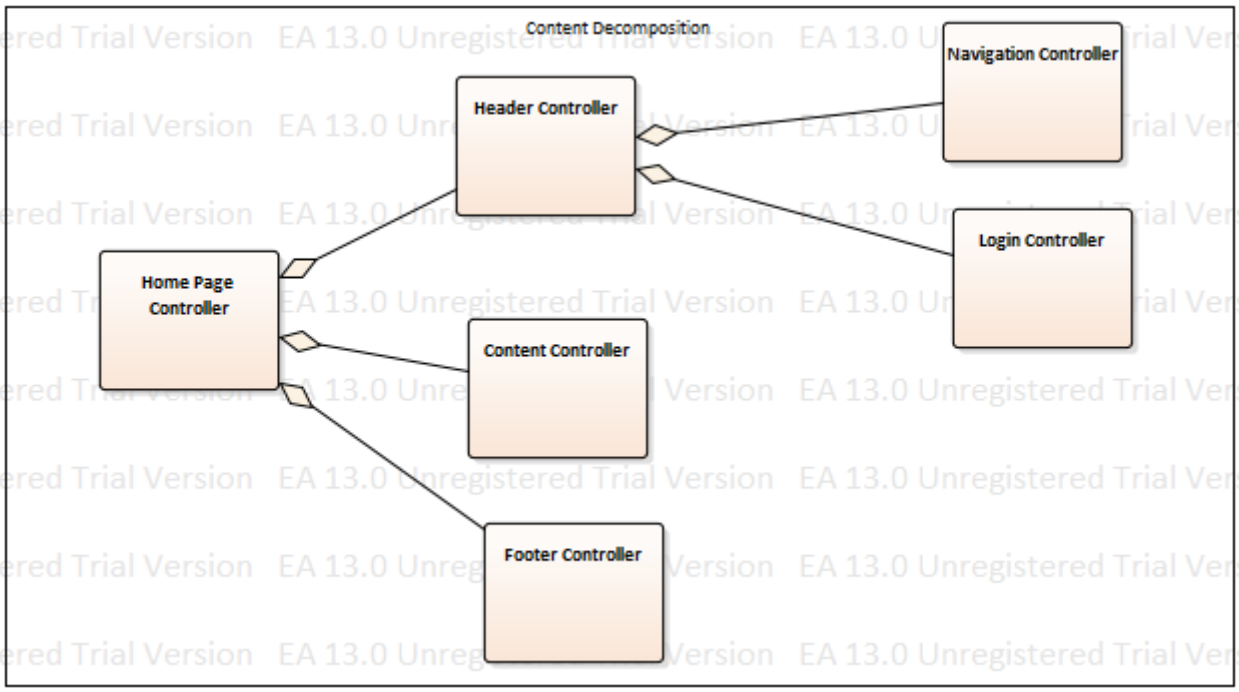
```
<!-- BEGIN block --> custom HTML <!-- END block -->
```

Both placeholders and blocks does not invalidate the standard of HTML syntax because they are not out of standard custom tags or out of HTML pseudo code. They are just simple text strings or comments inside canonical HTML code. Therefore, developers can write the View class simply by extending Framework's View base class or also by directly instantiating it. In both cases View will contains only pure PHP code and will be decupled from any HTML syntax. A View of this type will result enforced by Framework's Template Engine and it will be able to render automatically and dynamically, also by consuming some data provided from the Model or Controller, all blocks and placeholders existing into the external HTML template.
MVC model in conjunction with the Template Engine of the Frameworks offers other levels of decomposition, oriented to ***Decupling and Separation of Technologies.*** It gives to PHP View functionalities for rendering any data provided from external sources, like a database, into a GUI static design built into an external HTML file. It avoids the mixing of programming languages into a single source code file and permits to the designers to build the application design by using only client side technologies and to developers to builds pure PHP application without the needs of mixing each other their respective artifacts.



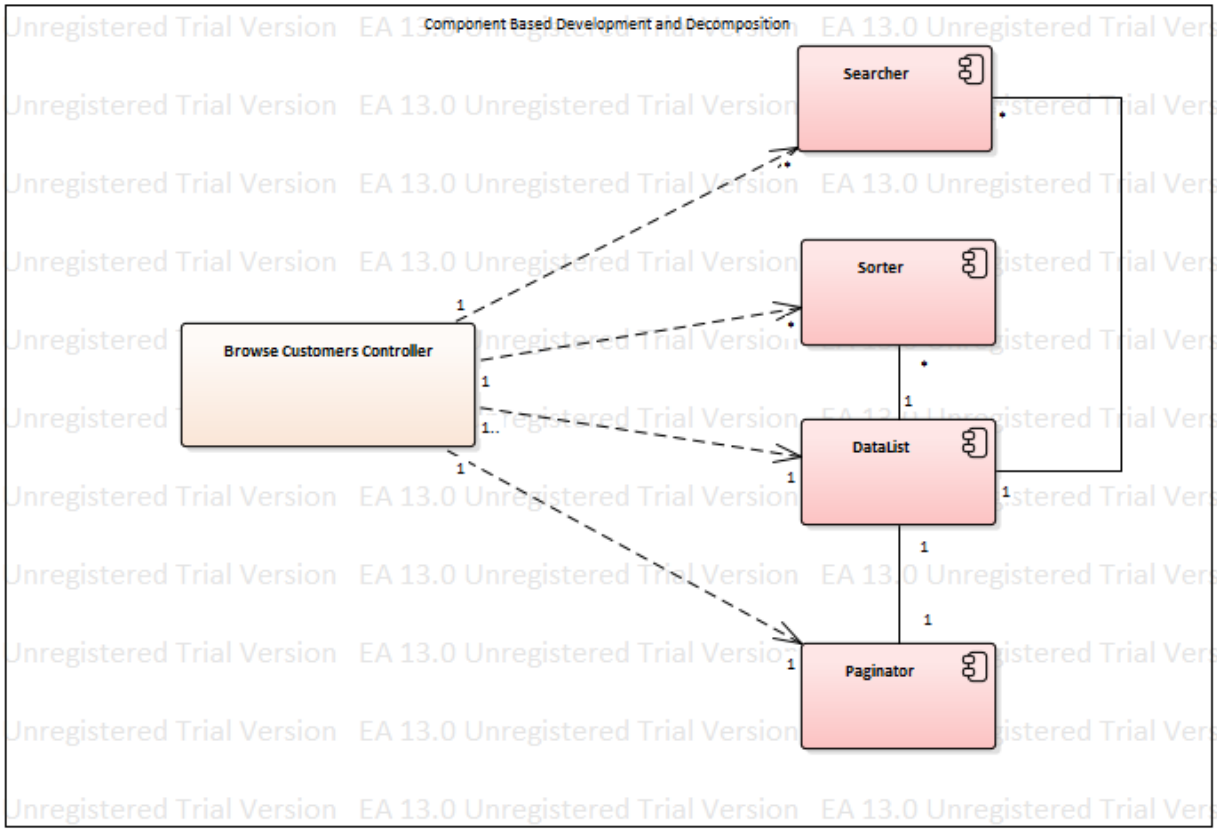
Hierarchical MVC

Framework permits developers to nesting controllers by enforcing a ***Hierarchical Content Decomposition*** concept. For example, developers can build a complex application's page by decomposing it into multiple sections implemented by controllers and nested into a single root controller representative of the HTML page. Each of child controllers is, potentially, a runnable stand-alone MVC instance and more, optionally, developers can reuse it inside a different root controller. By requesting the execution of the root controller, Framework will render it automatically, together with all its child and nested controllers. There is no limit for the nesting level of controllers into the hierarchy. It exclusively depends from the application's need or from a good decomposition analysis of the application's scenarios.
A typical example of use (and reuse) of MVC hierarchy is an ecommerce application where common section, like the toolbar, page footer and site navigation, must be present into different application's pages like browse products or product's detail. In those scenarios, the browse product page could be the main root controller, which contains its child controller's toolbar, footer and navigation. Therefore, you can reuse these controllers also by putting them into a product's detail root controller. Instantiating browse products controller, as well as product's detail, will automatically executing all its child controllers. Developers have the facilities to build and test those child controllers individually. Then they may compose the root scenarios by assembling them into controller classes designed as root.
Framework offers facilities for nesting controllers in a very simple way. In fact, developers can build a controllers' hierarchy simply by putting a special placeholder for its child controller into the View layer of the root controller. Framework will do the rest.



Component Based

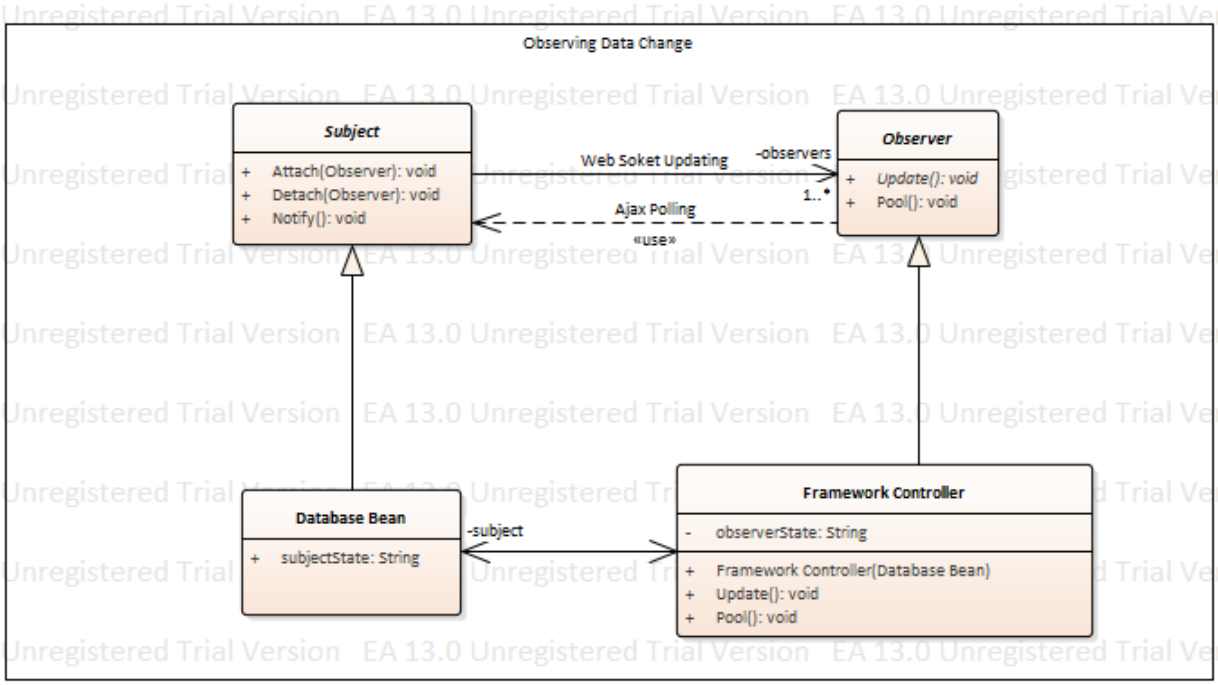
The ***Component Based Development,*** used for building many Framework's features, permits to developers another more level of applications decomposition and software reuse.
Framework's components, in fact, realizes common **Aspects** that can occurs, in a similar way, into different web applications. Many of these aspects are regarding database, for example: data listing, data listing and sorting, data listing and filtering, data listing and pagination, record management and common table's operations for select, insert, delete and update records. Framework offers a set of pre-built components for implementing the necessary server logic for these common database management aspects. These components are itself MVC objects with a Controller, are easy to use and developers can aggregate them into a root controller by using a composite criteria for building complex application pages. A component GUI can also easily adapted or replaced to reflect the application's experience simply by modifying or replacing its HTML template with a custom one. Component's server logic will remain fully reusable without the need of any source code modifications.



Observation of content change

A common limitation for web application is its intrinsic HTTP state less communication between server and client. One consequence of a state less application is the inability to update automatically a content when it change and the change is acted outside the application session (for example by another user that access to a shared database table). Although the MVC design pattern well describe the updating on content change among its layers, this aspect is hard to implement into a state less application like the web applications.

Desktop applications are not affected by this difficulty. Fortunately, for web application, AJAX can provides a valid polling solution to verify the content changing and more, the modern HTML Web Socket capabilities surely resolve this HTTP limitation. Of course, developers are able to implements their own AJAX or Web Socket hand coded solutions. Alternately, they can quickly use a special Framework feature that can do the job automatically without the need of writing any type of custom code. Controller base class was built by keeping in mind this protocol limitation. A special Controller method, *setAsObserver*, realizes the ***Observation of Content Changing*** by injecting, automatically, all necessary AJAX code into the template file of the View to do the job.

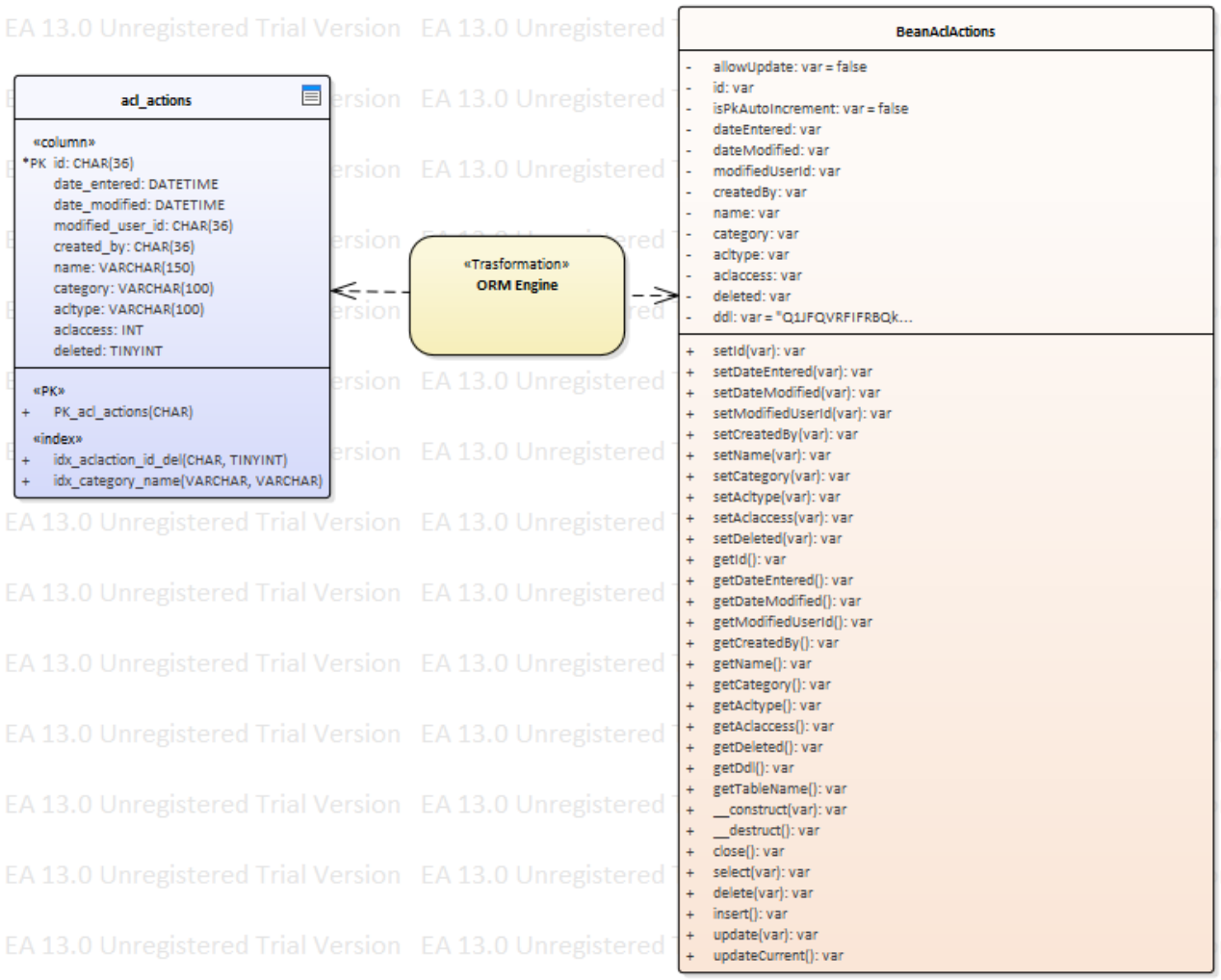


MySQL Database Integration

Framework provides a useful utility for the ***Object Relation Mapping*** of MySQL databases. The utility generates automatically Model classes for any tables of a given database schema. An auto generated Model class provides the following services:

- A constructor for managing a fetched table's row or for a adding a new one;
- Management for both single or composite Primary Keys
- Automatic mapping of the different date formats may occurs between application and database
- Destructor to automatically close database connection
- Defines a set of attributes corresponding to the table fields
- Setter and Getter methods for each attribute
- OO methods for simplify DML select, insert, update and delete operations.
- A facility for quickly updating a previously fetched row
- Useful methods to obtain table DDL and the last executed SQL statement
- Error handling of SQL statements
- Camel/Pascal case naming convention for Attributes/Class used for mapping Fields/Table
- Useful PHPDOC information about table, fields and the usage of class, attributes and methods.

Developers can quickly use these classes as Model representation of their application's MVC instances. Generates classes can also be used in conjunction with some Framework Components through a common Interface and a class Adapter provided by the Framework.
Furthermore, framework provides some generic databases classes for commons database operations.

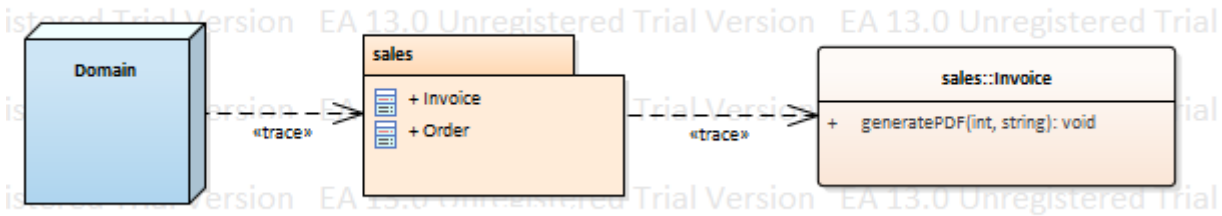


SEO friendly URL requests and auto routing

Application's user can invoke a MVC Controller and/or its methods with parameters by using a **SEO Friendly URL** notation where, controller name, methods and parameters appears separated by slashes. Framework automatically dispatches the request to the appropriate Controller by invoking the requested method and its parameters. The URL string can also contains the sub system name, which physically contains the Controller class to invoke. The following are some examples:

General format: **`http://domain/subsystem/controller/method/parameter1/parameter2`**

`http://domain/sales/invoice/genaratePDF/1/high`



Internationalization

Framework supports multi languages translations for the managed contents. Every application controllers Is able to consume external resource files containing the translations. A developer by building an application can use special placeholders in the format {RES: variable name}. Then he can put the translations for this variable into external file and Framework will load it automatically.

Security and Access Control

UNDER DEVELOPMENT - User and Roles Management, RBAC, Login/Logout

Validation

TODO - Data validations

File Management

TODO - File Open/Read/Write/Delete, File Upload/Download

Application Helpers

TODO - Email, URL Links, PDF Export, Picture management