Metrics

What is a Metric?

A metric is a measure of average execution time. Each metric is given a unique name.

How to use Metrics

To add a metric to a request, just include an XML element:

```
<metric name="URL: webtools/main" />
```

to the request map and the average response time for the URL will be maintained. Likewise, to add a metric to a service, just include an XML element:

```
<metric name="Service: createMaintsFromTimeInterval" />
```

There is also a mean to measure the average durations of chained requests which were not measured on the the "Stats Since Server Start" page (ie by ServerHitBin class), eg:

```
<request-map uri="myRequest">
    <event type="java" path="org.ofbiz.MyClass" invoke="myMethod">
        <metric name="Event: component/myMethod" />
    </event>
    <response name="success" type="view" value="successPage"/>
</request-map>
```

You can access the Web Tools page for metrics from the "Stats Since Server Start" page. There is a Metrics tab were the 3 types of metrics (services, events and chained request) are in the 1st column. Using prefixes like "URL" (normal request), "Event" (chained request) and "Service" allows automated sorting of metrics in this column.

See also the autocomplete documentation and/or the site-conf.xsd file for more details, notably on calculation

Facts

- **Metrics are kept in memory.**
- The metric element has an optional threshold attribute, so some action could be taken when the metric crosses a threshold. For example, in the following request map:

  ```
  <request-map uri="ViewMetrics">
      <security https="true" auth="true"/>
      <metric name="URL: webtools/ViewMetrics" threshold="1000"/>
      <response name="success" type="view" value="ViewMetrics"/>
      <!-- displays a friendly 'server busy' page -->
      <response name="threshold-exceeded" type="view" value="ServerBusy"/>
  </request-map>
  ```

  a ServerBusy view could be rendered if the average response time exceeded 1000 mS. This can be useful for providing a lively web experience on a heavy-traffic web page.
- By default, no thresholds can be associated to services or chained requests measures (we could not think about any uses).
- A heartbeat server could retrieve the metrics to check on server load or to provide histograms.

At first glance, this might seem like a duplication of ServerHitBin.java, but it isn't. The metrics Java code is in the base component, and it is designed to be used to measure any part of the project, not just web app requests.
In a nutshell, the ServerHitBin code is quite heavy - it uses the entity engine to store the metrics. The Metric design is from SEDA - it is meant to be used in a "feedback loop" that is used to control web application response times, so it is small and fast. Also, ServerHitBin stats are kept for every request, but the metrics code is more strategic - you maintain metrics only in the areas you are concerned with.