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. More specifically, no business logic is in the presentation layer.

Controllers are the only place where services are called, therefore the only place where the persistence layer is accessed. Technically no business logic is done here either, it just orchestrates the facilitation of business logic by converting primitive types in domain objects are calling services than enact upon the domain objects.

1. I stored in the provided tables the values the user entered. The orders table contains a weak reference to identifier of the client and item but that number could be anything because there is no referential integrity.
2. I created a class (OrdersSessionManager) that wraps an OrdersCollection class and a Servlet Session, to keep the controller as clean as possible, alleviate “magic” strings in the session as much as possible. It has methods to retrieve the current orders in the session, clear, remove and add orders to the session collection and commit those changes at a point the controller deems appropriate. The OrdersCollection class is a sub-class of ArrayList<IOrder> that overrides the add method so that whenever an order is added the orders are ordered by their client to make it easier for the view to disseminate between clients and create the tables.
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. As well as this it contains the url mappings for controller actions to servlets.
4. I created classes for controller actions to return to delegate the actual creation of the response to the base class to process. 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, it looks for a method that has the ActionAttribute annotation that matches the HTTP method and path of the request which it then invokes.