1. Models are clean – only contain properties, map directly to entities and real world objects.

Views only contain presentation logic – The views only read from the models.

Controllers are the only place where services are called, therefore the only place where the persistence layer is accessed.

1. I didn’t. A database is only considered relation is referential integrity is included. The database contains no foreign keys, only has weak references to other tables in the schema. SQL code inside application code is also disgusting. Could not add stored procedures to access data.
2. By using a session that holds an instance of the OrderCollection class. This is an immutable collection that automatically orders itself by the client everytime an order is added.
3. The DD contains the uri for the persistence source along with the username and password as context parameters. Values in the DD are only accessed by the Bootstrapper class whilst setting up services.
4. I created a base class for all contollers so that common code such as getting request parameters and formatting a response as JSON is inherited. Inside the controller base class a wrapper was created so that nasty if statements to map paths to particular actions could be removed. This was achieved by creating an annotation for methods inside controllers to announce that they should be called if the path matches. Using reflection upon the inherited class inside the base class wrapper method for processing requests if looks for a method that has the ActionAttribute annotation.