StrategyBuilder Service

The **StrategyBuilder Service** provides a convenient way to create an implementation of the Strategy design pattern.

Related Articles

- ShadowBuilder Service
- PipelineBuilder Service
- StrategyBuilder Service
- ChainBuilder Service

Another of the Gang Of Four patterns, the strategy pattern as implemented in Tapestry IoC is a kind of late binding.

The idea is that *adapters* for objects are accessed based on the *actual type* of an object. These adapters supply additional functionality. The adapters are located using a StrategyRegistry (API).

The lookup of adapters is based on an inheritance search; thus providing an adapter for type java.util.Map will match any object that implements the Map interface. The inheritance search works its way up the class hierarchy looking for a matching registration. If nothing is found, then all the interfaces directly or indirectly implemented by the selector class are checked. java.lang.Object is always the final match.

A runtime exception is thrown if no match can be found.

As a special case, the value null is searched for as if it were an instance of the class void.

The StrategyBuilder service (API) creates a service implementation around a strategy registry.

```
public interface StrategyBuilder
{
    <S> S build(StrategyRegistry<S> registry);
}
```

For a given interface (and matching StrategyRegistry), a service implementation is created. The service interface is determined from the strategy registry.

The first parameter of each method is the *selector*. Its type is used to locate an adapter.

The corresponding method of the adapter is then invoked, passing all parameters.

Every method of the service interface should take at least one parameter. Generally, such interfaces have only one or two methods.

Example

You will usually have a service configuration for defining the adapter registry.

You convert the configuration into a StrategyRegistry, and use that to build the final service:

```
  public static MyStrategyService build(Map<Class, MyStrategyService> configuration,
    @InjectService("StrategyBuilder")
    StrategyBuilder builder)
  {
     StategyRegistry<MyStrategyService> registry = StrategyRegistry.newInstance(MyStrategyService.class,
configuration);

     return builder.build(registry);
  }
```