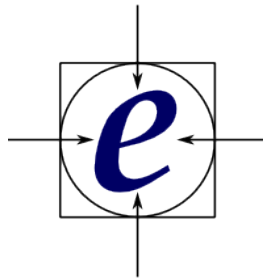
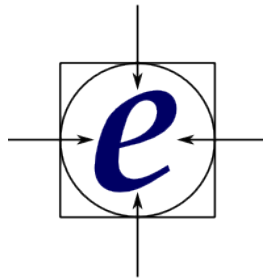


Spring MVC

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Cosa vediamo?



- iniezione di dipendenza *(ci piacciono i paroloni)*
- emme-vu-cì: ma che vol dì? *(ci piacciono i paroloni 2)*
- implementazione di un semplice controller
- gestire la logica di navigazione
- creare un vista
- impostare una form:
  - binding dei dati...
  - ...e validazione (e mostrare errori, se ce ne sono!)

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Spring e Dependency Injection

La **dependency injection** (DI) nella programmazione orientata agli oggetti è un modello di progettazione (*pattern*) il cui principio fondamentale si basa sulla separazione tra le implementazioni concrete degli oggetti e la risoluzione delle dipendenze.
La dependency injection è una forma specifica di "**inversione di controllo**" in cui ciò che viene invertito è il processo per ottenere la necessaria dipendenza (espressa solo in termini di interfacce).
Il termine è stato coniato da Martin Fowler.

In altre parole: *è una tecnica per disaccoppiare i componenti software tra loro dipendenti*.

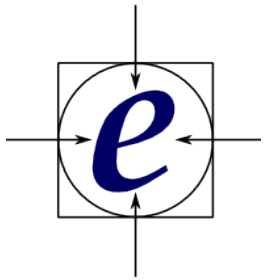
La dependency injection, invece di codificare le dipendenze direttamente nel programma, affida ad una terza parte il compito di istanziare i servizi necessari (in un contenitore che rappresenta il *contesto dell'applicazione*) e soddisfare le dipendenze.
Tutto ciò, in Spring, avviene attraverso un *file di configurazione:*

```
<bean id="bookDao" class="biz.elabor.library.dao.JdbcBookDao">
        <property name="dataSource" ref="dataSource" />
</bean>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${jdbc.driverClassName}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
</bean>
```

Questa è una implementazione concreta ma chi usa il Dao lo fa attraverso l'interfaccia “BookDao”

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Modello-Vista-Controllore



Il Paradigma MVC separa nettamente la logica di **business** di un'applicazione, da quella di **navigazione** e di **presentazione**.

Il Controller

1. gestisce la logica di navigazione ed interagisce con lo strato dei servizi per comandare la logica di business.

Il Modello

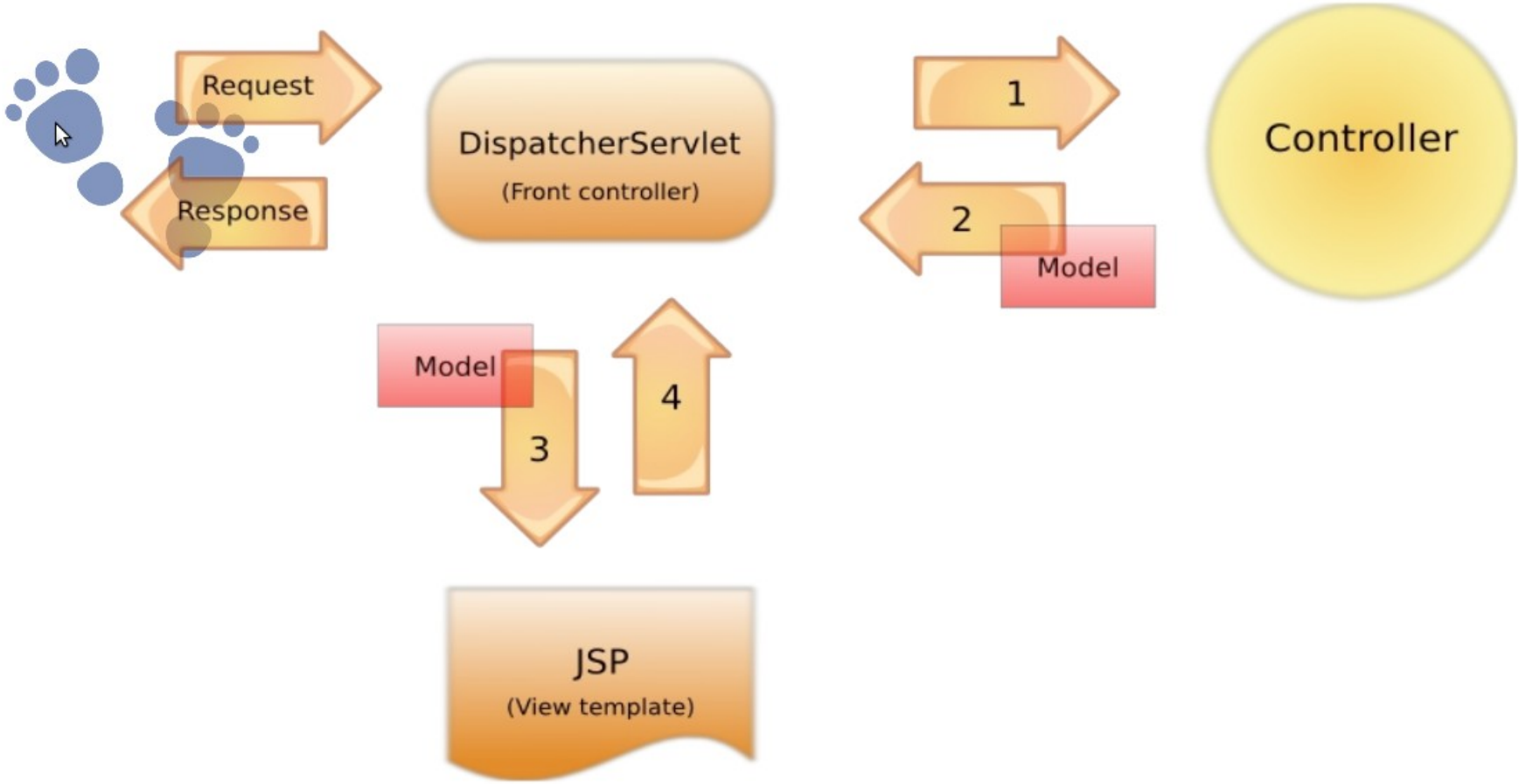
1. rappresenta il contatto tra il Controllore e la Vista;
2. contiene i dati da visualizzare nella Vista;
3. è popolato dal Controllore.

La Vista

4. estrae i dati dal modello;
5. è il contenuto da mostrare all'utente.

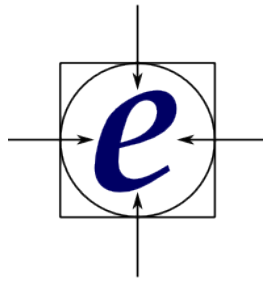
Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

MVC in azione



Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

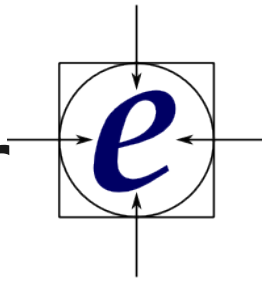
diciamolo al web.xml



```
<!-- Configures the @Controller programming model -->
<servlet>
    <description>Spring MVC Dispatcher Servlet</description>
    <servlet-name>library</servlet-name>
    <servlet-class>org.springframework.web.
                servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>library</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Ecco a voi il signor Controller



```
@Controller
@RequestMapping({"/my/hello.html"})
public class MyController {

   @RequestMapping
   public void index() {
      // do something useful
   }
}
```

I'm just a POJO!

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Configurazione del Contesto library-servlet.xml

```
<!-- Configures the @Controller programming model -->
<mvc:annotation-driven />

<!-- Forwards requests to the "/" resource to the "welcome" view -->
<mvc:view-controller path="/" view-name="home"/>

<!-- Scan for controllers and services -->
<context:component-scan
       base-package="biz.elabor.library.web.controllers” />

<!-- Views -->
<bean id="viewResolver"
       class="org.springframework....InternalResourceViewResolver">
    <property name="viewClass"
       value="org.springframework.web.servlet.view.JstlView">
    </property>
    <property name="prefix" value="/jsp/"></property>
    <property name="suffix" value=".jsp"></property>
</bean>
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Uso del Modello

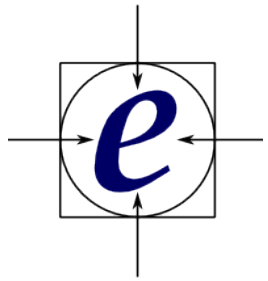
```
@Controller
public class HomePageController {

    @Autowired
    private BookService bookService;
    public void setBookService(BookService bookService) {
        this.bookService = bookService;
    }

    @RequestMapping("/home")
    public String homeHandler(ModelMap model) {
        model.addAttribute("books", this.bookService.getAllBooks());
        return "home";
    }

…
}
```

Questo è il nome della vista da usare.
Anche il parametro di ritorno è “variabile”:

- Void (risoluzione di default)
- Stringa (path alla vista)
- Oggetto Vista (non necessariamente jsp)

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

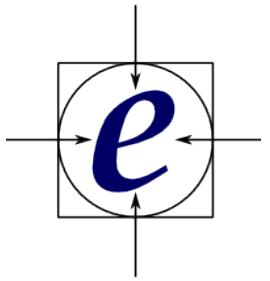
...e la home.jsp



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

<c:forEach items="${books}" var="book">
    <tr>
        <td>
            <c:url var="viewBook" value="/book/view/${book.isbn}"></c:url>
            <a href="${viewBook}">
                <c:out value="${book.isbn}"></c:out>
            </a>
        </td>
        <td>
            <c:url var="authorBooks"
                value="/author/books/${book.author.id}"></c:url>
            <a href="${authorBooks}">
                <c:out value="${book.author.name}"></c:out>
            </a>
        </td>
        <td><c:out value="${book.title}"></c:out></td>
    </tr>
</c:forEach>
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Uso del Modello (reprise)



```
@Controller
public class HomePageController {

    …

    @ModelAttribute("books")
    public List<Book> getBooks() {
        return this.bookService.getAllBooks();
    }

}
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Recepire parametri... (1/2)

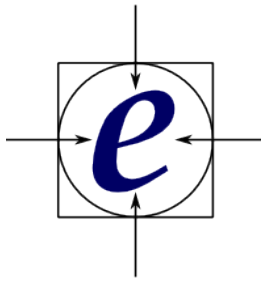
http://www.mylibrary.it/book/view/1234

```
@Controller
public class ViewBookController {

    @Autowired
    private BookService bookService;
    public void setBookService(BookService bookService) {
        this.bookService = bookService;
    }

    @RequestMapping(value="/book/view/{isbn}", method=RequestMethod.GET)
    public String get(@PathVariable String isbn, ModelMap model) {
        Book book = this.bookService.getBook(isbn);
        model.addAttribute(book);
        return "book/view";
    }

    …

}
```

firma variabile!

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

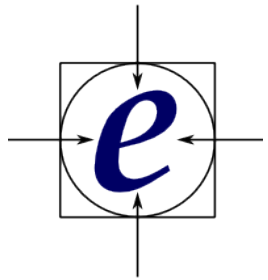
Recepire parametri... (2/2)



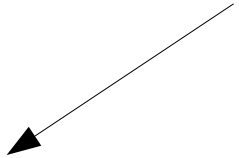
```
@Controller
public class ViewBookController {

    @Autowired
    private BookService bookService;
    public void setBookService(BookService bookService) {
        this.bookService = bookService;
    }

    @RequestMapping(value="/book/view",
                    method=RequestMethod.POST, params="delete")
        public String delete(@RequestParam String isbn, ModelMap model) {
            this.bookService.delete(isbn);
            return "redirect:/home";
        }

}
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Prepariamo una form



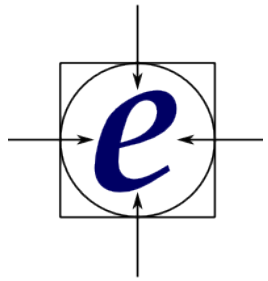
http://www.mylibrary.it/book/edit?isbn=1234



```
@Controller
public class EditBookController {

    @RequestMapping(value="/book/edit", method=RequestMethod.GET)
    public String get(@RequestParam(required=false) String isbn,
                      ModelMap model) {
        Book book;
        if (isbn != null) {
            book = this.bookService.getBook(isbn);
        } else {
            book = this.bookService.createBook();
        }
        model.addAttribute(book);
        return "book/edit";

    …

}
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

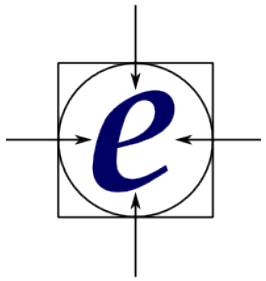
/book/edit.jsp



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

<form:form modelAttribute="book" method="POST">

    <dl>
        <dt>Isbn</dt>
        <dd>
            <form:input path="isbn" />
            <form:errors path="isbn" cssClass="error" />
        </dd>
        <dt>Titolo</dt>
        <dd>
            <form:input path="title" />
            <form:errors path="title" cssClass="error" />
        </dd>
        ...
    </dl>

    <div class="actions">
        <input type="submit" name="add" value="Aggiungi" />
    </div>

</form:form>
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

e ora processiamo la form!



```
@Controller
public class EditBookController {

    …

    @RequestMapping(value="/book/edit", method=RequestMethod.POST)
    public String post(@Valid Book book, BindingResult result) {
        if (result.hasErrors()) {
            return "/book/edit";
        }
        this.bookService.saveBook(book);
        return "redirect:/home";
    }
}
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Come solo @Valid?!?

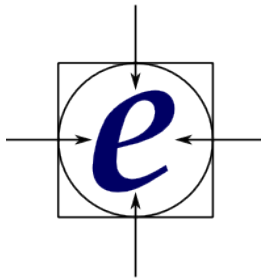
```
public class Book implements Serializable {

    @NotBlank
    String isbn;

    @NotBlank
    @Size(max=50)
    String title;

    @NotNull
    @Past
    @DateTimeFormat(style="S-")
    Date finished;

    public Book() {
        this.finished = new Date();
    }

    ...
}
```

org.hibernate.validator.constraints

javax.validation.constraint

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

ma anche @NotNull non basta...

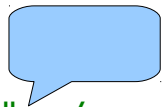


```
<!-- Configures the @Controller programming model -->
<mvc:annotation-driven validator="validator" />

<bean id="validator"
        class="org.springframework.validation.
                .beanvalidation.LocalValidatorFactoryBean" />
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

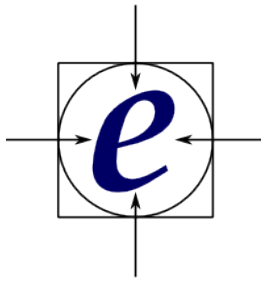
E per gli attributi complessi?

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

<form:form modelAttribute="book" method="POST">

    <dl>
        ...
        <dt>Author</dt>
        <dd>
            <form:select path="author" items="${authors}"
                         itemLabel="name" itemValue="id" />
            <form:errors path="author" cssClass="error" />
        </dd>
        <dt>Finito</dt>
        <dd>
            <form:input path="finished" />
            <form:errors path="finished" cssClass="error" />
        </dd>
    </dl>

</form:form>
```



Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

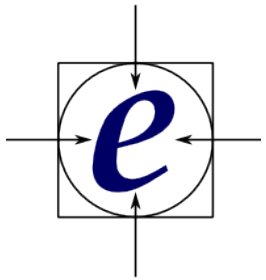
a loro ci pensa il ConversionService...



```
<!-- Configures the @Controller programming model -->
<mvc:annotation-driven conversion-service="conversionService" />

<bean id="conversionService"
    class="org.springframework.format.support
                .FormattingConversionServiceFactoryBean">
    <property name="converters">
        <list>
            <bean class="biz.elabor.library.web.
                            converters.AuthorConverter" />
        </list>
    </property>
</bean>
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

...a cui abbiamo dato un aiutino



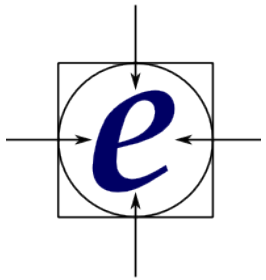
```
public class AuthorConverter implements Converter<String, Author> {

    @Autowired
    private BookDao bookDao;
    public void setBookDao(BookDao bookDao) {
        this.bookDao = bookDao;
    }

    @Override
    public Author convert(String authorId) {
        Integer id = Integer.valueOf(authorId);
        return this.bookDao.findAuthorById(id);
    }

}
```


Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Domande?

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz

Riferimenti



- Spring documentation:
  - http://www.springsource.org/documentation
- Spring reference:
  - http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html
- Spring in Action:
  - http://www.manning.com/walls4/
- Eclipse J2EE e SpringIDE:
  - Eclipse J2EE - http://www.eclipse.org/downloads/moreinfo/jee.php
  - SprindIDE - http://dist.springframework.org/release/IDE
  - STS - http://www.springsource.com/developer/sts

Ivan Ricotti - eLabor sc
i.ricotti@elabor.biz
http://www.elabor.biz