Tapestry5HowToIocOnly

source: mini_app.zip

Here is a simple example on how to use only tapestry-ioc in your app.

Using IOC has many advantages, so why not get used to it even in the smallest apps.

if you are comfortable with maven: pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>tapestry.mini-app</groupId>
  <artifactId>mini-app</artifactId>
  <version>0.0.1</version>
  <build>
    <finalName>mini-app</finalName>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
          <optimize>true</optimize>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>org.apache.tapestry</groupId>
      <artifactId>tapestry-ioc</artifactId>
      <version>${tapestry-release-version}</version>
    </dependency>
  </dependencies>
  <properties>
    <tapestry-release-version>5.0.6</tapestry-release-version>
  </properties>
</project>
```

or add these libraries to your path

```
javassist-3.4.ga.jar
log4j-1.2.14.jar
tapestry-ioc-5.0.5.jar
slf4j-api-1.4.3.jar
slf4j-log4j12-1.4.3.jar
```

Your main class is fairly simple(Main.java)

```
package tapestry.mini;

import org.apache.tapestry.ioc.Registry;
import org.apache.tapestry.ioc.RegistryBuilder;

import tapestry.mini.services.Hello;
import tapestry.mini.services.MiniAppModule;

public class Main {

    public static void main(String[] args) {
        RegistryBuilder builder = new RegistryBuilder();
        builder.add(MiniAppModule.class);

        Registry registry = builder.build();
        registry.performRegistryStartup();


        Hello hello = registry.getService(Hello.class);
        hello.sayHello();

        //for operations done from this thread
        registry.cleanupThread();
        //call this to allow services clean shutdown
        registry.shutdown();
    }
}
```

MiniAppModule.java:

```
package tapestry.mini.services;

import org.apache.tapestry.ioc.ServiceBinder;

public class MiniAppModule {

    public static void bind(ServiceBinder binder){
        binder.bind(Hello.class);
        binder.bind(Output.class, OutputImpl.class);
    }
}
```

A small class that also shows automatic dependency injection: Hello.java

```
package tapestry.mini.services;

public class Hello {

    private final Output _output;

    public Hello(Output output) {
        _output = output;
    }

    public void sayHello(){
        _output.say("Hello world");
    }
}
```

a very simple dependancy: Output.java

```
package tapestry.mini.services;

public interface Output {
    public void say(String text);
}
```

A simple implementation of course: OutputImpl.java

```
package tapestry.mini.services;

public class OutputImpl implements Output{
    public void say(String text){
        System.out.println(text);
    }
}
```

It's also important to a log4j configuration file. I.e. for maven you can put it to `src/main/resources/log4j.properties`. This is an example config file:
{{{#!properties log4j.rootCategory=WARN, A1

A1 is set to be a ConsoleAppender. log4j.appender.A1=org.apache.log4j.ConsoleAppender

A1 uses [PatternLayout]. log4j.appender.A1.layout=org.apache.log4j.PatternLayout log4j.appender.A1.layout.ConversionPattern=[%p] %c{1} %m%n

log4j.category.org.apache.tapestry=error log4j.category.tapestry=error log4j.category.tapestry.ioc.ClassFactory=error

log4j.category.org.apache.tapestry5.ioc.FredModule=debug log4j.category.org.apache.tapestry5.ioc.AdviceDemoModule.Greeter=debug

log4j.category.com.example=debug log4j.category.org.apache.tapestry5.ioc.services.TapestryIOCModule.PeriodicExecutor=debug

# This makes things very verbose, but can be useful to figure what's happening and how long its taking. log4j.category.org.apache.tapestry5.ioc.
Registry=debug log4j.category.org.apache.tapestry5.ioc.RegistryBuilder=debug

log4j.category.org.apache.tapestry5.ioc.services.TapestryIOCModule.PlasticProxyFactory=debug
}}}