



# Genie logiciel

---

## Deployment

Philippe Dugerdil



# Steps to create the “hello world” service

---

1. Launch Eclipse for JavaEE
2. Create a **Dynamic Web Project**
3. Copy or DragDrop the common web.xml file into  
`projectName/WebContents/WEB-INF/`
4. Populate the project with the application classes into  
`projectName/JavaResources/src/`
5. Export the project in `*.war` format
6. Deploy it on the server
7. Test it

2.

Specify the target system to get the required runtime libraries (javax.ws.rs.\*)

**New Dynamic Web Project**

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

**Dynamic Web Project**

Project name: HelloWorld

Project location

☒ Use default location

Location: C:\Users\philippe\Documents\eclipseworkspace\HelloWorld Browse...

Target runtime

Apache Tomcat v8.0 New Runtime...

Dynamic web module version

3.1

Configuration

Default Configuration for Apache Tomcat v8.0 Modify...

A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name: EAR New Project...

Working sets

☐ Add project to working sets

Working sets: Select...

? < Back Next > Finish Cancel

# Project's file hierarchy

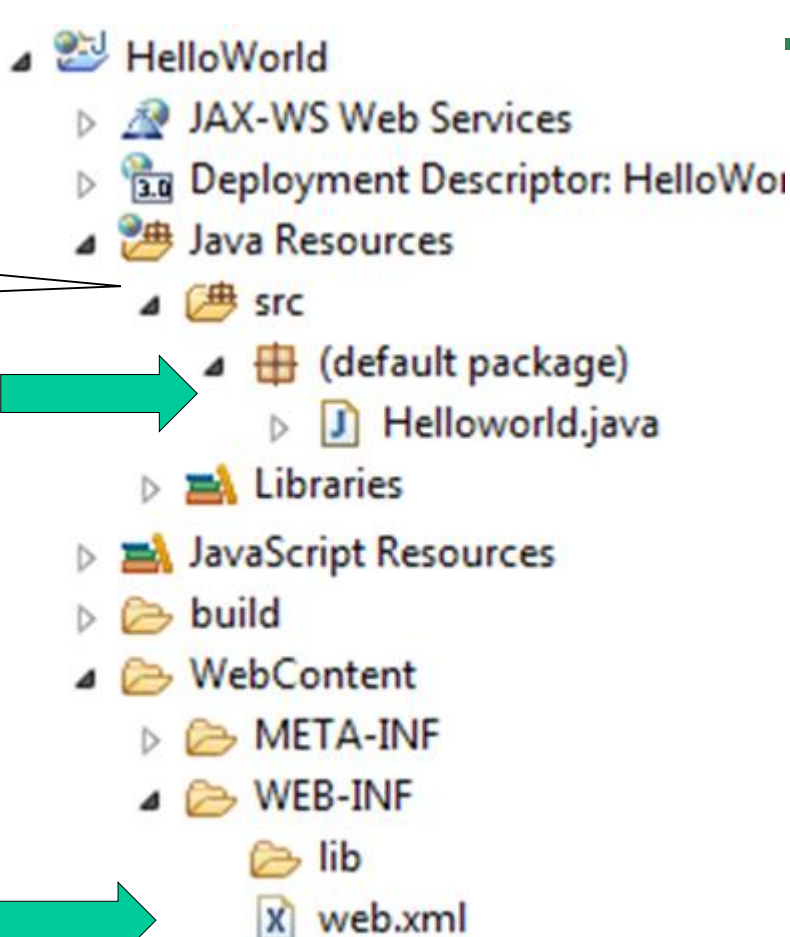


Will become the name  
of the webapp in Tomcat

Location of REST's Resource  
classes and other POJO classes

4.

3.





## 4. Application class

```
import javax.ws.rs.*;
```

Located in the TOMCAT server. This is why we must specify the target system at the web project creation

```
@Path("/")
```

```
public class Hello {
```

```
@GET
```

```
@Path("helloworld")
```

```
public String getHelloTXT() {
```

```
    return "Hello World";
```

```
}
```

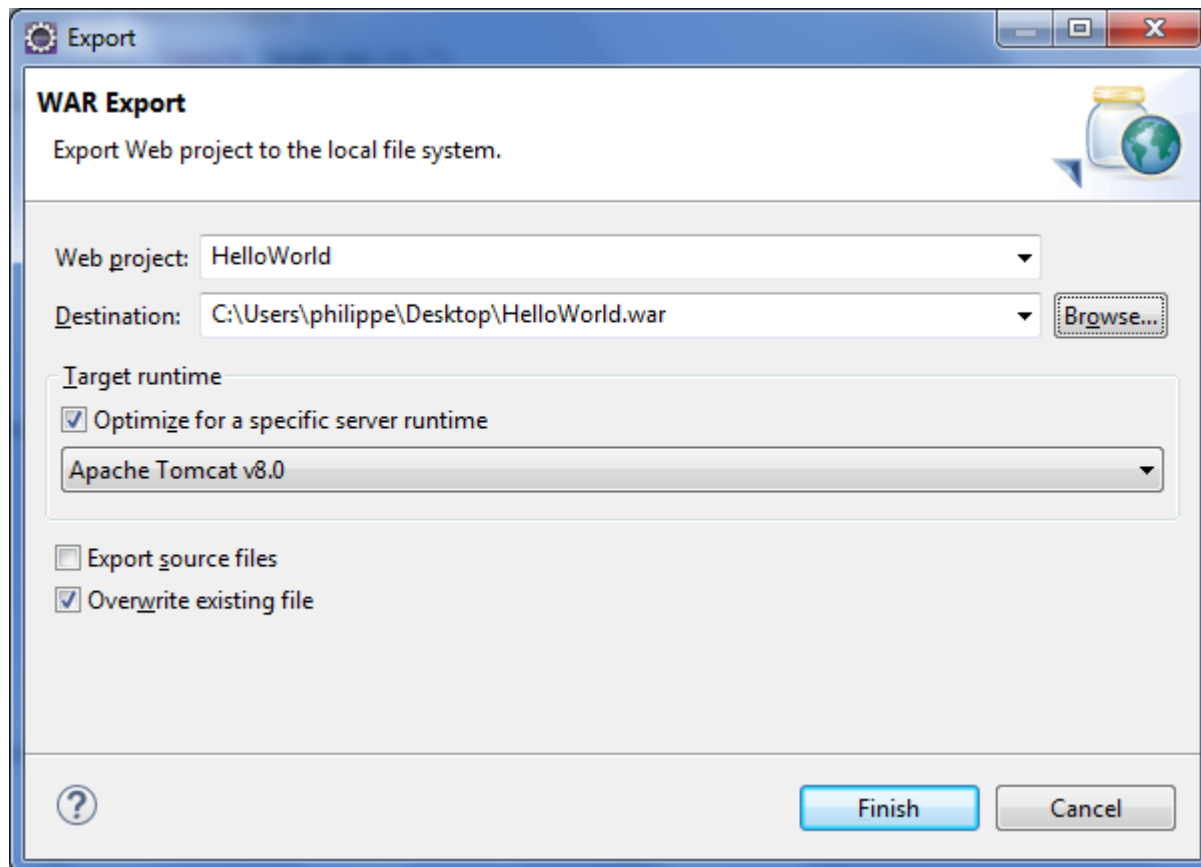
```
}
```

ProjectName = webapp name

The URL will then be *domainName/HelloWorld/helloworld*



## 5. Exporting in \*.war format





# HelloWorld.war

ZIP file with:

└─ HelloWorld

└─ META-INF

└─ MANIFEST

└─ WEB-INF

└─ classes

└─ Helloworld

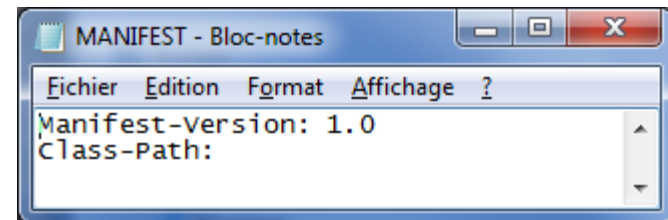
└─ lib



web



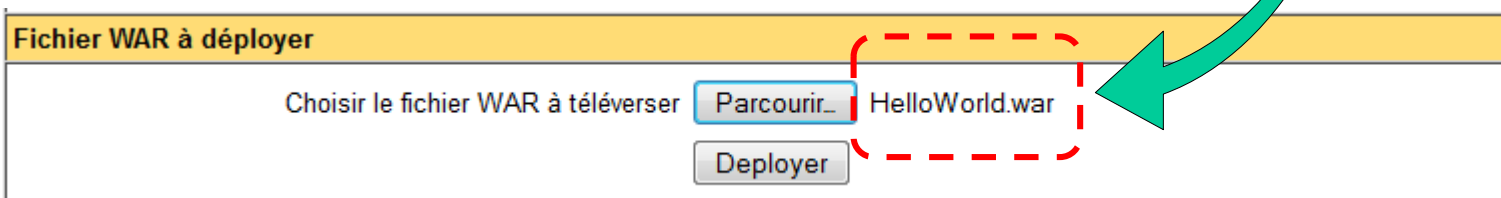
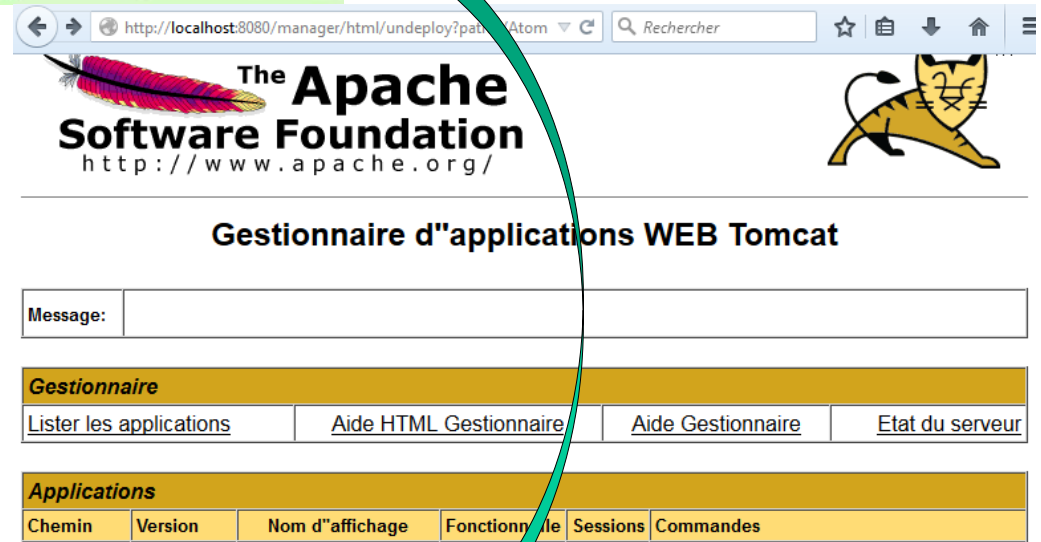
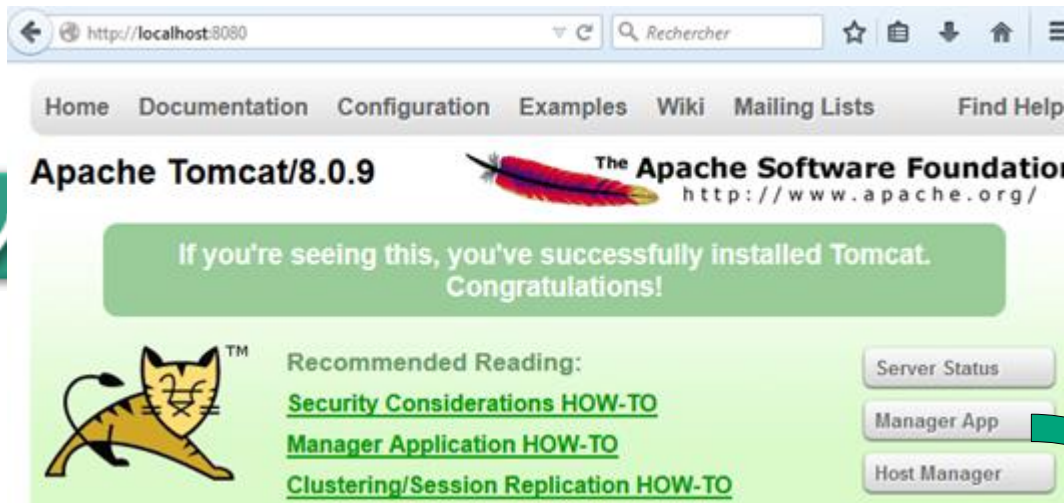
MANIFEST.MF



Helloworld.class

“standard” web.xml file

# Deployment manager







## 6. Tomcat deployment

---

- Web application can be deployed either by:
  - Static deployment by copying files to the relevant Tomcat directories. The server must then be stopped to copy the files and restarted

Or:

- Dynamic deployment using the Tomcat manager. The apps must be exported in \*.WAR format.

Or:

- Eclipse deployment: linking Eclipse to the server to deploy automatically from Eclipse

When a WAR archive is generated under Eclipse and deployed on Tomcat, the **java project name** becomes the name of the **/webapp subdirectory** (hence part of the path to the application).



# Result

## Gestionnaire d'applications WEB Tomcat

Message: OK

### Gestionnaire

[Lister les applications](#)

[Aide HTML Gestionnaire](#)

[Aide Gestionnaire](#)

[Etat du serveur](#)

### Applications

Chemin	Version	Nom d'affichage	Fonctionnelle	Sessions	Commandes
/	None specified	Welcome to Tomcat	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/HelloWorld	None specified	RESTWebApp	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes

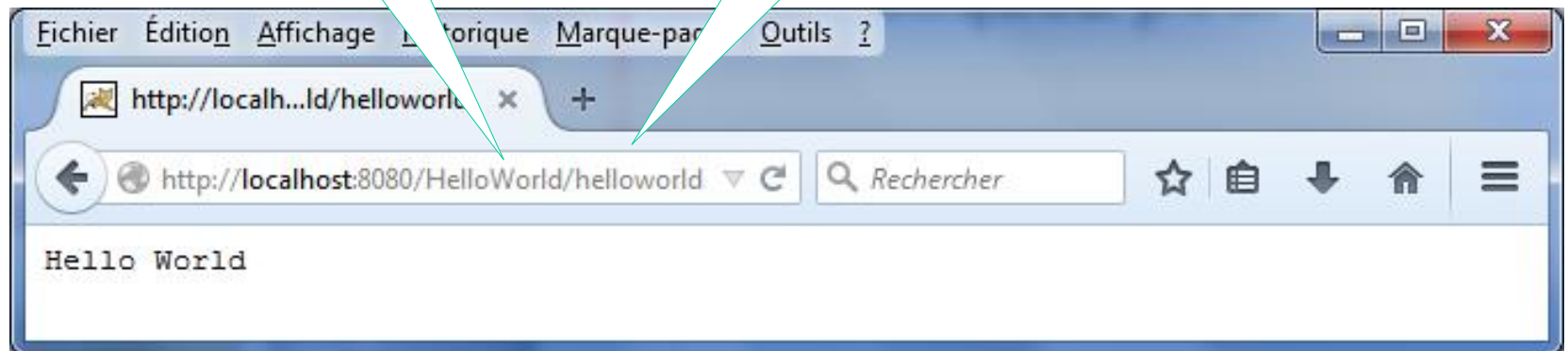
web.xml's <display-name>



# Launching the app

webapps' subdirectory name

URL segment defined at application level



Domain name: local machine  $\Rightarrow$  `localhost:8080`