Quick start

Your first application

Meecrowave relies on JAX-RS and CDI so to start you just need to write a JAX-RS endpoint:

```
@Path("kitchen")
@ApplicationScoped
public class HelloKitchen {
    @GET
    public String getMenu() {
        return "good things";
    }
}
```

Then booting Meecrowave is as easy as launching:

```
try (final Meecrowave meecrowave = new Meecrowave().bake()) {
    new Scanner(System.in).nextLine();
}
```

You should get some output containing:

```
[19:54:55.397][INFO][main][.meecrowave.cxf.CxfCdiAutoSetup] REST Application: / ->
org.apache.cxf.cdi.DefaultApplication
[19:54:55.399][INFO][main][.meecrowave.cxf.CxfCdiAutoSetup]      Service URI: /kitchen
-> org.app.HelloKitchen
[19:54:55.401][INFO][main][.meecrowave.cxf.CxfCdiAutoSetup]               GET
/kitchen/ ->      String getMenu()
```

And you can check it works doing:

```
curl http://localhost:8080/kitchen
```

You're in a hurry? Use groovy!

this feature is supported starting from version 0.3.0 only.

Create a file called `hello.groovy`:

```
@Grab('org.apache.meecrowave:meecrowave-core:0.3.0')

import org.apache.meecrowave.Meecrowave

import javax.ws.rs.GET
import javax.ws.rs.Path
import javax.enterprise.context.ApplicationScoped

@Path("hello")
@ApplicationScoped
class Hello {
    @GET
    hi() {
        "hi"
    }
}

new Meecrowave().bake().await()
```

then

```
groovy hello.groovy
```

Finally you can test it:

```
curl http://localhost:8080/hello
```

Sample

https://github.com/apache/meecrowave/tree/trunk/sample module is a ready to use hello world using meecrowave.