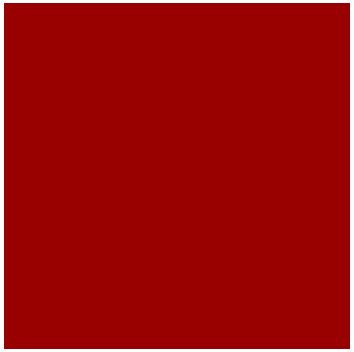
Aplicaciones Open Source

Semana09
19/05/18

Carlos A. Quinto Cáceres
pcsicqui@upc.edu.pe

Agenda

- Spring
- Thymeleaf.



Logros del día

- Al final de la clase, los alumnos podrán:
  - Definir el framework Spring, sus componentes y la utilidad de uso.
  - Hacer uso de Spring para el desarrollo de proyectos.
  - Conocer la sintaxis de Thymeleaf.

Spring

Framework - Definición

- Un framework es un marco de trabajo que facilita el desarrollo de software.
- Proporciona un esqueleto, patrón que el programador debe seguir, preocupándose únicamente de la codificación.
- Spring es un framework basado en Java orientado a aplicaciones de gran magnitud.

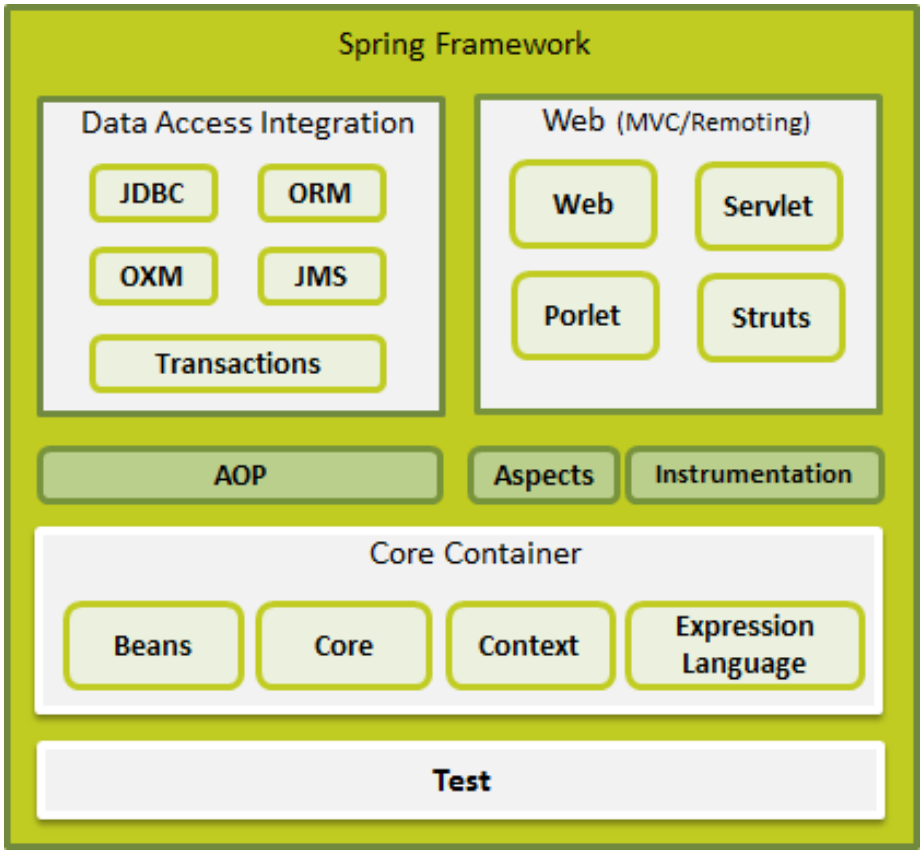
Framework - Ventajas

- Agiliza la codificación de aplicaciones.
- Es modular y estándar.
- Permite modificar o ampliar el software con mayor facilidad.
- Soporte constante por los desarrolladores y la comunidad.

Spring

- Spring es un framework de código abierto que nos permite un fácil desarrollo de aplicaciones Java.
- Un framework define una estructura con soluciones a diversos problemas.
- Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control.

Módulos



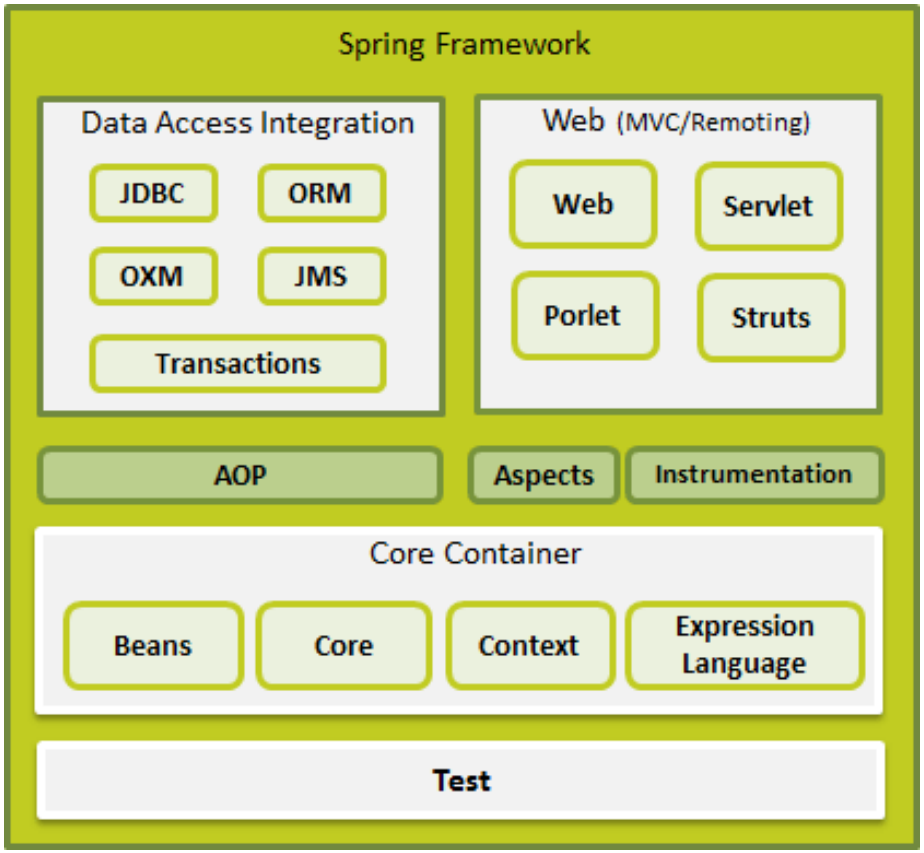
Core container

- Los módulos Beans y Context proveen la parte fundamental del framework.
- Se incluye el manejo de inversión de control (IoC) y la inyección de dependencias.

AOP

- Permite la implementación de la programación orientada a aspectos.

Módulos



Data access integration

- Contienes los módulos para abstraer JDBC y la integración con ORM incluyendo JPA.

Web

- Contiene los módulos para el desarrollo de aplicaciones Web, incluyendo el patrón MVC.

Ventajas

- Plantillas predefinidas
  Proporciona plantillas para JDBC, Hibernate, JPA y otras tecnologías.
- Es un framework bien diseñado para aplicaciones Web usando MVC.
- Fácil de probar
  La inyección de dependencias facilita las pruebas de la aplicación.
- Ligero
  No es necesario heredar clases o implementar interfaces.
- Abstracciones
  Abstracciones de JDBC, JPA y JTA.
- Diversas declaraciones
  Como caché, validaciones, transacciones y formatos.

SpringBoot

- SpringBoot es una parte del proyecto de Spring que permite levantar aplicaciones web sin necesidad de tener corriendo un contenedor de servlet como Tomcat o un servidor de aplicaciones aparte.
- Esto es posible ya que la dependencia de SpringBoot nos provee de un tomcat embebido por lo que el despliegue se torna mucho más agil a partir de ahora

Spring MVC

- Esta basado en peticions
- Proporciona una arquitectura MVC.
- Separa los diferentes aspectos de una aplicación:
  - **Model**

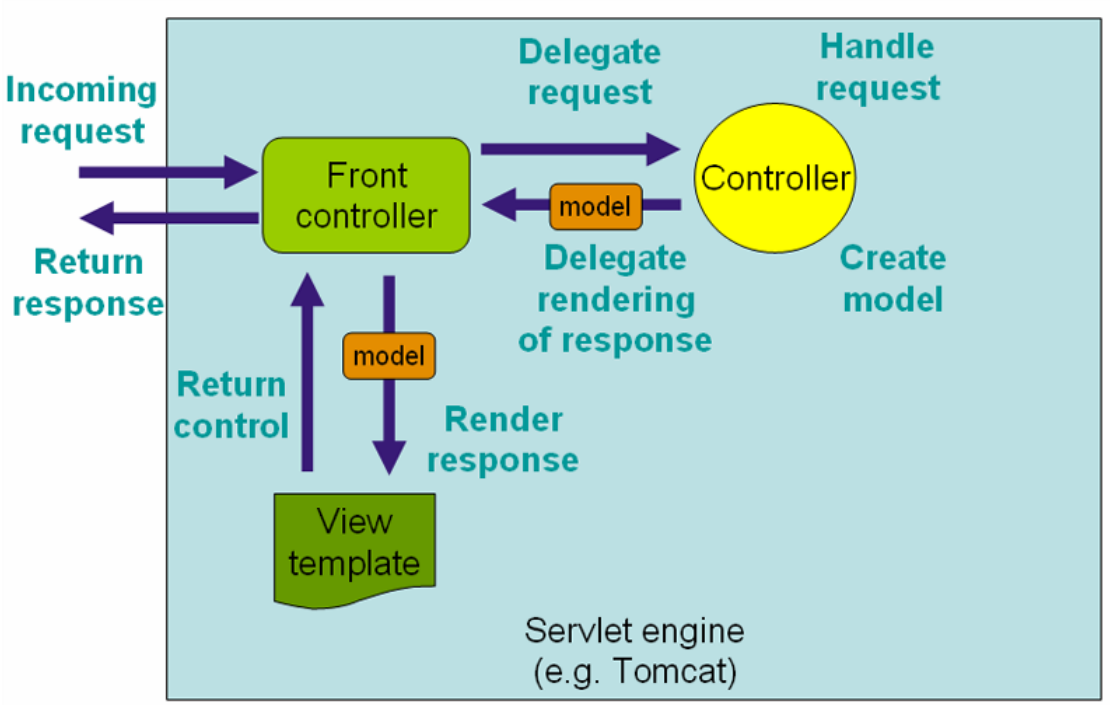
    Encapsula los datos de la aplicación, se representan por POJO's.
  - **View**

    Es responsable de presentar los datos generalmente en HTML.
  - **Controller**

    Es responsable de procesar las peticiones y trabajar con los modelos que luego serán presentados en la vista.
- Esta diseñado alrededor de DispatcherServlet.

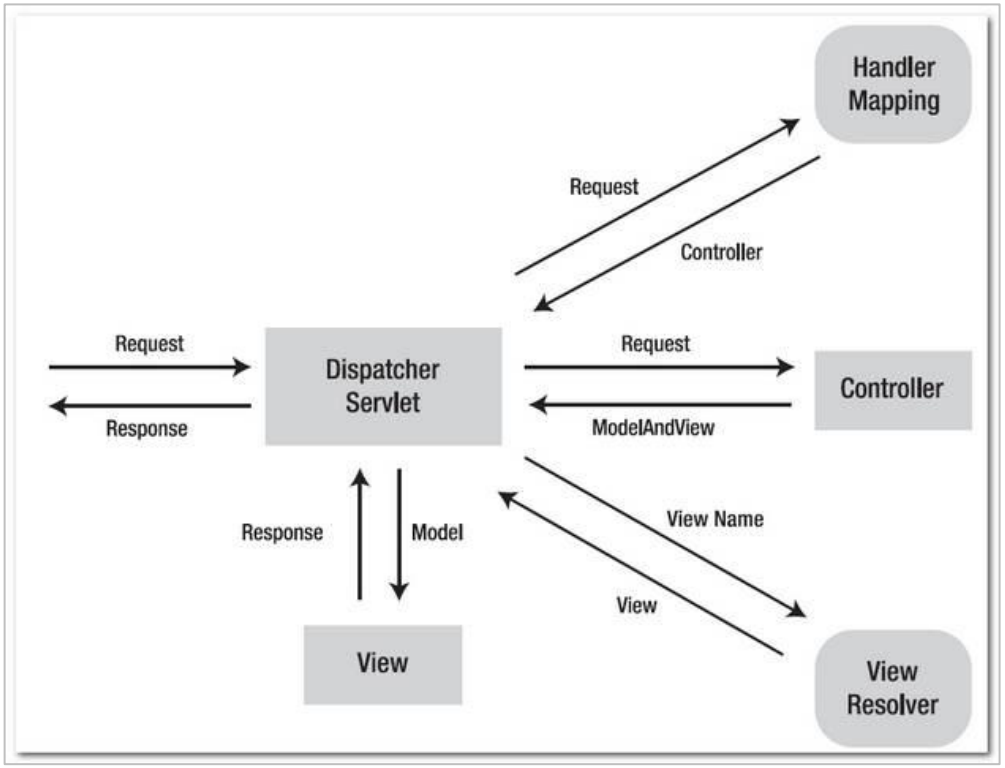
DispatcherServlet

- Envía las solicitudes a los controladores.
- Es una expresión del patrón de diseño Front Controller.



DispatcherServlet

- Permite manejar todas las peticiones y respuestas HTTP.



Controller

- Los controladores son invocados por DispatcherServlet para ejecutar la funcionalidad para la que han sido desarrollados.
- Las clases que van a cumplir el rol de controladores deben de tener la anotación:
  - **@Controller**
- Para enlazar las URL´s a las clases y sus métodos se utiliza la anotación:
  - **@RequestMapping (@GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping)**

View

- Sirven de presentación para las aplicaciones.
- Spring soporta varios tipos de presentación:
  - JSP
  - HTML
  - PDF
  - Excel
  - XML
  - XSLT
  - Json
  - JasperReports, etc.

```
model.addAttribute("serverTime", formattedDate );

return "home";
```

```
<body>
<h1>Hello world!</h1>
<P>The time on the server is ${serverTime}.</P>
</body>
```

Controller

```
@RequestMapping(value="/clientes/listado1", method=RequestMethod.GET)
public String listado1(Model model){

    ClienteManager cManager = new ClienteManager();
    model.addAttribute("clientes", cManager.listado());
    model.addAttribute("pagina", 1);

    return "cliente_listado";
}
```

```
public ModelAndView listado2(ModelAndView mav){

    ClienteManager cManager = new ClienteManager();
    mav.addObject("clientes", cManager.listado());
    mav.addObject("pagina", 2);

    mav.setViewName("cliente_listado");

    return mav;
}
```

```
public ModelAndView listado3(){

    ModelAndView mav = new ModelAndView("cliente_listado" );

    ClienteManager cManager = new ClienteManager();
    mav.addObject("clientes", cManager.listado());
    mav.addObject("pagina", 3);

    return mav;
}
```

Controller

- El atributo **value** define la URL que va a ser administrada por el controlador.
- El atributo **method** define el método HTTP que va ser manejado.

```
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
```

Thymeleaf

Concepto

- Es un motor de plantilla de Java que se ejecuta del lado del servidor.
- Puede procesar HTML, XML, JavaScript, CSS e incluso texto sin formato.

```
<html xmlns:th="http://www.thymeleaf.org">
```

```
<input type="text" name="userName" value="James Carrot" th:value="${user.name}" />
```

Enlaces

- texto
- texto
- texto

Iteración

```
<tr th:each="prod : ${prods}">
  <td th:text="${prod.name}">Onions</td>
  <td th:text="${prod.price}">2.41</td>
  <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
</tr>
```

```
<tr th:each="prod,iterStat : ${prods}" th:class="${iterStat.odd}? 'odd'">
  <td th:text="${prod.name}">Onions</td>
  <td th:text="${prod.price}">2.41</td>
  <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
</tr>
```

Condicional

```
<tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">
  <td th:text="${prod.name}">Onions</td>
  <td th:text="${prod.price}">2.41</td>
  <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
  <td>
    <span th:text="${#lists.size(prod.comments)}">2</span> comment/s
    <a href="comments.html"
       th:href="@{/product/comments(prodId=${prod.id})}"
       th:if="${not #lists.isEmpty(prod.comments)}">view</a>
  </td>
</tr>
```

```
<a href="comments.html"
   th:href="@{/comments(prodId=${prod.id})}"
   th:unless="${#lists.isEmpty(prod.comments)}">view</a>
```

Switch

```
<div th:switch="${user.role}">
  <p th:case="'admin'">User is an administrator</p>
  <p th:case="#{roles.manager}">User is a manager</p>
  <p th:case="*">User is some other thing</p>
</div>
```

Formularios I

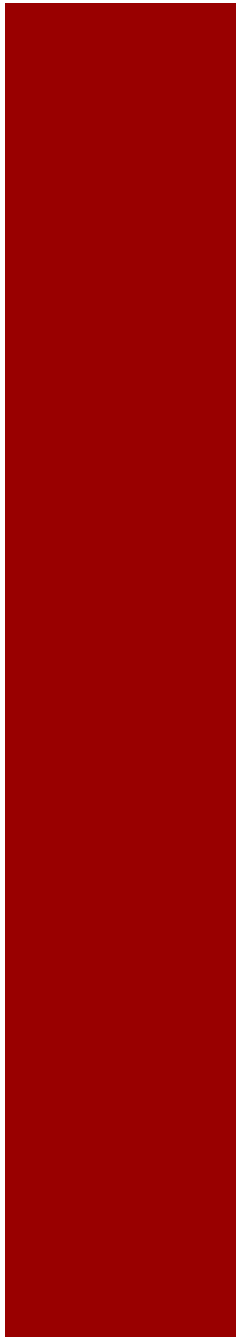
```java
@RequestMapping(value = "/alumnos/agregar", method=RequestMethod.GET)
public String agregar(Model model){
    Alumno objAlumno = new Alumno();

    model.addAttribute("alumno", objAlumno);
    return "alumno/agregar";
}
```

```html
<form action="#" th:action="@{/alumnos/guardar}" th:object="${alumno}" method="post">
    <table>
        <tr>
            <td><label th:text="Nombres"></label></td>
            <td><input type="text" th:field="*{nombres}" /></td>
        </tr>
        <tr>
            <td><label th:text="Apellidos"></label></td>
            <td><input type="text" th:field="*{apellidos}" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="Guardar" /></td>
        </tr>
    </table>
</form>
```

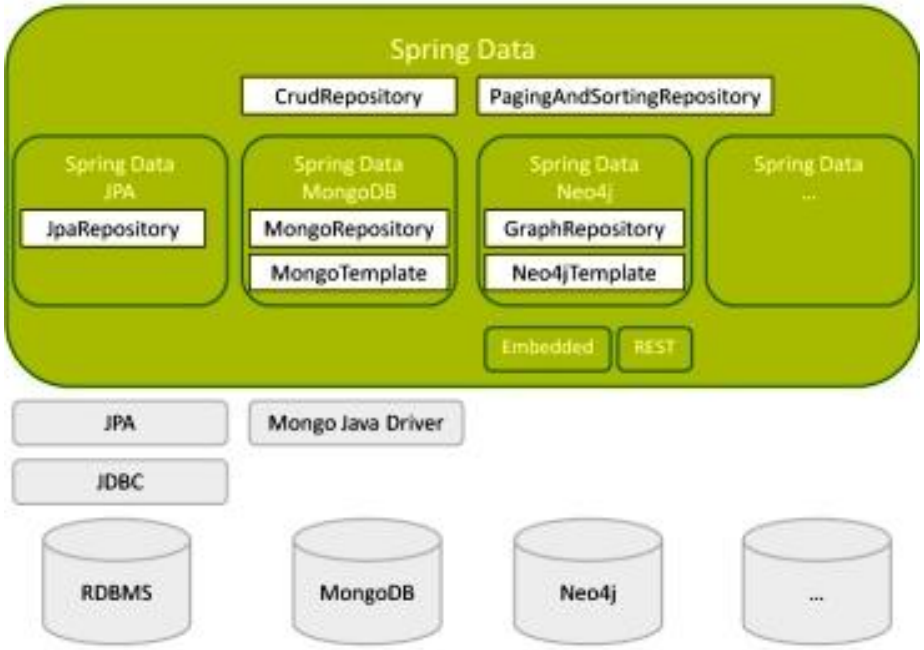
Formularios II

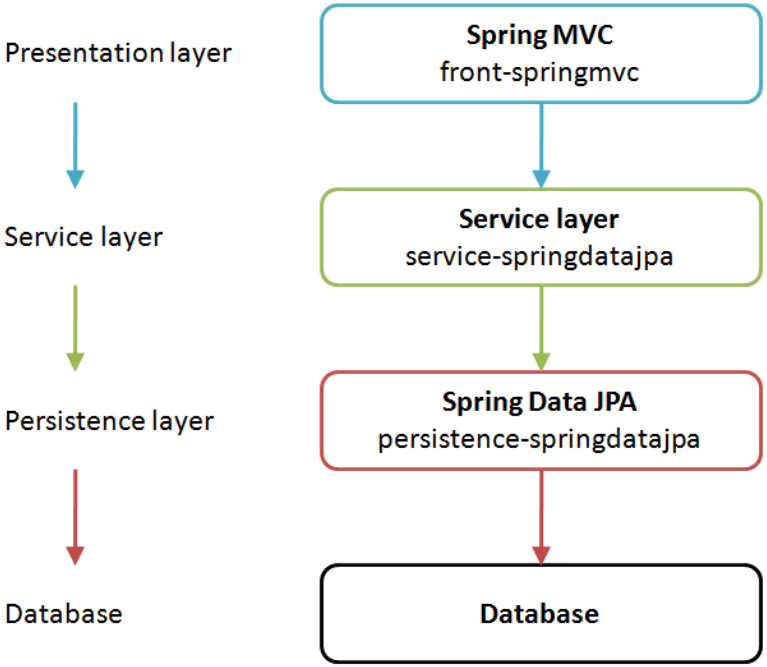
```
@RequestMapping(value = "/alumnos/guardar", method = RequestMethod.POST)
public String guardar(@ModelAttribute Alumno alumno) {

    //alumno.getNombres();
    //alumno.getApellidos();

    return "alumno/agregar";
}
```

Spring Data JPA

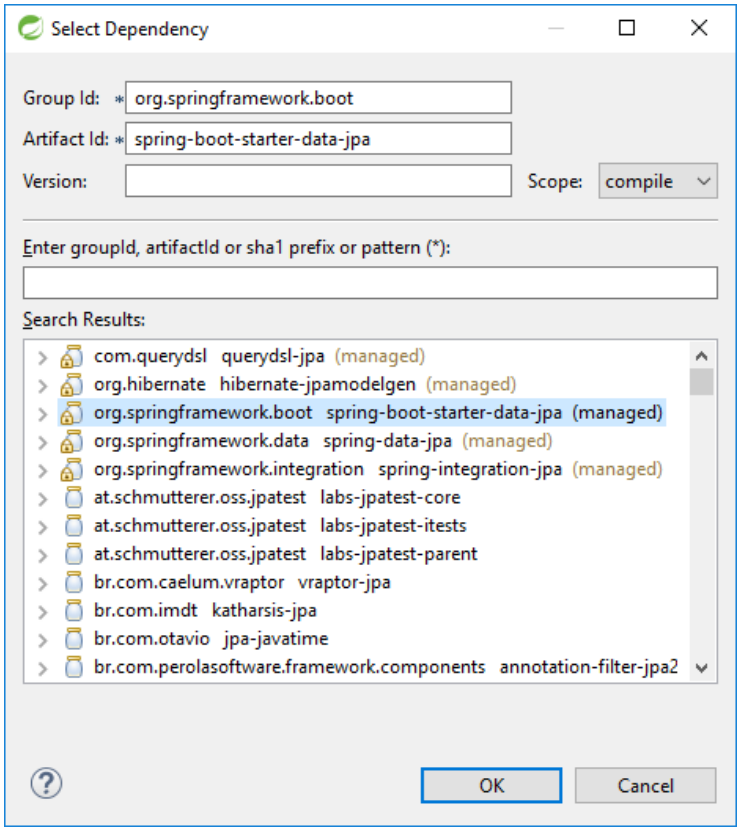


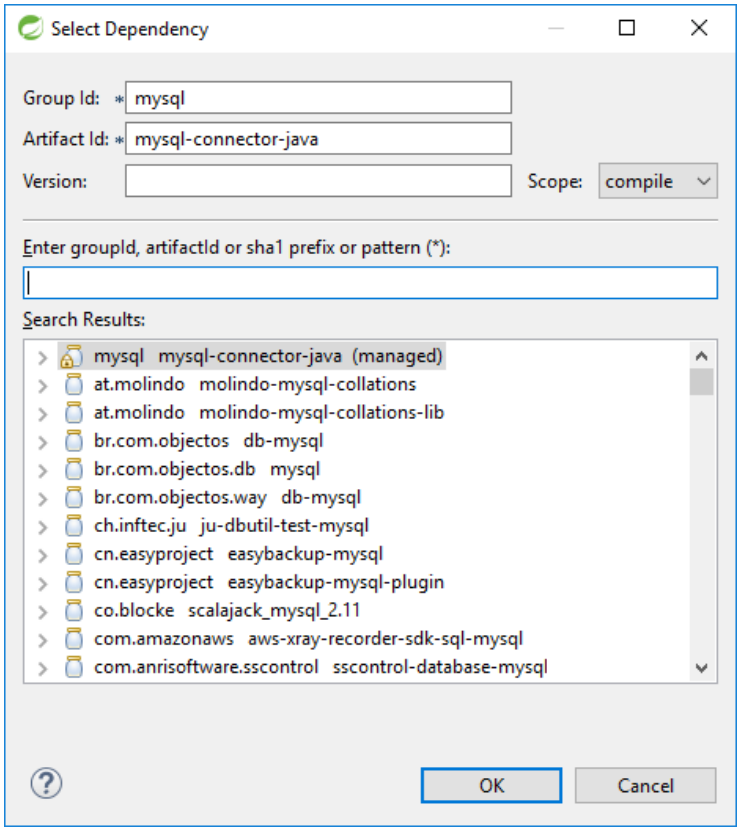
Spring


Spring Data
CrudRepository
PagingAndSortingRepository
Spring Data JPA
JpaRepository
Spring Data MongoDB
MongoRepository
MongoTemplate
Spring Data Neo4j
GraphRepository
Neo4jTemplate
Spring Data ...
Embedded
REST
JPA
Mongo Java Driver
JDBC
RDBMS
MongoDB
Neo4j
...


Presentation layer
Service layer
Persistence layer
Database
Spring MVC
front-springmvc
Service layer
service-springdatajpa
Spring Data JPA
persistence-springdatajpa
Database

Dependencias





Enlace de interés

- Spring MVC
  - http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html