This topic concerns **multiple *EntityServers* in the same Application Server**.

You can load balance the Application Server and its *EntityServer*; that's one way to have multiple *EntityServers* in the mix. But here we are discussing something different: not multiple server instances but multiple *EntityServers* within a single Application Server instance.

An ApplicationServer hosts multiple *EntityServers* when it fields requests from clients that target different data source environments.

Many applications require different operating environments. Multi-tenant applications might maintain a separate environment per tenant. More commonly we see different environments for different phases of deployment e.g., dev, test, stage, and prod.

Some applications have multiple data sources per environment (e.g., a Customer database and an Accounting database). That leads to the cross product of environment and data source; you could have dev, test, stage, and prod environments each with its own Customer and Accounting database.

You could specify a different server address for each environment. That implies multiple deployments which might be fine for four environments. It is probably economically infeasible to deploy and maintain that many server instances in a multi-tenant application in which each tenant has its own environment, especially if you are considering deploying to the cloud. It's helpful to have the option of hosting multiple environments in the same Application Server ... and then load balancing that server.

DevForce supports that option through named environments. You identify an environment with a string called the "data source extension". It works like this:

- Each client request always targets a specific environment. The request typically comes from an EntityManager. That *EntityManager* is bound to a particular environment when it is created. You create it, explicitly or implicitly, with a particular data source extension string; henceforth all of its requests include that data source extension.
- On the server, a gateway called the EntityService inspects the request and ensures that an *EntityServer* for the requested data source extension is running. This *EntityServer* then services all subsequent requests from clients using this data source extension.