HttpComponents

HTTP Components

The HttpComponents are the successor to Commons HttpClient, version 3. Incorporating the lessons learned from Commons HttpClient, they provide a modular set of functional units with specific responsibilities for various aspects of HTTP based communication.

Below, you will find a list of the functional units with a short description. But first you should read about the different levels of grouping used throughout the project. This is essential if you don't want to get confused by discussions on the developer mailing list.

Levels of Grouping

Component
Module
unit (informal)

The Component is the coarsest level of grouping. Components have their own release cycle, and each release will update everything that is in the component. Components are separate projects in JIRA, and they have their own Subversion subtree in .../httpcomponents.

The second level of grouping is the Module. A component holds one or more modules, where each module is a distinct compilation unit and will be packaged in a separate JAR. All modules of a component are updated with each release of the component. In Subversion, each module has it's own subtree in the respective component tree. Module subtrees are labelled `module-<name>`. In the JIRA project for the component, you will find a JIRA component for each module.

While component and module are groupings that affect the layout of the Subversion repository for HttpComponents, the finest level of grouping is of an informal nature. It doesn't even have an official term. We started by calling these informal units *components*, but that is also used for the release units now. In some cases, the modules correspond to informal units. But there are other cases where different informal units are joined in one module, distinguished only by Java package names. You are likely to find a JIRA component defined for an informal unit, to simplify our issue tracking. It is still up for discussion which of the informal units should become modules. It does require some effort to create a module, and in the early stages of development the code may not be separated sufficiently to put it into a distinct compilation unit.

In discussions on the mailing list, the same name is used to refer to component, module, or informal unit. For example, HttpCore or http-core is an informal unit, a distinct module, and a component that includes other modules. On the other hand, HttpClient or http-client is an informal unit and a component, but shares a module with other informal units. The context will usually disambiguate the meaning, otherwise just ask. When referring specifically to a module, its label can be used. The module HttpCore is called `module-main` in component HttpCore, whereas `module-client` in component HttpClient is shared between several informal units, including HttpClient.

List of Informal Units

HttpCore

`module-main` in component HttpCore

The core defines abstractions and provides default implementations for HTTP related stuff like messages, headers, entities and connections. It also includes a protocol execution framework based on interceptors. HttpCore has no external dependencies. We keep the source compatible with Java 1.3, but the released binaries will only be tested against Java 5.0. If you plan to use it in a Java 1.3 environment, you should build your own binaries from the source.

HttpNIO, HttpNIOSSL

`module-nio` in component HttpCore, depends on `module-main`

The NIO modules add support for non-blocking Java NIO to the core. Up to and including the release alpha6, NIO support for SSL was in a separate module because of it's dependency on Java 5. After we upgraded the Java requirement to 5.0 for everything except `module-main`, HttpNIOSSL was merged into HttpNIO.

HttpAuth

`module-client` in component HttpClient, depends on `module-main`

External dependencies: commons-codec

This informal unit provides support for HTTP authentication mechanisms like BASIC and DIGEST.

HttpCookie

`module-client` in component HttpClient, depends on `module-main`

This informal unit provides support for HTTP cookies in various flavors.

HttpConn

`module-client` in component HttpClient, depends on `module-main`

External dependencies: commons-logging

This informal unit provides support for client-side management of connections based on traditional, blocking IO.

HttpClient

`module-client` in component HttpClient, depends on `module-main`, (others in) `module-client`

External dependencies: commons-codec, commons-logging

The client module and informal unit is the replacement for HttpClient 3. It depends on module-main in component HttpCore and on the other informal units currently entangled in module-client.

HttpDispatch (fka. HttpAsync)

defunct in component HttpAsync

The dispatch unit originally went by the name of HttpAsync and was put into a separate component to have an independent release cycle. Work had to be suspended because there were (and still are) too many more important things to take care of in the other components. The non-existing informal unit was informally rebaptised because many people associate the term async with Java NIO, which does not match the intended scope of HttpDispatch. The component HttpAsync kept the name, but is currently unused.