Transaction Types

Each type of transaction in an ACF application is represented by a transaction type key. Keys are defined in code using annotations. At startup, the framework builds a list of valid transaction types from annotated classes.

```
@TransactionTypes(namespace=ZooDemoTransactionTypes.NAMESPACE, onlyAnnotatedValues=true)
public class ZooDemoTransactionTypes {
        public static final String NAMESPACE = "ZooDemoTransactionTypes";

        @TransactionType
        public static final String GET_ZOO = "GET_ZOO";

        @TransactionType
        public static final String ADD_ANIMAL = "ADD_ANIMAL";
}
```

Each type of transaction in an ACF application is represented by a transaction type key. Keys are defined in code using annotations. At startup, the framework builds a list of valid transaction types from annotated classes.

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());
    }
}
```