The @Controller annotation indicates that a particular class serves the role of a *controller*

*@RequestMapping* Typically the class-level annotation maps a specific request path

The **Spring Framework** is an [open source](http://en.wikipedia.org/wiki/Open_source) [application framework](http://en.wikipedia.org/wiki/Application_framework) and [inversion of control](http://en.wikipedia.org/wiki/Inversion_of_control) container for the [Java platform](http://en.wikipedia.org/wiki/Java_platform).

Upon initialization of a DispatcherServlet, Spring MVC looks for a file named *[servlet-name]-servlet.xml* in the WEB-INF directory of your web application and creates the beans defined there, overriding the definitions of any beans defined with the same name in the global scope.

The [*DispatcherServlet*](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html) class is the [front controller](http://en.wikipedia.org/wiki/Front_Controller_pattern)[[13]](http://en.wikipedia.org/wiki/Spring_Framework) of the framework and is responsible for delegating control to the various interfaces during the execution phases of a [HTTP request](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol).

The ***Front Controller Pattern*** is a software [design pattern](http://en.wikipedia.org/wiki/Design_pattern) listed in several pattern catalogs. The pattern relates to the design of web applications. It "provides a centralized entry point for handling requests.

Front controllers are often used in web applications to implement workflows

[HandlerExceptionResolver](http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html) - Maps exceptions to views also allowing for more complex exception handling code.

[ViewResolver](http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html) - Resolves logical String-based view names to actual View types.

[LocaleResolver](http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html) - Resolves the locale a client is using, in order to be able to offer internationalized views

[ThemeResolver](http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html) - Resolves themes your web application can use, for example, to offer personalized layouts

[MultipartResolver](http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html) - Parses multi-part requests for example to support processing file uploads from HTML forms.

Abstraction is specifying the framework and hiding the implementation level information.

Abstraction reduces the complexity by hiding low level details.

Following is the sequence of events corresponding to an incoming HTTP request to *DispatcherServlet*:

1. After receiving an HTTP request, *DispatcherServlet* consults the *HandlerMapping* to call the appropriate *Controller*.

2. The *Controller* takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet*.

3. The *DispatcherServlet* will take help from *ViewResolver* to pickup the defined view for the request.

4. Once view is finalized, The *DispatcherServlet* passes the model data to the view which is finally rendered on the browser.

All the above mentioned components ie. HandlerMapping, Controller and ViewResolver are parts of *WebApplicationContext* which is an extension of the plain *ApplicationContext* with some extra features necessary for web applications.

A **persistence framework** is middleware that assists and automates the storage of program data into databases, especially relational databases. It acts as a layer of abstraction between the application and the database, typically bridging any conceptual differences between the two.

Many persistence frameworks are also object-relational mapping (ORM) tools (e.g. Hibernate, iBATIS SQL Maps, Slick, and Java Ultra-Lite Persistence). Such frameworks map the objects in the application domain to data that needs to be persisted in a database. The mappings can be defined using either XML files or metadata annotations.

 This will describes a collection of standards, conventions and guidelines for writing Java code that is easy to understand, to maintain, and to enhance.

*Target Audience*

Professional Software developers who are involved in:

Writing Java code that is easy to maintain and to enhance

Increasing their productivity

**Email hosting** is a service that runs **email** servers

Simple Mail Transfer Protocol (**SMTP**) servers handle the sending of your e‑mail messages to the Internet. The **SMTP** server handles outgoing e‑mail