Différences entre HttpSession et @SessionAttributes

Démonstration 7 du module 4

L'objectif de cette démonstration est de comparer HttpSession et @SessionAttributes

Contexte

Compléter le projet précédent

Déroulement

2 attributs : 1 par @SessionAttributes et 1 par HttpSession

1. **Création d'un nouveau contrôleur : SessionController**

- Il va gérer toutes les url avec /session
- Il va ajouter un attribut en session SpringSession
- Il va ajouter un attribut en HttpSession
- Il va permettre de comparer leur utilisation et leur activation
- Enfin, il va permettre de supprimer les attributs.

2. **Déclaration du contrôleur**

```
@Controller
@RequestMapping("/session")
@SessionAttributes({"SpringSession"})
public class SessionController {
```

3. **Ajout d'attributs :**

- Ajout d'un attribut par le contexte de Spring
- Ajout d'un attribut par HttpSession

```
@GetMapping
public String setSessionAttribute(Model model, HttpSession http) {
        model.addAttribute("SpringSession", "Anne-Lise");
        http.setAttribute("HttpSession", "Stéphane");
        return "view-session";
}
```

- View-session.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Demo Spring Web</title>
<link rel="stylesheet" href="/css/general.css">
<link rel="stylesheet" href="/css/demo-table.css">
<link rel="icon" href="/images/logoENI.png">
</head>
<body>
	<header>
		<h1 id="nav-title">
			<a href="/">Démonstrations Spring Web</a>
		</h1>
		<nav>
			<ul>
				<li><a href="/session">Test Session</a></li>
				<li><a href="/session/invalidate">Invalidation de la Session</a></li>
			</ul>
		</nav>
	</header>
	<main>
		<h1>Attribut en Session</h1>
		<h2>@SessionAttributes : attribut SpringSession</h2>
		<p data-th-text="${SpringSession}"></p><br>

		<h2>HttpSession  : attribut HttpSession</h2>
		<!-- Pour accéder aux attributs dans HttpSession, il  faut utiliser le mot clef session -->
		<p data-th-text="${session.HttpSession}"></p><br>
	</main>
</body>
</html>
```

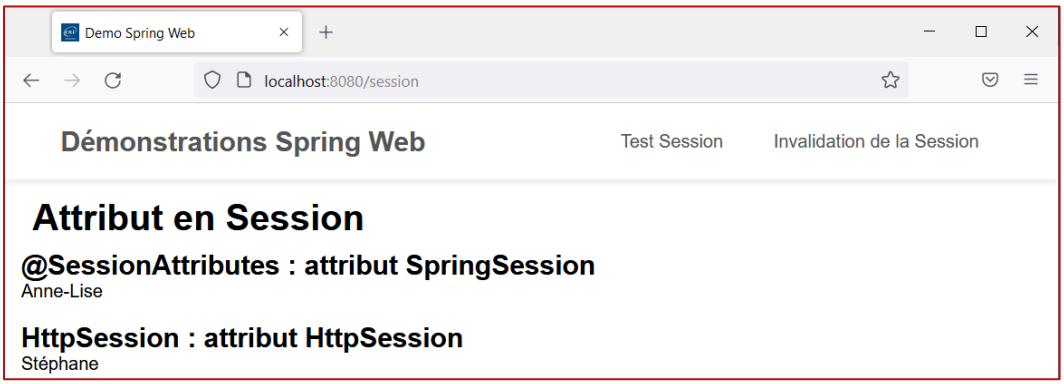
4. **Index.html :**

- Ajouter le lien suivant dans la barre de navigation de la page index.html

```
<li><a href="/session">Test Session</a></li>
```

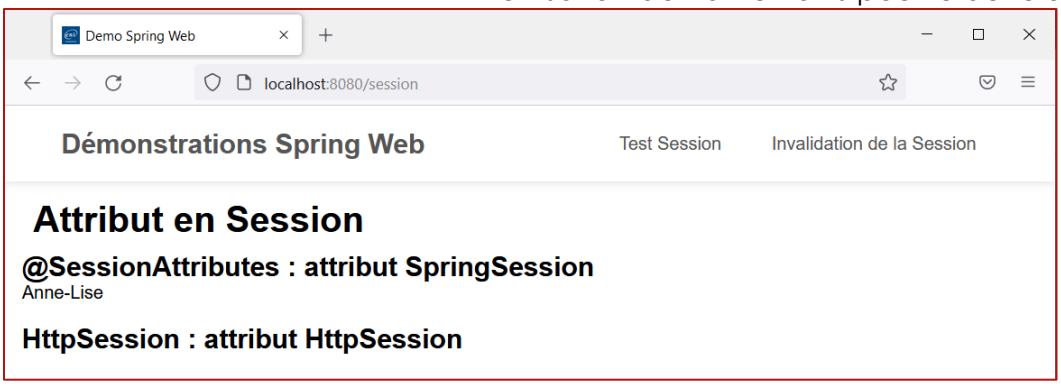
5. **Exécution**

- Les 2 attributs s'affichent correctement :



Remarque 1 :

- Pour accéder à HttpSession avec Thymeleaf, il faut utiliser le mot clef session.
  - Par défaut, seul le contexte de Spring est accessible. (Model)
  - Vous pouvez le valider en retirant le mot clef session, la donnée ne s'affiche plus.



Changer l'attribut SpringSession avec HttpSession

1. **Modification de SessionController :**

- Dans le contrôleur, ajouter une ligne pour écraser la valeur de l'attribut SpringSession par HttpSession.

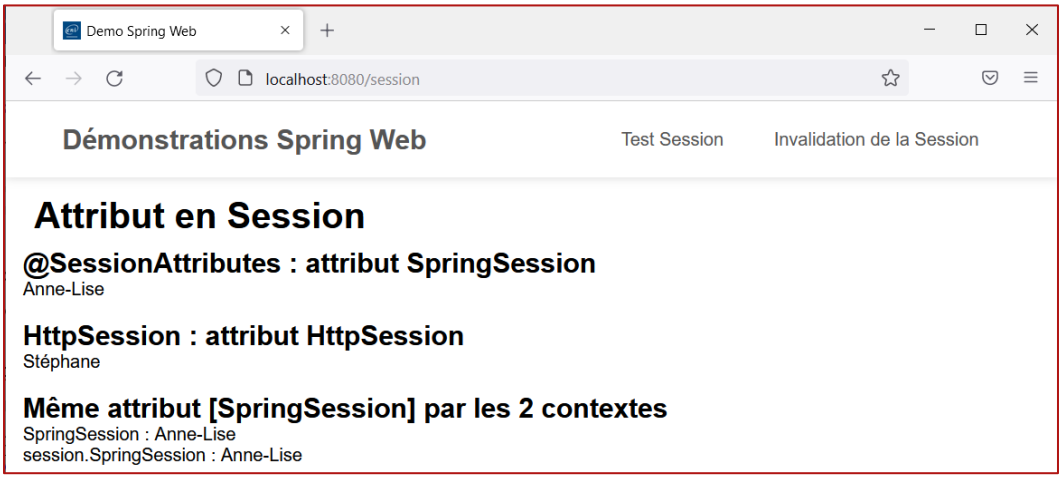
```
@GetMapping
public String setSessionAttribute(Model model, HttpSession http) {
        model.addAttribute("SpringSession", "Anne-Lise");
        http.setAttribute("HttpSession", "Stéphane");

        http.setAttribute("SpringSession", "Julien");
        return "view-session";
}
```

2. **Modification de vierw-session.html :**

- Ajout des 3 lignes suivantes, pour afficher l'attribut SpringSession avec les 2 contextes :

```
<h2>Même attribut [SpringSession] par les 2 contextes</h2>
<p data-th-text="${SpringSession}"></p>
<p data-th-text="${session.SpringSession}"></p>
```

3. **Exécution :**



Remarque 2 :

- Bien que dans le code, nous ayons écraser SpringSession avec HttpSession, c'est toujours la valeur de Spring qui est affichée dans tous les cas.
- HttpSession ajoute l'attribut immédiatement au contexte de session
- @SessionAttributes injecte les attributs en session au chargement de la vue.

- Du coup, SpringSession, vaut dans tous les cas la valeur de la ligne :

```
model.addAttribute("SpringSession", "Anne-Lise");
```

- @SessionAttributes + model.addAttribute ; injectent l'attribut en session juste avant le chargement de la page à la fin de la requête courante.

Clôture de session

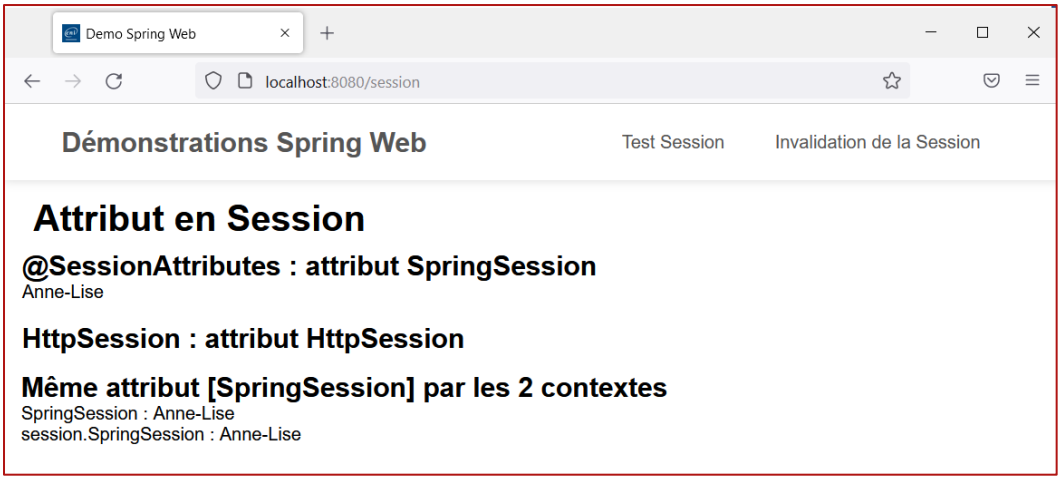
- Dans le cas de HttpSession, il faut normalement utiliser invalidate()
  - La session est nulle dès l'appel de cette méthode
- Dans le cas de Spring, pour supprimer les attributs injectés par @SessionAttributes, il faut utiliser org.springframework.web.bind.support.SessionStatus et setComplete()
  - Attention, dans le cas de Spring, il faut une redirection pour avoir un « Refresh » complet et donc la suppression des attributs
  - Sinon, vous pourrez constater qu'ils sont encore présents.

1. **HttpSession - invalidate() :**

- Modification de SessionController :
  - Ajout d'une méthode pour invalider HttpSession dans un premier temps :

```
@GetMapping("/invalidate")
public String invalidateSession(Model model, HttpSession http, SessionStatus status) {
	http.invalidate();

	return "view-session";
}
```

- Exécution :



Remarque 3 :

- Seul l'attribut « HttpSession » a disparu.
- L'attribut « SpringSession » est toujours présent, et il vaut toujours la même valeur
  - Il faut donc comprendre qu'au moment de l'affichage de la vue :
  - La session d'origine est bien invalidée d'où la disparition de l'attribut « HttpSession »

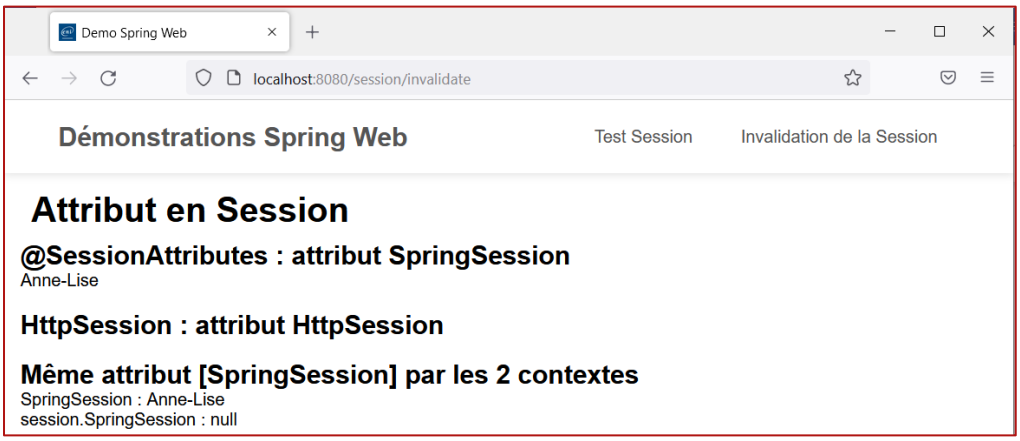
- Mais Spring a recréé une session pour injection l'attribut « SpringSession »

2. SessionStatus - setComplete() :

- Modification de la méthode invalidateSession de SessionController :
  - Appel de setComplete() :

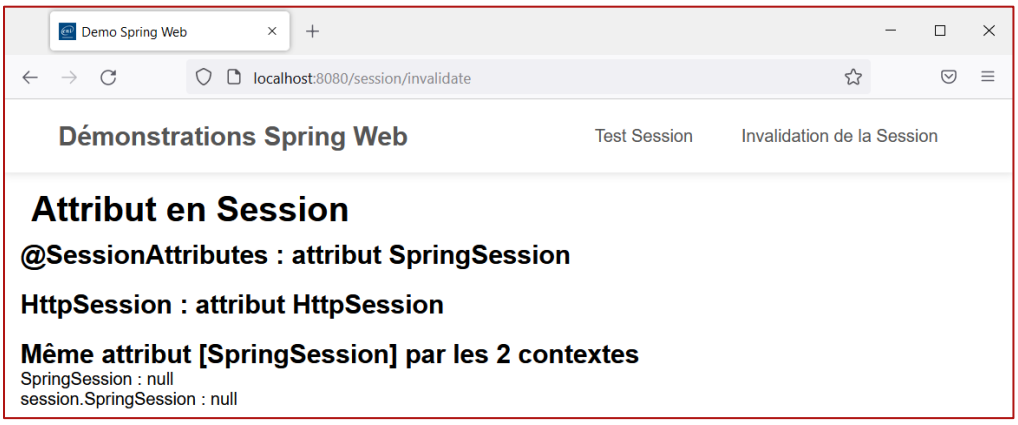
```
@GetMapping("/invalidate")
public String invalidateSession(Model model, HttpSession http, SessionStatus status) {
	//HttpSession invalidée
	http.invalidate();

	//Suppression des attributs de @SessionAttributs
	status.setComplete();

	return "view-session";
}
```

- Exécution :



Remarque 4 :

- L'attribut « HttpSession » est toujours inexistant.
- Constater que HttpSession, n'est plus capable de remonter l'attribut « SpringSession »
- Seul le Model de Spring permet encore de récupérer l'attribut
  - Il faut savoir, que Spring décharge complètement les attributs qu'après la fin de l'affichage de la vue courante.
- Faire « Refresh » de la page dans le navigateur



- A présent, tous les contextes sont vides.

3. Gestion du déchargement de contexte de Spring :

- Pour que le contexte de Spring soit vide dès l'affichage de la vue, il faut produire une redirection dans le contrôleur
- Ajout d'une méthode pour afficher la vue view-session.html
- Redirection vers cette méthode depuis la méthode invalidateSession

```
@GetMapping("/invalidate")
public String invalidateSession(Model model, HttpSession http, SessionStatus status) {
	//HttpSession invalidée
	http.invalidate();

	//Suppression des attributs de @SessionAttributs
	status.setComplete();

	//Redirection pour produire le refresh
	return "redirect:/session/refresh";
}

@GetMapping("/refresh")
public String refresh(){
	return "view-session";
}
```

Remarque 5 :

- Le fait de faire une redirection, permet de forcer le « refresh »,
- Ainsi dès l'affichage de la vue, les attributs sont déchargés complètement

4. Affichage d'un attribut Spring sur une vue d'un autre contrôleur :

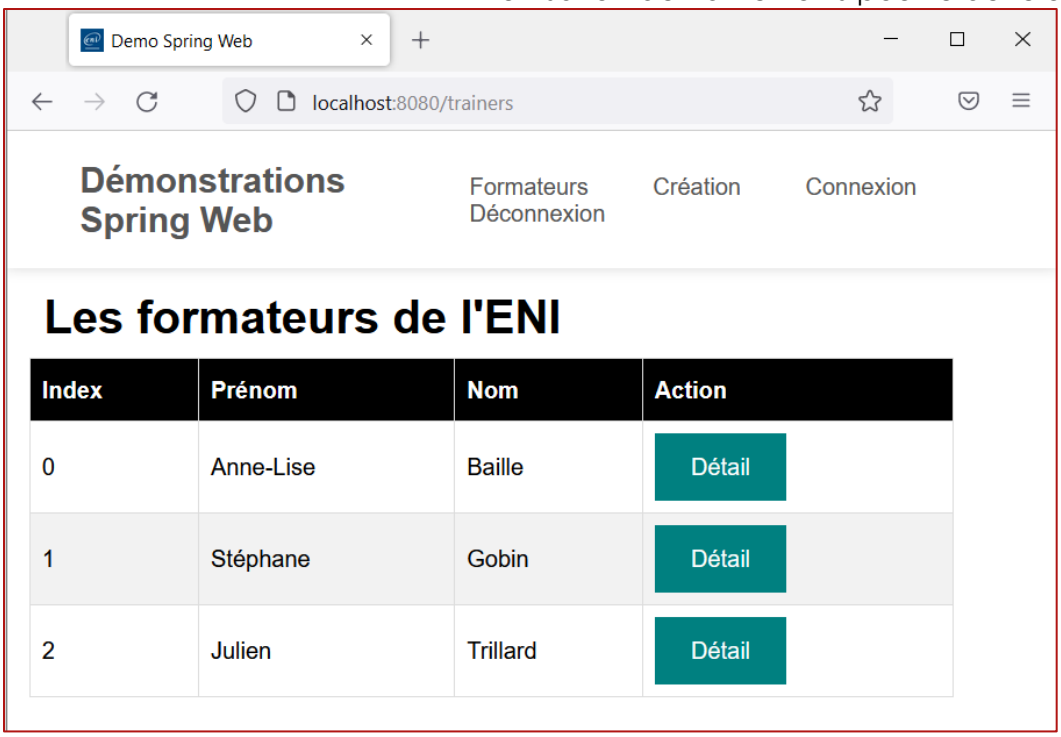
- Ajouter dans les 2 templates précédents l'affichage de l'attribut SpringSession s'il existe
  - Ajout dans l'entête des templates d'un bloc avec un test « if »

```
<h1 id="nav-title">
	<a href="/">Démonstrations Spring Web</a>
	<th:block data-th-if="${SpringSession != null}">
		<span data-th-text="${'[' + SpringSession + ']'}"></span>
	</th:block>
</h1>
```

  - Ajout dans la barre de navigation des templates des liens pour la connexion et déconnexion :

```
<nav>
	<ul>
		<li><a href="/trainers">Formateurs</a></li>
		<li><a href="/trainers/create">Création</a></li>
		<li><a href="/session">Connexion</a></li>
		<li><a href="/session/invalidate">Déconnexion</a></li>
	</ul>
</nav>
```

- Exécution :
  - Cliquer sur le lien « Test session » pour injecter l'attribut
  - Puis revenir sur la page d'accueil et cliquer sur le lien « Formateurs »



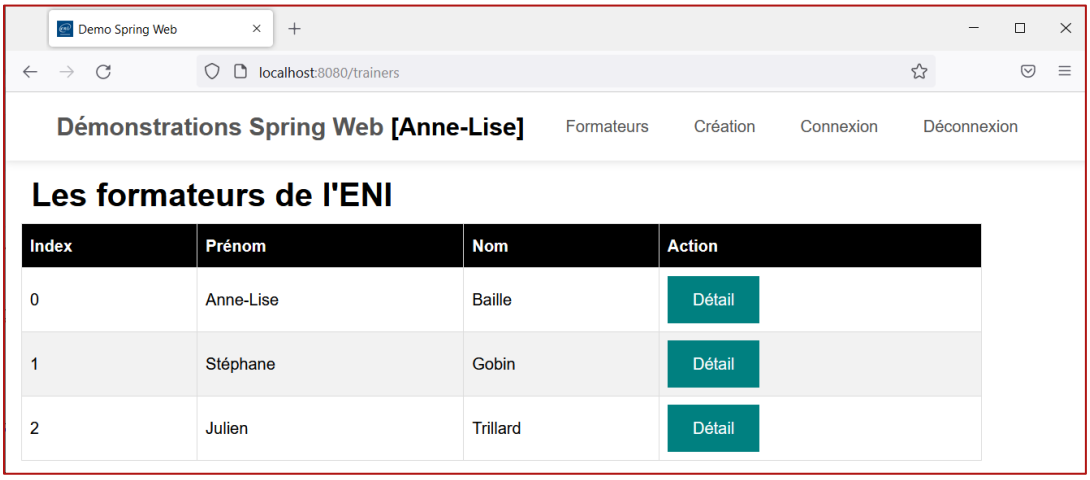
Remarque 6 :

- L'attribut en session « SpringSession » n'est pas présent pour les vues des autres contrôleurs

Ajouter dans la liste de l'annotation @SessionAttributes du contrôleur TrainerController, «SpringSession »

```
@SessionAttributes({ "allCourses", "SpringSession" })
public class TrainerController {
```

- Retester :
  - Cliquer sur le lien « Déconnexion »
  - Cliquer sur le lien « Connexion » pour injecter l'attribut
  - Puis revenir sur la page d'accueil et cliquer sur le lien « Formateurs »



Remarque 7 :

- Pour que l'attribut « SpringSession » soit affichage sur chaque vue de l'application, chaque contrôleur doit déclarer @SessionAttributes({"SpringSession" })
- De même pour récupérer l'attribut dans le Model d'un autre contrôleur, il faudra que celui-ci le déclare dans l'annotation @SessionAttributes({"SpringSession" })

Conclusion :

- Attention, dans quel contexte, vous charger les attributs.
- Pour gérer intelligemment une application Spring, il faudrait éviter de mélanger les contextes.
- Utiliser principalement @SessionAttributes
- Et penser à faire une redirection sur l'invalidation de la session, pour que l'utilisateur ne soit pas perturbé.
- Attention, si l'attribut SpringSession, veut être utilisé par plusieurs contrôleur, il faut que chaque contrôleur le déclare avec @SessionAttributes({"SpringSession" })