Aliases

Deprecated since 5.2 — Use ServiceOverrides instead. Aliases will be removed starting in 5.3.

Related Articles

- Page Life Cycle
- Component Rendering
- Component Events
- Page Navigation
- Request Processing
- Component Events FAQ

See IoC Cookbook - Overriding IoC Services

Introduction

Tapestry goes to great lengths so that you can use the Inject annotation on a field and provide no additional data, yet end up with the correct object or service.

In many cases, Tapestry must match a field type to an available IoC service.

If there is only single service in the registry that implements the service, Tapestry will utilize that service.

When there is more than one such service, it is necessary to disambiguate which service is to be injected. To disambiguate globally (across all injections), you must create an alias from the service interface directly to the particular service.

This takes the form of a contribution to the Alias service.

The Alias service has additional purposes: first, it allows for spot overrides on injected services, based on the application's mode. Currently, the only mode is "servlet", but future modes may include "portlet" and possibly "offline".

Secondly, the companion AliasOverrides service configuration allows for spot overrides of specific services, without disturbing the rest of the network of services within the IoC Registry.

Contributing an Alias

To contribute a new service to the Alias service, you must first decide on a logical name. Often, this is the name of the service interface implemented by the service.

You can then contribute into the Infrastructure service's configuration:

AppModule.java (partial)

```
public static void contributeAlias(@InjectService("MyService") MyService myService,
        Configuration<AliasContribution> configuration)
{
    configuration.add(AliasContribution.create(MyService.class, myService));
}
```

The above example follows a typical pattern; the service to be vended is injected into the contributor method, using the explicit InjectService annotation. A contribution is made providing the service type.

Notice that the contribution doesn't *have* to be a service; you can just instantiate an object inside the contribution method and contribute that. That's what we're doing in the example, though we're using a create() static method rather than *new* (just to smooth out some Java Generics ugliness).

Contributing to AliasOverrides

To override a service, you need to know its service interface name.

You can then make a contribution to the AliasOverrides service configuration, as described in the previous section.

The object contributed as an override will mask the default contribution.