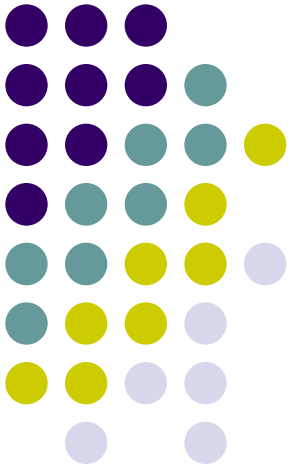
Spring

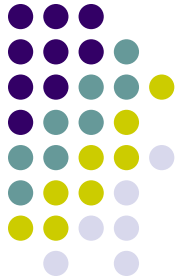
MVC



Contenidos

- Intro
- Spring MVC
- Ejemplos

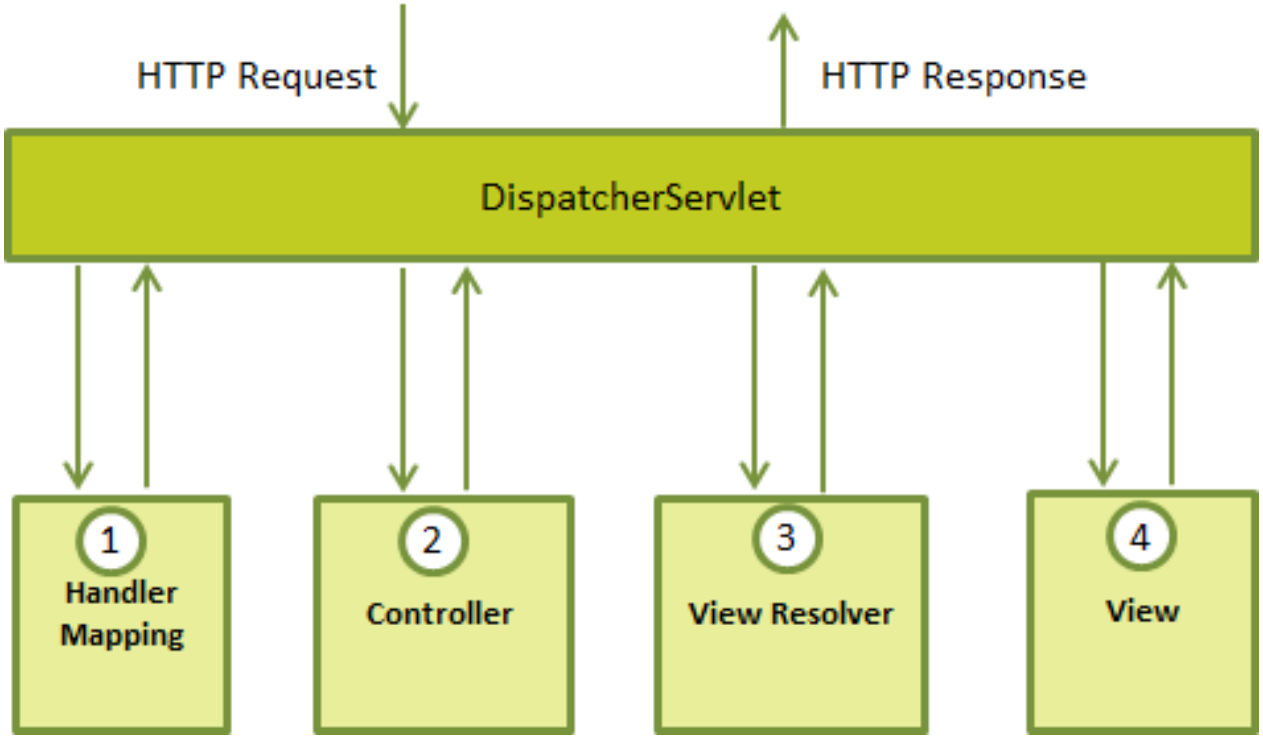
Spring MVC



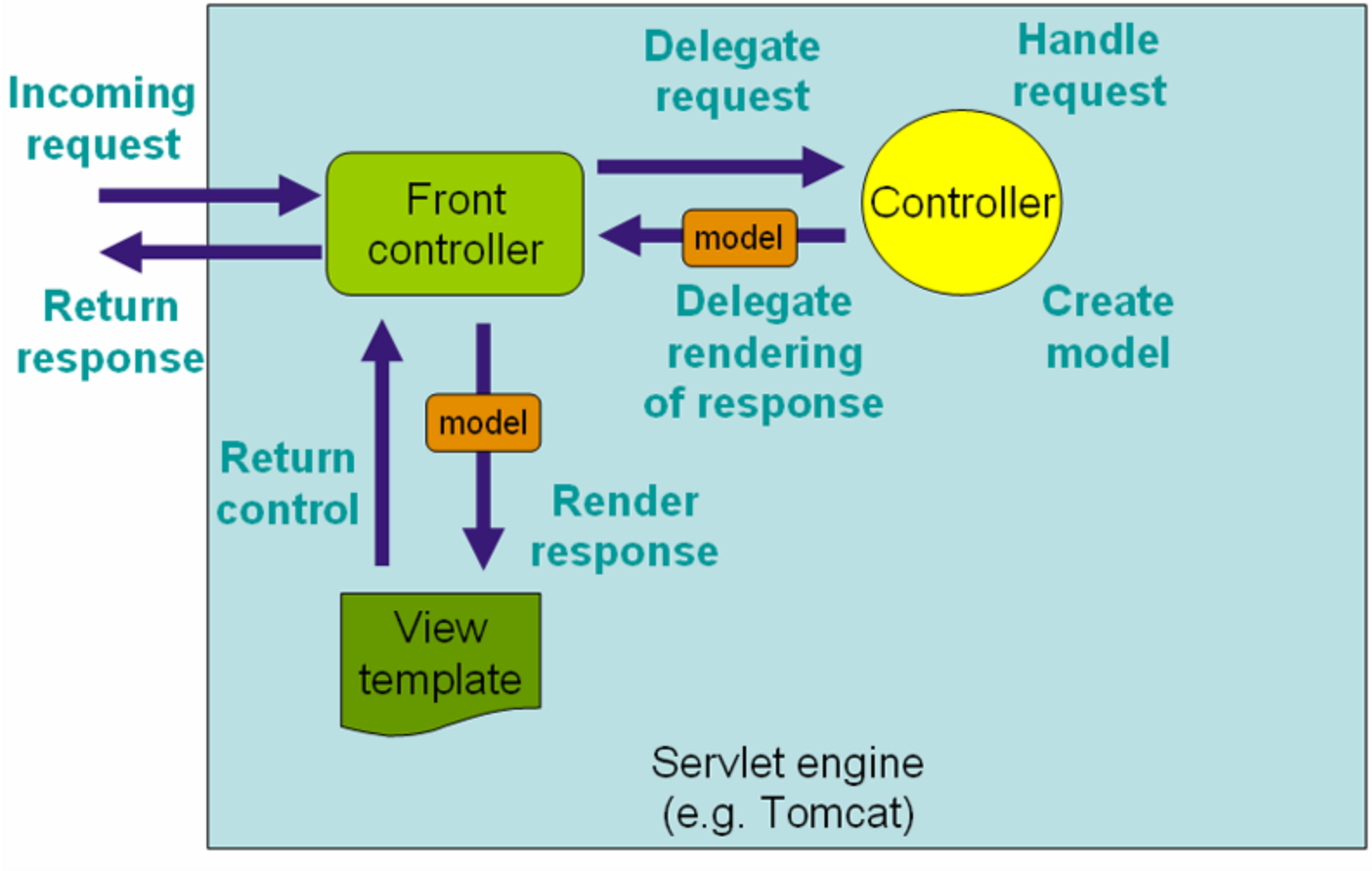
Es un framework MVC + REST, construido sobre el núcleo de *Spring*. Este framework es altamente configurable y permite el

uso de múltiples tecnologías para las vistas (JSP, Velocity, Tiles, iText). Integracion con otras herramientas web como Tapestry y Struts 1 - 2.

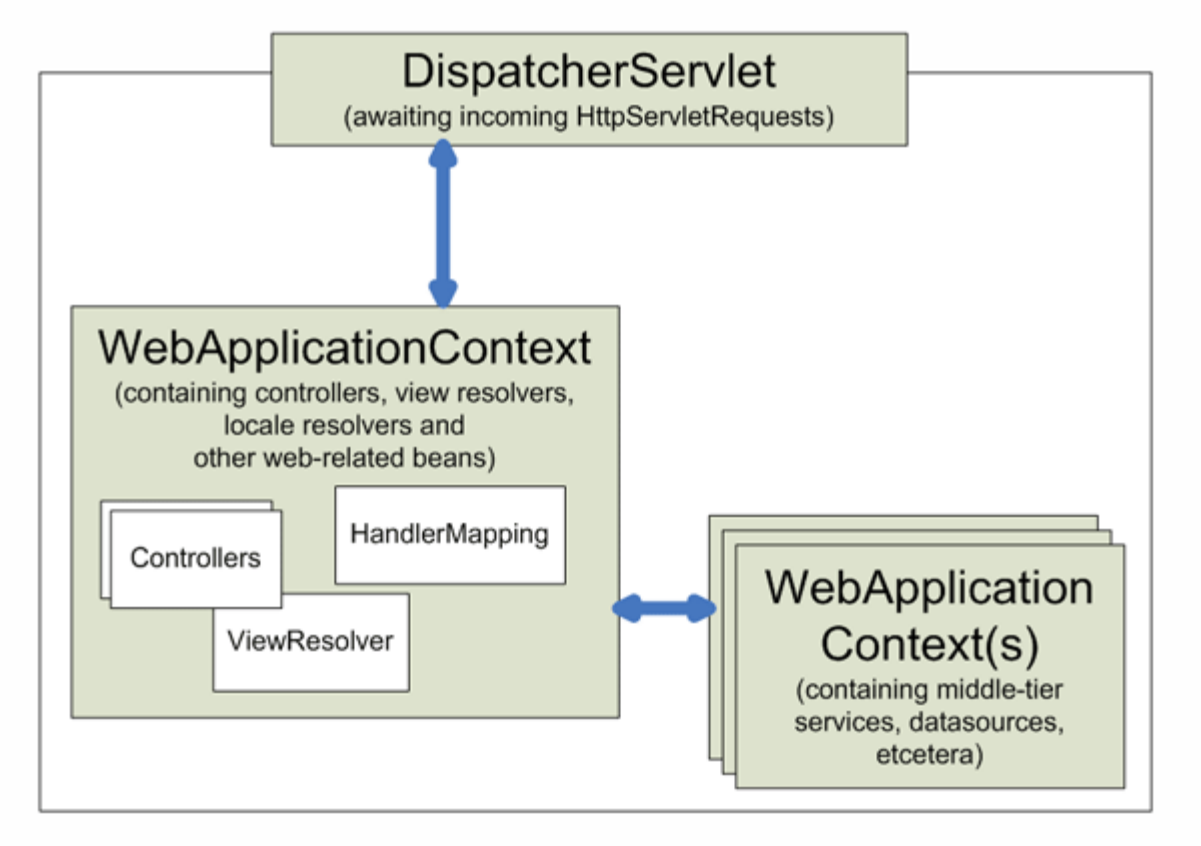
Spring MVC


HTTP Request
HTTP Response
DispatcherServlet
1
Handler Mapping
2
Controller
3
View Resolver
4
View

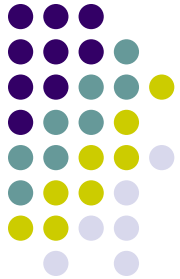
Spring MVC


Incoming request
Delegate request
Handle request
Front controller
Controller
model
Return response
Delegate rendering of response
Create model
model
Return control
Render response
View template
Servlet engine (e.g. Tomcat)

Spring MVC - Configuración

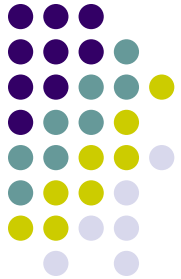


Web.xml - Configuración



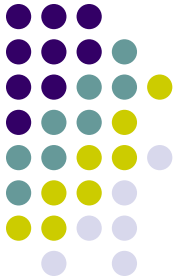
```
<web-app>

    <servlet>
        <servlet-name>disp-general</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>disp-general</servlet-name>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>

</web-app>
```

Web.xml - Configuración



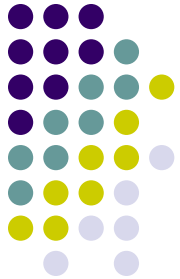
```
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>classpath:applicationContext.xml</param-value>
</context-param>

<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Dispatcher-servlet.xml - Configuración



```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <context:component-scan base-package="mipaquete.ejemplo.web"/>

    // ...

</beans>
```

Hola Mundo MVC

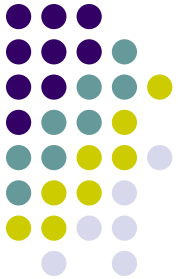


```
@Controller
@RequestMapping(value = "/holamundo")
public class HolaMundoController{

   @RequestMapping(value = "/saludar", method = RequestMethod.GET)
   public String saludarATodos(Model model) {
      model.addAttribute("textoSaludo", "Hola Spring MVC Framework!");
      return "paginasaludo";
   }

}
```

Hola Mundo MVC

```
//paginasaludos.jsp
<html>
  <head>
    <title>Hola</title>
  </head>
  <body>
    <h2>${textoSaludo}</h2>
  </body>
</html>
```

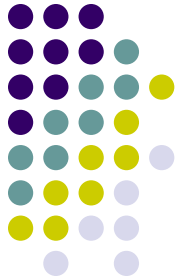
Controladores



@Controller
@RequestMapping(value="/personas")
public class PersonasController { ….

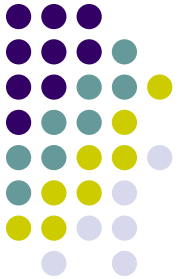
```
@RequestMapping(value="/listar")
public String listar(Model model)  {
   List<Producto> productos = productoService.buscarProductos();
   model.addAttribute("productos", productos);
   return null;
}
```

Métodos



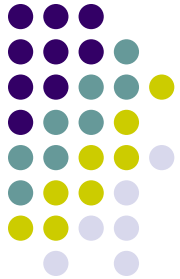
```
@Controller
@RequestMapping(value="/personas")
public class PersonasController { ….

        @RequestMapping(value="/inicio")
        public String index(Model model)  { ……

        @RequestMapping(value="/listar")
        public String recuperarLista(Model model)  { ……
```

Redirect



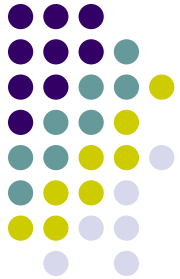
```
@RequestMapping…….
public String confirmar(@RequestParam long idProducto, Model model) {
  Producto producto = productoService.buscarProducto(id);
  model.addAttribute("producto", producto);
  return "redirect:/productos/ventas.html";
}
```

URL Template - RequestParam



```
@RequestMapping…….
public String ver(@RequestParam long idProducto, Model model) {
  Producto producto = productoService.buscarProducto(id);
  model.addAttribute("producto", producto);
  return …….;
}

@RequestMapping……..
public String ver(@RequestParam("id") long idProducto, Model model)  {
  Producto producto = productoService.buscarProducto(id);
  model.addAttribute("producto", producto);
  return ……..;
}
```

URL Template - RequestParam



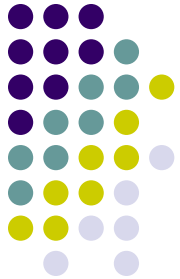
@RequestMapping……..

```
public String ver(@RequestParam(value="id", defaultValue="0", required=false) long
idProducto, Model model)  {
  Producto producto = productoService.buscarProducto(id);
  model.addAttribute("producto", producto);
  return null;
}
```

@RequestMapping……..

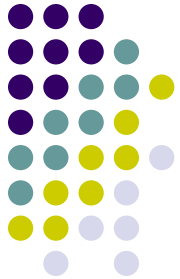
```
public String ejemploGeneral(@RequestParam String texto, HttpSession session,
WebRequest request,Model model)  {
.....
}
```

URL Template - PathVariable



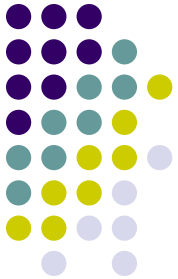
```
@RequestMapping(value="/owners/{ownerId}",
method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model)
{
  Owner owner = ownerService.findOwner(ownerId);
  model.addAttribute("owner", owner);
  return "displayOwner";
}
```

URL Template - PathVariable



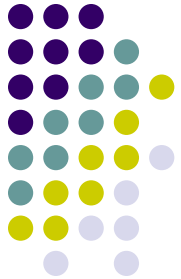
```
@RequestMapping(value="/owners/{ownerId}/pets/{petId}",
method=RequestMethod.GET)
public String findPet(@PathVariable String ownerId, @PathVariable
String petId, Model model) {
  Owner owner = ownerService.findOwner(ownderId);
  Pet pet = owner.getPet(petId);
  model.addAttribute("pet", pet);
  return "displayPet";
}
```

Form Handling



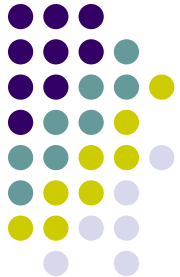
```
public class PersonaForm {
   private String nombre;
   private Long id;

   public void setNombre(String nombre) {
      this.nombre = nombre;
   }
   public String getNombre() {
      return nombre;
   }

   public void setId(Long id) {
      this.id = id;
   }
   public Long getId() {
      return id;
   }
}
```

Form Handling



```
@RequestMapping(value = "/nuevaPersona", method = RequestMethod.GET)
  public String nuevaPersona(Model model) {
     model.addAttribute("personaForm", new PersonaForm());
     return ....;
  }

  @RequestMapping(value = "/guardarPersona", method = RequestMethod.POST)
  public String guardarPersona(@ModelAttribute("personaForm")PersonaForm
personaForm,
  Model model) {
     .......
     return .....;
  }
```

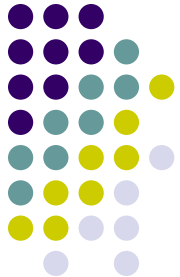
Form Handling



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

```
<form:form method="POST" action="guardarPersona" commandName="personaForm">
   <form:hidden path="id" />
   <table>
    <tr>
        <td>Nombre</td>
        <td><form:input path="nombre" /></td>
    </tr>
    <tr>
        <td colspan="2">
            <input type="submit" value="Submit"/>
        </td>
    </tr>
</table>
```

Form Handling – Tags



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

<form:input path="userName" />

<form:password path="password" />

<form:textarea path="address" rows="5" cols="30" />

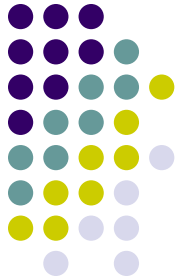
<form:hidden path="secretValue" />

<form:checkbox path="receiveNewsletter" />

<form:checkboxes items="${webFrameworkList}" path="favFramework" />

<form:checkboxes items="${webFrameworkList}" itemLabel="nombre" itemValue="id" path="idsList" >

Form Handling – Tags



<form:radiobutton path="sex" value="M"/>Male

<form:radiobuttons path="favNumber" items="${numberList}" />

<form:radiobuttons path="idsList" items="${numberList}" itemLabel="nombre" itemValue="id" />

<form:select path="idCategoriaProducto" items="${categorias}" itemLabel="nombre" itemValue="id">

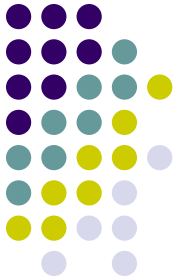
<form:errors path="nombre" />

<form:form *method="POST" action="save.html"*>

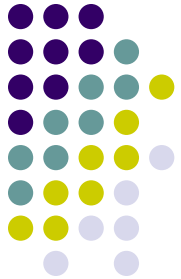
<form:form commandName=*"miForm" method="POST" action="save.html"*>

@Valid

```
public class CustomerForm {

        @NotEmpty //make sure name is not empty
        String name;

        @Range(min = 1, max = 150) //age need between 1 and 150
        int age;

        //getter and setter methods

}
```



@Valid

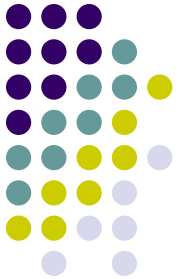


```
@Controller
@RequestMapping("/customer")
public class SignUpController {

        @RequestMapping(value = "/signup", method = RequestMethod.POST)
        public String addCustomer(@Valid CustomerForm customerForm, BindingResult result) {
                if (result.hasErrors()) {
                        return "SignUpForm";
                } else {
                        return "Done";
                }
        }

        @RequestMapping(method = RequestMethod.GET)
        public String displayCustomerForm(ModelMap model) {
                model.addAttribute(" customerForm ", new CustomerForm());
                return "SignUpForm";
        }
}
```

@Valid

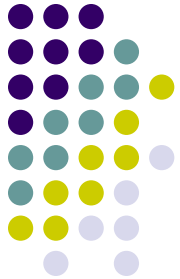
```
<form:form method="POST" commandName="customerForm" action="customer/signup">
        <form:errors path="*"  element="div" />
        <table>
                <tr>
                    <td>Customer Name :</td>
                    <td><form:input path="name" /></td>
                    <td><form:errors path="name" cssClass="error" /></td>
                </tr>
                <tr>
                     <td>Customer Age :</td>
                    <td><form:input path="age" /></td>
                    <td><form:errors path="age" cssClass="error" /></td>
                </tr>
                <tr>
                    <td colspan="3"><input type="submit" /></td>
                </tr>
        </table>
</form:form>
```

ViewResolver + JSON / XML



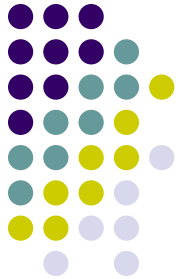
```
@ResponseBody
@RequestMapping(value="/ver/{id}", method=RequestMethod.GET)
public Cliente ver(@PathVariable long id)  {
        .......
        return clienteX;
}
```

ViewResolver + JSON / XML



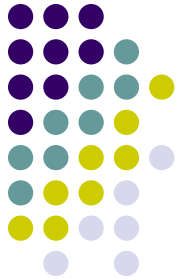
```
@ResponseBody
@RequestMapping(value="/listar", method=RequestMethod.GET)
public List<CategoriaProducto> listar()  {
        return categorias;
}
```

ViewResolver + JSON / XML



```
@ResponseBody
@RequestMapping(value="/guardar", method=RequestMethod.POST)
public Long guardar(@RequestBody Cliente cliente)  {
…..
}
```

El lenguaje de expresiones EL

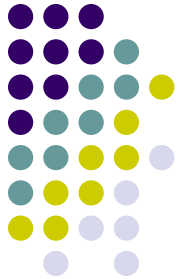


El lenguaje de expresiones EL simplemente define un poderoso mecanismo para expresar expresiones simples en una sintáxis muy sencilla.

Su combinación con las etiquetas de las 4 librerías antes mencionadas proveen mucha flexibilidad y poder para el desarrollo de páginas dinámicas.

En EL las expresiones están delimitadas por ${ }.

El lenguaje de expresiones EL



Algunos ejemplos del uso de EL son:
${anExpression}
${aList[4]}
${anObject.aProperty} acceso a la propiedad de un objeto
${anObject["aPropertyName"]} entrada en un mapa con propiedad aPropertyName

Existen una serie de variables implícitas definidas en EL:

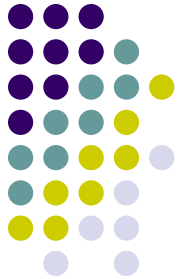
pageScope, requestScope, sessionScope, y applicationScope: colecciones de mapas que mapean nombres de variables en esos contextos a valores
param and paramValues: parámetros pasados con la petición de la página, lo mismo que en JSP
header and headerValues: cabeceras pasadas en la petición de la página
cookie: mapa que mapea nombres de cookies a los valores de las mismas

Librería core

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

Permiten llevar a cabo las siguientes acciones:

-Visualizar
-Asignar valores
-Control de flujo
-Otras acciones de utilidad



Librería core

Para visualizar valores utilizamos:

```
<c:out value="${var}" default="0"  />
```

Asignar una variable en una página:

```
<c:set var=“clienteID" value=“4507" scope="session" />
```

scope indica el contexto en el que se define la variable

```
<c:set var=“contenidoCell">
      <td>
        <c:out value="${texto}"/>
      </td>
</c:set>
```

Remueve una variable

```
<c:remove value="${texto}“ />
```

Librería core

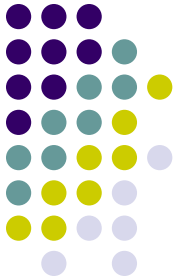
Para llevar a cabo simples condiciones if:

```
<c:if test="${costo >= 1000000}">
   Es muy caro el producto
</c:if>
```

En el caso del switch se puede emular con

```
<c:choose>
   <c:when test="${item== 'book'}">
     ...
   </c:when>
   <c:when test="${item== 'electronics'}">
     ...
   </c:when>
   <c:otherwise>
     ...
   </c:otherwise>
</c:choose>
```

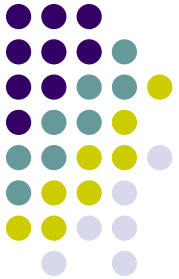
Librería core



Para iterar sobre una colección se define c:foreach. Se pueden especificar índice de comienzo, final e incremento con los atributos begin, end y step.

```
<table>
<c:forEach var=“nombre" items="${nombres}">
      <tr><td><c:out value="${nombre}"/></td></tr>
</c:forEach>
</table>
```

Librería core



Para codificar URLs se puede utilizar c:url:

```
<c:url value=“/productos/ver.jsp" var=“urlProducto">
   <c:param name=“id" value="${produtoID}"/>
</c:url>

<a href='<c:out value="${urlProducto}"/>'>Ver producto</a>
```