Spring Form Tag Library

- Apartir de la versión 2.0 Spring agregó tags para facilitar el Data Binding entre los elementos de nuestros formularios HTML.
  - ✓ Los tags de Spring para formularios facilitan el acceso a los datos del modelo. Esta característica permite desarrollar nuestros archivos JSP más fáciles de desarrollar, leer y mantener en un futuro.

Configuración

- Para usar los tags de esta librería, tenemos que agregar la siguiente directiva al archivo JSP:

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
```

- Una vez agregada esta librería, podemos usar diferentes tags para renderizar formularios HTML. Algunos ejemplos de tags de los más usados son los siguiente:
  - ✓ <form:form> → equivalente al tag <form> de HTML.
  - ✓ <form:input> → equivalente al tag <input type="text"> de HTML.
  - ✓ <form:password> → equivalente al tag <input type="password"> de HTML.
  - ✓ <form:select> → equivalente al tag <select> de HTML.
  - ✓ <form:option> → equivalente al tag <option> que va dentro de un <select> de HTML.
  - ✓ <form:textarea> → equivalente al tag <textarea> de HTML.
  - ✓ y muchos más….

Ejemplo de formulario HTML

Formulario HTML

```
<form:form action="#" method="post" enctype="multipart/form-data" modelAttribute="pelicula">
   <label for="titulo">Título</label>
   <form:input type="text" class="form-control" path="titulo" id="titulo" required="required" />
   <label for="duracion">Duracion</label>
   <form:input type="text" class="form-control" path="duracion" id="duracion" required="required" />
   <label for="clasificacion" class="control-label">Clasificacion</label>
   <form:select id="clasificacion" path="clasificacion" class="form-control">
        <form:option value="A">Clasificacion A</form:option>
        <form:option value="B">Clasificacion B</form:option>
        <form:option value="C">Clasificacion C</form:option>
   </form:select>
   <button type="submit" class="btn btn-danger" >Guardar</button>
</form:form>
```

```
@GetMapping("/create")
public String crear(@ModelAttribute Pelicula pelicula) {
    return "peliculas/formPelicula";
}
```

```
@PostMapping("/save")
public String guardar(@ModelAttribute Pelicula pelicula, BindingResult
result ... ){

   if (result.hasErrors()) {
     System.out.println("Existieron errores");
     return "peliculas/formPelicula";
   }
   . . .
}
```

NOTA: Al hacerse una petición(GET) a la URL /create, Spring MVC busca el objeto película en el modelo. Si no encuentra uno, crea una nueva instancia utilizando el constructor por default y lo agrega automáticamente al modelo.

NOTA: Al hacerse una petición(POST) a la URL /save, Spring MVC busca el objeto película en el modelo. Si lo encuentra, realiza el Data Binding entre los parámetros de la petición (inputs del form) y los asigna a cada propiedad del objeto película utilizando sus métodos setters para cada atributo.Finalmente, se agregá el objeto película automáticamente al modelo. Si existieron errores durante el Data Binding, el formulario conservará los datos capturados debido a que el objeto película existirá.

Data Binding – Utilizando objetos compuestos

```
public class Pelicula {
   private int id;
   private String titulo;
   private int duracion;
   private String clasificacion;
   private String genero;
   private String imagen = "cinema.png";
   private Date fechaEstreno;
   private String estatus="Activa";

   private Detalle detalle;

   // Setter-Getters
}
```

```
public class Detalle {
   private int id;

   private String director;

   private String actores;

   private String sinopsis;

   private String trailer;

   public Detalle() {    }

   // Getters - Setters
}
```

```
<form:form action="#" method="post" enctype="multipart/form-data" modelAttribute="pelicula">
   <label for="titulo">Título</label>
   <form:input type="text" class="form-control" path="titulo" id="titulo" required="required" />
   <label for="duracion">Duracion</label>
   <form:input type="text" class="form-control" path="duracion" id="duracion" required="required" />
   . . .
   <label for="director">Director</label>
   <form:input type="text" class="form-control"      path="detalle.director" id="director" required="required" />
   <label for="actores">Actores</label>
   <form:input type="text" class="form-control"      path="detalle.actores" id="actores" required="required" />
   . . .
   <button type="submit" class="btn btn-danger" >Guardar</button>
</form:form>
```

Tag <form:select> con objetos compuestos

Generar un <select> dinámico a partir de una lista de objetos de modelo: List<Pelicula>

Controller

```
@GetMapping(value = "/create")
public String crear(@ModelAttribute Horario horario, Model model ) {
  List<Pelicula> listaPeliculas = servicePeliculas.buscarTodas();
  model.addAttribute("peliculas", listaPeliculas);
  return "horarios/formHorario";
}
```

Formulario HTML

```
<form:select path="someProperty" items="${peliculas}" itemValue="id" itemLabel="titulo" />
```

Codigó HTML generado

```
<select name="someProperty">
   <option value="1">Power Rangers</option>
   <option value="2">La bella y la bestia</option>
   <option value="3">Contratiempo</option>
   <option value="4">Kong La Isla Calavera</option>
   <option value="5">Life: Vida Inteligente</option>
</select>
```