Siempre deberemos crear primero la clase, después el controlador, a continuación, el formulario de registro y por último la visualización de los datos.

El modelo sirve de puente entre el controlador y las vistas

**JSP Tags**: Separan Java de HTML, se integran en HTML, minimizan el código, hacen un código reutilizable y con un mantenimiento más sencillo

Validaciones:

Para poner en castellano los errores comunes (p.e. TypeMismatch) debemos crear una carpeta llamada **resources** que contenga un fichero **messages.properties** donde pondremos el tipo de error junto al campo que lo puede producir y lo igualamos al mensaje que queremos obtener:

typeMismatch.elAlumno.edad=Valor numérico no válido

Para crear nuestra propia validación (*poner el validador con arroba delante del campo*) tendremos que crear un paquete que contenga nuestros ficheros de validación dentro del nivel del paquete que contiene nuestro programa. En su interior crearemos un interface que contenga la información a controlar (mensaje, comprobador, etc) y una clase que comprobará la validación:

***CPostalSevilla ->***

@Constraint(validatedBy=CPostalSevillaValidacion.**class**) //futura clase que contendrá la lógica de la validacion

@Target({ElementType.***METHOD***,ElementType.***FIELD***}) //Destino de nuestra validación a métodos o campos

@Retention(RetentionPolicy.***RUNTIME***) //Cuándo comprobaremos si la comprobación se cumple (en tiempo de ejecución, en este ejemplo)

**public** **@interface** CPostalSevilla {

// Definir el CP por defecto

**public** String value() **default** "41";

// Definir el mensaje de error

**public** String message() **default** "El CP debe comenzar por 41";

// Definir los grupos

Class<?>[] groups() **default** {};

// Definir los payloads

Class<? **extends** Payload>[] payload() **default** {};

}

***CPostalSevillaValidacion ->***

**public** **class** CPostalSevillaValidacion **implements** ConstraintValidator<CPostalSevilla, String>{

@Override

**public** **void** initialize(CPostalSevilla elCodigo) {

prefijoCodigoSevilla=elCodigo.value();

}

@Override

**public** **boolean** isValid(String arg0, ConstraintValidatorContext arg1) {

**boolean** valCodigo;

**if**(arg0!=**null**) valCodigo=arg0.startsWith(prefijoCodigoSevilla);

**else** **return** valCodigo=**true**;

**return** valCodigo;

}

**private** String prefijoCodigoSevilla;

}

Tras esto sólo tendríamos que incluir nuestra validación sobre el campo donde queramos aplicarla, por ejemplo, en la clase Alumno podríamos poner:

@CPostalSevilla

**private** String codigoPostal;