

Jakarta EE

Leonardo De Boni

December 6, 2022

## CONTENTS

1	Introduzione	2
1.1	Ottenere un indirizzo ip	2
1.2	Connettersi al server	2
2	Servlet	2
2.1	Servlet annotations	2
3	Struttura di un progetto web	3
4	Resources	3
5	Filters	3
5.1	Filter annotations	3
6	JSP (Jakarta Server Pages)	3
6.1	Expression Language	4
6.2	Direttive	4
6.3	Tag Library	4
7	Pattern MVC (Model View Controller)	4
7.1	Struttura MVC	4

## 1 INTRODUZIONE

Il package che ci interessa é: `java.net`. Le classi che ci interessano sono:

- `InetAddress`: rappresenta un indirizzo ip
- `Socket`: permette di creare un client
- `ServerSocket`: permette di creare un Server

### 1.1 Ottenere un indirizzo ip

Per ottenere l'indirizzo ip a partire da un nome possiamo fare così:

```
InetAddress address = InetAddress.getByName(String hostName);
```

### 1.2 Connettersi al server

Per connettersi a un server dopo aver ottenuto l'indirizzo possiamo creare un client così:

```
Socket socket = new Socket(InetAddress address, int port)
```

Dal socket possiamo ottenere un `InputStream`:

```
InputStream in = socket.getInputStream();
```

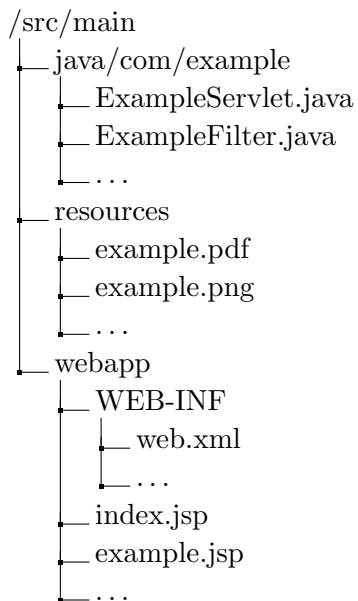
## 2 SERVLET

Classe che mi permette di elaborare una richiesta. Per crearne uno devo estendere la classe `HttpServlet`. I metodi che posso sovrascrivere che corrispondono alle varie richieste sono i seguenti:

- `doGet` (`HttpServletRequest req`, `HttpServletResponse res`)
- `doPost` (`HttpServletRequest req`, `HttpServletResponse res`)
- `doHead` (`HttpServletRequest req`, `HttpServletResponse res`)
- `doOptions` (`HttpServletRequest req`, `HttpServletResponse res`)
- `doTrace` (`HttpServletRequest req`, `HttpServletResponse res`)
- `doPut` (`HttpServletRequest req`, `HttpServletResponse res`)
- `doDelete` (`HttpServletRequest req`, `HttpServletResponse res`)

### 2.1 Servlet annotations

### 3 STRUTTURA DI UN PROGETTO WEB



### 4 RESOURCES

### 5 FILTERS

#### 5.1 Filter annotations

### 6 JSP (JAKARTA SERVER PAGES)

JSP mi permette di scrivere codice java all'interno di HTML. Per fare ciò posso usare delimitatori dedicati:

- **Scriptlets:** `<% ... istruzioni java ... %>`
- **Expressions:** `<%= ... espressioni java ... %>` sarà mostrato al client
- **Declarations:** `<%! ... espressioni java ... %>` posso dichiarare metodi

Nelle pagine JSP ho a disposizione vari oggetti impliciti:

- `request` `HttpServletRequest`
- `response` `HttpServletResponse`
- `out` `OutputStream` (`response.getOutputStream()`)
- `page` `this`
- `exception` `Throwable`
- `config` `ServletConfig`

6.1 Expression Language

6.2 Directive

6.3 Tag Library

7 PATTERN MVC (MODEL VIEW CONTROLLER)

7.1 Struttura MVC