Table of Contents

The analyzer factory creates the default analyzers like:

- StartupAnalyzer
- LoginAnalyzer
- MenuAnalyzer
- WorkScreenAnalyzer
- ErrorAnalyzer
- DownloadUploadAnalyzer

Every analyzer handles the request from the mobile app and adds one or more responses like:

- StartupAnalyzer adds ApplicationMetaData
- LoginAnalyzer adds AuthenticationData (if connected) or LoginView (if not connected) and an optional InformationView (if not connected and change password was detected)

The project uses the DefaultAnalyzerFactory but it's possible to use your own factory. To change the default factory, simply set the init parameter in your deployment descriptor: **analyzerfactory**

Set a full qualified java class and be sure that your class extends **DefaultAnalyzerFactory**.

```
<servlet>
  <servlet-name>MobileServlet</servlet-name>
  <servlet-class>com.sibvisions.rad.server.http.rest.ServerServlet</servlet-
class>

  <init-param>
    <!-- Application class name -->
    <param-name>org.restlet.application</param-name>
    <param-
value>com.sibvisions.rad.server.http.rest.ApplicationAdapter</param-value>
  </init-param>

  <init-param>
    <param-name>session-timeout</param-name>
    <param-value>10</param-value>
  </init-param>

  <init-param>
    <param-name>analyzerfactory</param-name>
    <param-value>com.sibvisions.demo.MyAnalyzerFactory</param-value>
  </init-param>

</servlet>

<servlet-mapping>
  <servlet-name>MobileServlet</servlet-name>
  <url-pattern>/services/mobile/*</url-pattern>
</servlet-mapping>
```

In your own analyzer, simply override the specific method, e.g.

```
public LoginAnalyzer createLoginAnalyzer(CommandRouter pRouter,
```

```
ApplicationRequest pRequest)
{
    return MyLoginAnalyzer(pRouter, pRequest);
}
```

The analyzer itself could look like this example:

```
public class MyLoginAnalyzer extends LoginAnalyzer
{
    protected MyLoginAnalyzer(ApplicationResource pResource,
ApplicationRequest pRequest)
    {
    super(pResource, pRequest);
    }

    @Override
    public void analyze(List<ApplicationResponse> pResponse) throws
Throwable
    {
        super.analyze(pResponse);

        //your code here
    }
}
```