Chapitre 5 : JavaBean

Ce court chapitre a pour unique objectif de vous présenter un type d'objet un peu particulier : le JavaBean. Souvent raccourci en "bean", un JavaBean désigne tout simplement un composant réutilisable. Il est construit selon certains standards, définis dans les spécifications de la plate-forme et du langage Java eux-mêmes : un bean n'a donc rien de spécifique au Java EE.

1. Pourquoi le JavaBean ?

Un bean est un simple objet Java qui suit certaines contraintes, et représente généralement des données du monde réel.
Voici un récapitulatif des principaux concepts mis en jeu. :

- **Les propriétés :** un bean est conçu pour être paramétrable. On appelle "propriétés" les champs non publics présents dans un bean. Qu'elles soient de type primitif ou objets, les propriétés permettent de paramétrer le bean, en y stockant des données.
- **La sérialisation :** un bean est conçu pour pouvoir être persistant. La sérialisation est un processus qui permet de sauvegarder l'état d'un bean, et donne ainsi la possibilité de le restaurer par la suite. Ce mécanisme permet une persistance des données, voire de l'application elle-même.
- **La réutilisation :** un bean est un composant conçu pour être réutilisable. Ne contenant que des données ou du code métier, un tel composant n'a en effet pas de lien direct avec la couche de présentation, et peut également être distant de la couche d'accès aux données (nous verrons cela avec le modèle de conception DAO). C'est cette indépendance qui lui donne ce caractère réutilisable.
- **L'introspection :** un bean est conçu pour être paramétrable de manière dynamique. L'introspection est un processus qui permet de connaître le contenu d'un composant (attributs, méthodes et événements) de manière dynamique, sans disposer de son code source. C'est ce processus, couplé à certaines règles de normalisation, qui rend possible une découverte et un paramétrage dynamique du bean !

2. Structure

Un bean :

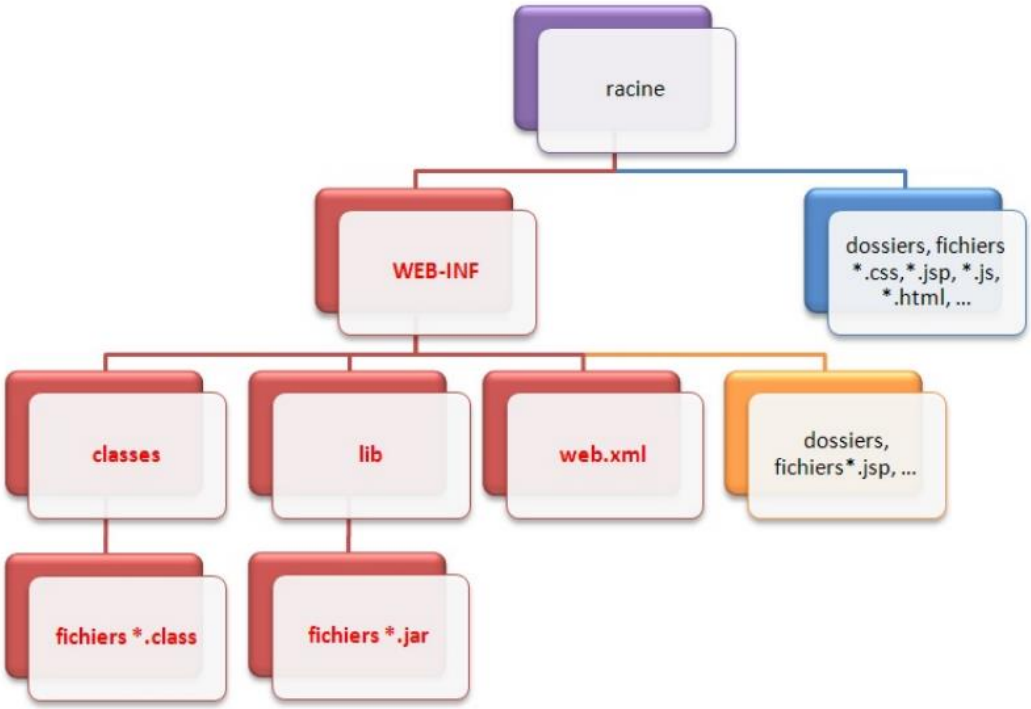
- doit être une classe publique ;
- doit avoir au moins un constructeur par défaut, public et sans paramètres. Java l'ajoutera de lui-même si aucun constructeur n'est explicité ;
- peut implémenter l'interface Serializable, il devient ainsi persistant et son état peut être sauvegardé ; ne doit pas avoir de champs publics ;
- peut définir des propriétés (des champs non publics), qui doivent être accessibles via des méthodes publiques getter et setter, suivant des règles de nommage.

Exemple :

```
package com.iad.bean;

import java.io.Serializable;

public class MonBean implements Serializable {
	private String nom;
```

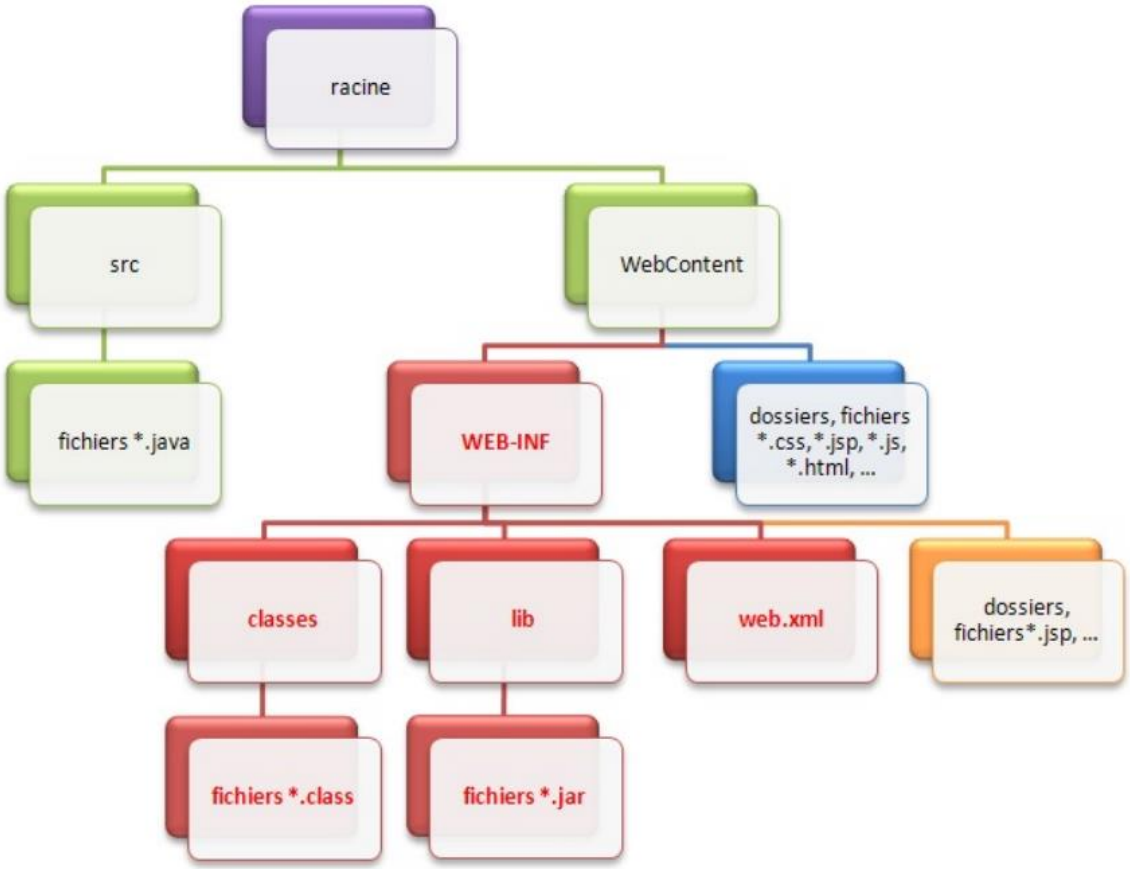
```
	private String prenom;
	private boolean admis;
	public MonBean() {

	}
	public MonBean(String nom, String prenom, boolean admis) {
		super();
		this.nom = nom;
		this.prenom = prenom;
		this.admis = admis;
	}

	public String getNom() {
		return nom;
	}

	public void setNom(String nom) {
		this.nom = nom;
	}

	public String getPrenom() {
		return prenom;
	}

	public void setPrenom(String prenom) {
		this.prenom = prenom;
	}

	public boolean isadmis() {
		return admis;
	}

	public void setadmis(boolean admis) {
		this.admis = admis;
	}



}
```

Structure standard de toutes les applications java EE



Structure des fichiers d'une application web JSP/Servlet

Structure sous eclipse :



Exemple :

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

	String paramAuteur = req.getParameter( "auteur" );
	String message = "Ma première transmission " + paramAuteur;
	req.setAttribute( "test", message );

	/*Bean*/
	MonBean monBean= new MonBean("Mohamemed","Daher",true);
	req.setAttribute("firstBean", monBean);

	this.getServletContext().getRequestDispatcher("/WEB-
INF/test2.jsp").forward(req, resp);
}
```

JSP :

```
<%@page import="com.iad.bean.MonBean"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Document</title>
</head>
<body>
<p>Ceci est une page générée depuis une JSP.</p>
<p>
<%
String attribut = (String) request.getAttribute("test");
out.println( attribut );
String parametre = request.getParameter( "auteur" );
out.println( parametre );

%>
<!-- Récuperation d'un bean -->  <br/>
<%
MonBean bean= (MonBean) request.getAttribute("firstBean");
out.println( bean.getPrenom() );
out.println( bean.getNom() );


%></p>
</body>
</html>
```