Déploiement avec Spring-boot

Déploiement d'un War Spring-boot sur serveur Tomcat

Installation de Tomcat

- Télécharger et installer Tomcat 8 sur votre serveur : https://tomcat.apache.org/download-80.cgi
- Démarrer le serveur à partir du script **bin/startup** (.bat ou .sh)

Configuration du manager

- Editer le fichier conf/tomcat-users.xml :
- Ajouter le rôle **manager-gui** et 1 utilisateur le possédant

```
  <role rolename="manager-gui"/>
  <user username="admin" password="0000" roles="manager-gui"/>
```

- Redémarrer le serveur
- Accéder au Manager App à partir de l'adresse http://127.0.0.1:8080 en cliquant sur le bouton Manage app

Configuration de l'application Spring-boot

Dans le fichier **pom.xml**, modifier :

Pour déployer un fichier **war** et non un classique **jar**

```
<packaging>war</packaging>
```

Pour que le nom du war reste le même, sans intégrer le numéro de version

```
  <build>
      <finalName>${project.artifactId}</finalName>
      <plugins>
          <plugin>
              <groupId>org.springframework.boot</groupId>
              <artifactId>spring-boot-maven-plugin</artifactId>
          </plugin>
      </plugins>
  </build>
```

Ajout de la dépendance Tomcat dans dependencies

```
<dependency>
   <groupId>org.springframework.boot</groupId>
```

```
   <artifactId>spring-boot-starter-tomcat</artifactId>
   <scope>provided</scope>
</dependency>
```

Modifier la classe de l'application Spring-boot pour qu'elle hérite de **SpringBootServletInitializer** :

```
@SpringBootApplication
public class ParisH2Application extends SpringBootServletInitializer {

    public static void main(String[] args) {
        SpringApplication.run(ParisH2Application.class, args);
    }
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder)
{
        return builder.sources(ParisH2Application.class);
    }
}
```

Context-path

Le **context-path** correspond à la partie de l'url située après l'adresse du serveur et correspondant à une application.
Dans l'url **http://127.0.0.1:8080/paris-h2/joueurs**, le context-path est **/paris-h2**

Prise en compte pour les requêtes statiques en HTML

Dans le fichier template principal **header.html**, définir la base des urls :

```
<head>
    ...
    <base href="http://127.0.0.1:8080/paris-h2/">
    ...
</head>
```

Il est ensuite possible d'intégrer les ressources en spécifiant un chemin ne tenant pas compte de ce contexte :

```
    <link rel="stylesheet" href="css/style.css">
    <script src="vueJS/button.js"></script>
```

Prise en compte pour les requêtes dynamiques en java

Il est possible de récupérer le context-path dans les contrôleurs :

```
```

```
@Controller
public class TestController {
    @Value("#{servletContext.contextPath}")
    private String servletContextPath;
}
```

Intégration de vue JS

En production, utiliser la version compressée de la librairie :

```
    <script src="https://unpkg.com/vue/dist/vue.min.js"></script>
```

Générer à nouveau les composants VueJS, et copier les dans le dossier **WEB-INF/classes/static/vueJS**