Accepting Requests

Aviator Core Framework includes a bundled web server that can host JAX-RS REST services. If you're using web sockets to communicate, you can skip this step.

```
@POST
@Path("/zoo/animals")
@Produces(MediaType.APPLICATION_JSON)
public void addAnimal(Animal animal, @Suspended final AsyncResponse response) {
    AviatorMessage<Animal> message = new AviatorMessage<Animal>(
            new AviatorTransactionType(
                ZooDemoTransactionTypes.NAMESPACE,
                ZooDemoTransactionTypes.ADD_ANIMAL),
            animal
    );
    this.subscriberManager.registerResponder(message, ReportingEvents.transactionComplete, response);
    try {
        message.submit();
    } catch (Exception e) {
        response.resume(Response.serverError().entity(e).build());
    }
}
```

Applying Business Logic

Handlers apply business logic at specific points in the pipeline. Handlers are decorated with @AviatorHandler, indicating the transaction type and pipeline stages they are invoked for.

```
@AviatorHandler(namespace=ZooDemoTransactionTypes.NAMESPACE,
                transactionType=ZooDemoTransactionTypes.ADD_ANIMAL,
                events={PlatformEvents.executePreConsensus, PlatformEvents.executeConsensus},
                payloadClass=Animal.class)
public void addAnimal(AviatorMessage<Animal> message, SocketDemoState state) {
    //todo:  improve this so that we're testing if an animal of the same name exists, and failing if so
    Animal animal = message.payload;
    switch (animal.getSpecies()) {
        case "lion":
            state.addLion(animal.getName());
            break;
        case "tiger":
            state.addTiger(animal.getName());
            break;
        case "bear":
            state.addBear(animal.getName());
            break;
    }
}
```