Middleware:

* Middleware is the “software glue” between the operating system and applications on each side of a client-server architecture, some refer to it as the “dash” in client-server.
* I like to think of middleware as the software that provides services to applications beyond those available from the underlying operating system – it connects applications running on the server side, and passes data between them.
* Thus, middleware allows processes running on different machines to interact (where natively they would not be able to).
* We’re not going to dig deeply into the Rails middleware – we’ll focus more on how the MVC design pattern is implemented on top of it.

Rails Middleware

* In Rails, a middleware stack, called Rack, is automatically provided.
* Rack provides a unified and simple interface that allows applications to “talk to” web servers, including Mongrel, Thin, Phusion, Apache, etc. I.e., Rack is responsible for handling HTTP requests and response.
* Rack is used to group and organize modules, typically written in Ruby, and to specify the dependencies between them. Rack::Builder puts these together, creating a stack-like structure that can be used by application, from the root of the application, type:

$ rake middleware

* Other Ruby frameworks, e.g., Sinatra, are also built on top of Rack.
* In this class, we’ll use the default middleware configuration that Rails provides – but it’s useful to know what’s “under the hood”.
* When we execute:

$ rails server

A Rack::Server object is created and attached to the web server (WEBrick by default), and the middleware components are loaded up.

The Rack::Server#start method starts the web server running, listening on the designated port for HTTP requests.