Web application using JSF, JSP and Java servlets

M.I. Capel

ETS Ingenierías Informática y
Telecomunicación
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
Email: manuelcapel@ugr.es
http://lsi.ugr.es/mcapel/

October, 9th 2017
Máster Universitario en Ingeniería Informática

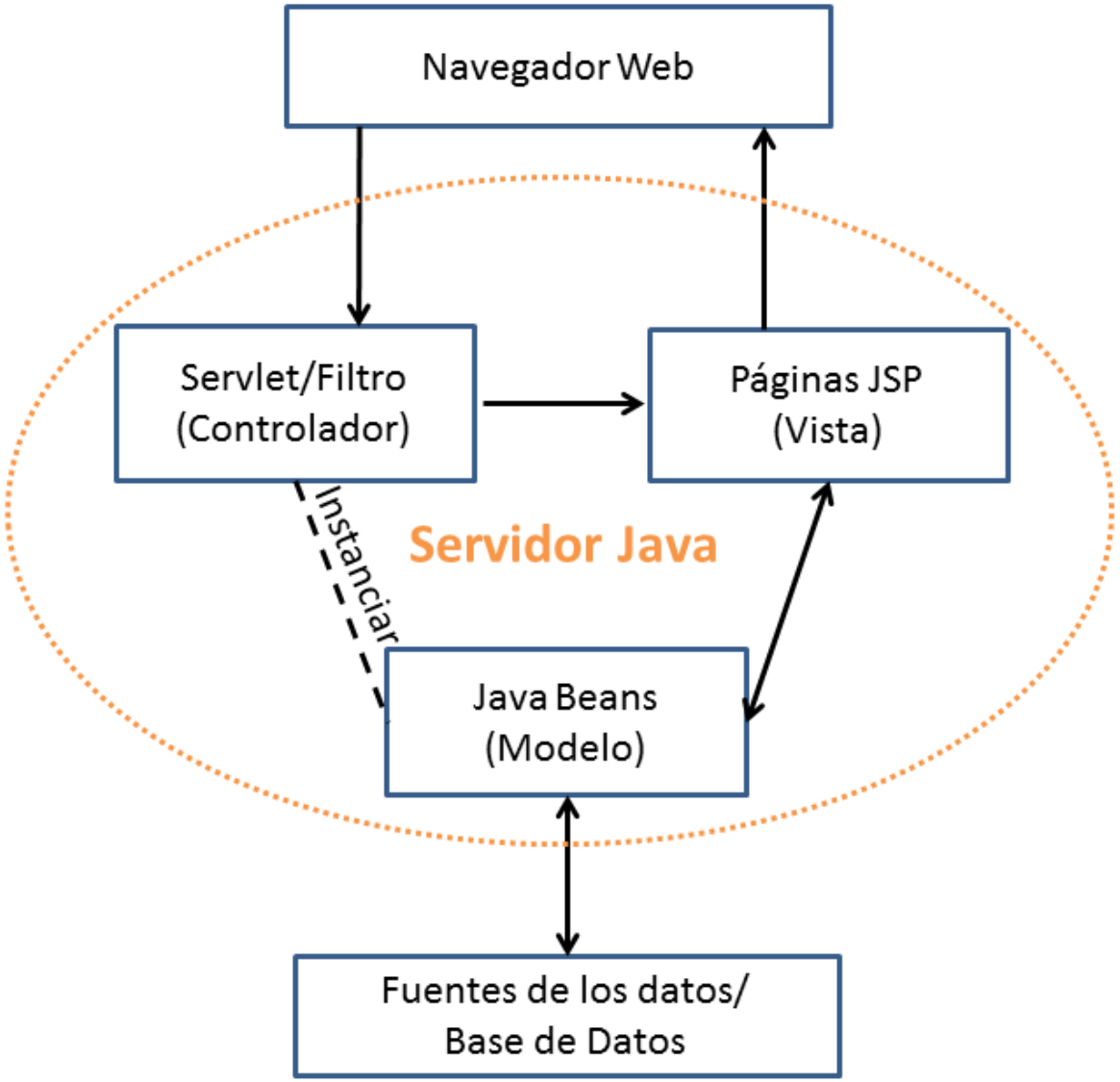
Java Servlet Faces (JSF)

Definition

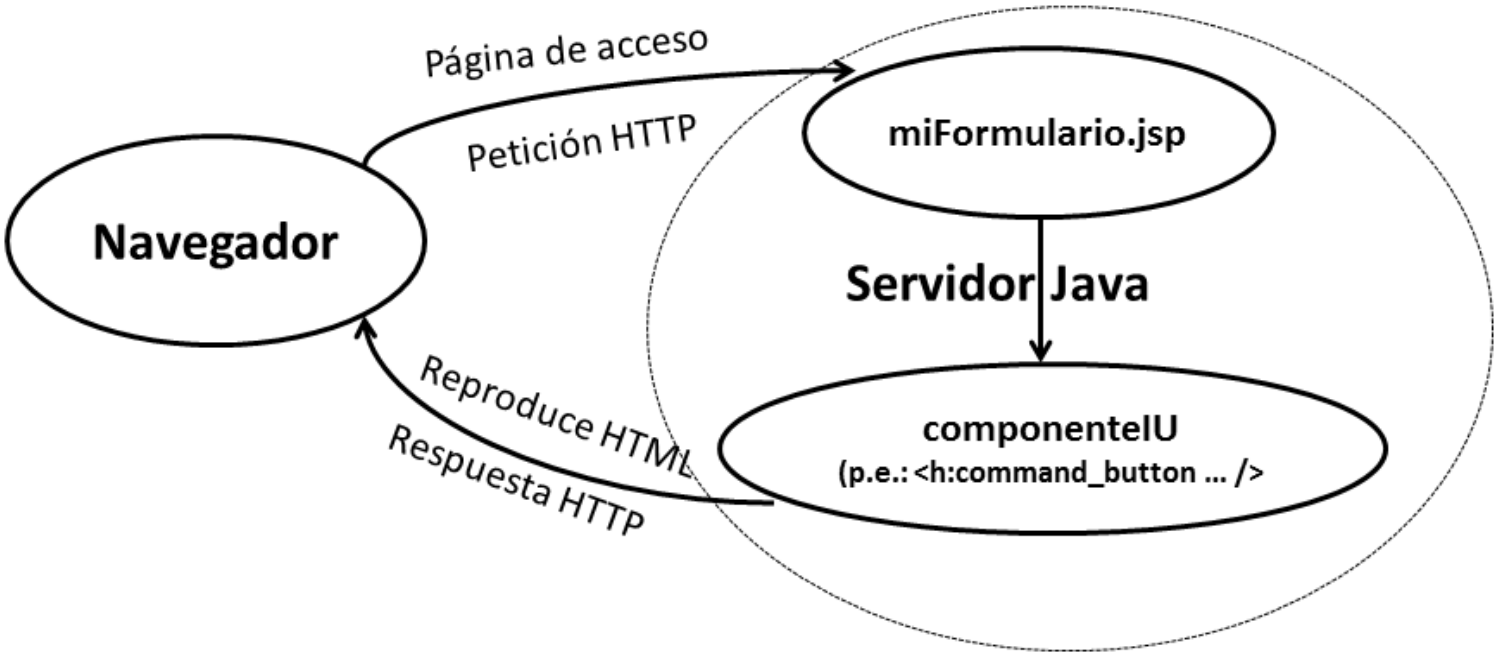
The system JSF is a MVC design pattern-based framework, which simplifies building User Interfaces (UI) for server applications with UI made up of components, inside a JSP page

With JSP technology, the event handler cannot accept `http` demands to a server's component directly nor can manage UI components as server objects

MVC of a Web application that only uses JSP



Software architecture of an application that uses JSF technology



The framework JSF II

Benefits

- JSF eases conections between UI *"widgets"* with data sources and event handlers on the server side
- It provides a standard API for software components development
- It allows reuse and extension of current UI-components standars
- It eases data transfer between UI components
- It manages the state of a UI-components that passes through several servers
- It helps to connect events (client–part) with the application code that must handle these events (server–part)

Advantages of using JSF technology

- With JSF technology we can connect events generated at the client tier with the Web application code at the server side
- To map *UI components* to server's-side objects

Advantages of using JSF technology-II

A JSP page can manage all the objects that take part in the Web application in which we are interested to:

- component's objects: label type `< h : command_button... >`,
- action listeners of actions that start from *validator* and *conversor* objects, which are normally included in component labels, such as: `<f:validate _longrange minimum="0" maximum="10"/>` ,
- Model's objects that contain the Web application data,
- the remainder of functionality of the application components

JSF software architecture

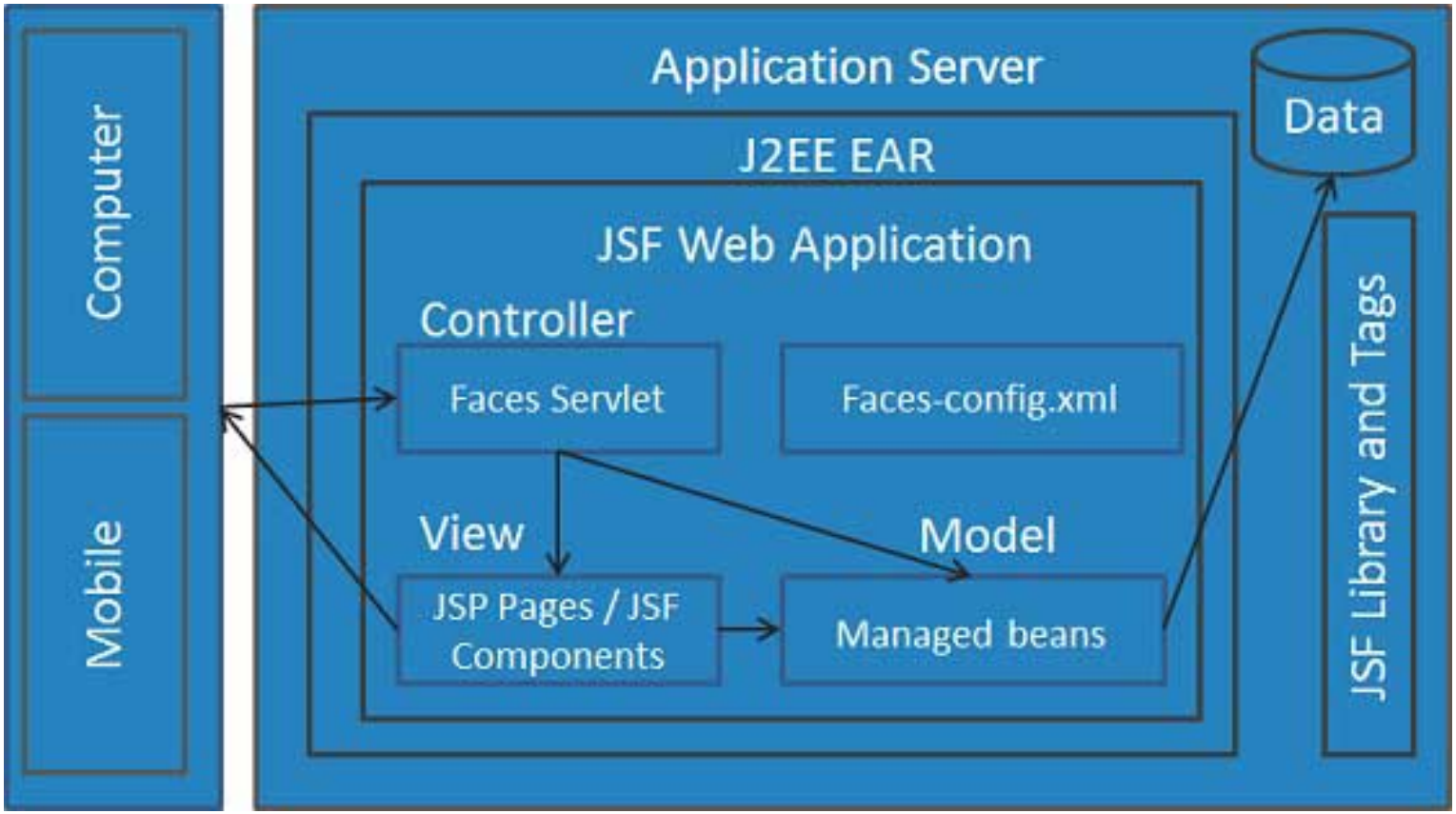


Figure: Architecture of JSF

“Controller”

Faces Servlet

- It is in charge of capturing all the client tier demands
- It initiates all the JSF framework components before the JSF page shows up
- Fundamental services to build the controller are imported from the library `javax.faces.webapp.FacesServlet`
- The controller must be programmed in the file`web.xml`: files ended by `.jsf` and `.xhtml` in the current scope are sought for building the views

Java Managed Beans

Fundamental concept:

A *bean* Java class but managed within JSF.

managed bean: elements

- *beans* are serializables,
- have `getter()` and `setter()` methods,
- have the business logic : all the information associated to a *bean* that is in HTML forms of the aplicación

“Model”

- It is a package included in `src/main/java`, which contains the classes: `.java` as *Managed Beans* (MB)
- Java “beans” can be managed by JSF by using a configuration file in `XML` or through **anotations**:
  - `@ManagedBean(name= , eager= )`
  - `@(Request|None|View|Session|Application|Custom)Scoped`
  - `@ManagedProperty`

"Model"–II

```
1  package prueba ;
2  import javax.faces.bean.ManagedBean ;
3  import javax.faces.bean.ManagedProperty ;
4  import javax.faces.bean.RequestScoped ;
5
6  @ManagedBean (name = "holaMundo", eager = true)
7  @RequestScoped
8  public class HolaMundo{
9  @ManagedProperty ( value ="#{mensaje}" )
10 //completar
11 public HolaMundo ( ){
12 System.out.println( "HolaMundo ha comenzado ! " ) ;
13 }
14 public String getMensaje( ){
15 //completar
16 }
17 public void setBeanMensaje ( Mensaje m ) {
18 //completar
19 }
```

"Model"–III

```
1  package holamundoMIC;
2  import java.io.Serializable;
3  import javax.faces.bean.ManagedBean;
4  import javax.faces.bean.ManagedProperty;
5  import javax.faces.bean.RequestScoped;
6  @ManagedBean(name="holaMUndo2",eager=true)
7  @RequestScoped
8  public class HolaMundo2 implements Serializable{
9  @ManagedProperty(value="#{mensaje}")
10 private MensajeBean mensajeBean;
11 private String mensaje= "Nada aun!";
12 public HolaMundo2(){
13 System.out.println("Hola mundo ha comenzado!");
14 System.out.println(mensaje);}
15 public String getMensaje(){
16 if(mensajeBean != null){
17 mensaje= mensajeBean.getMensaje();}
18 return mensaje;
19 }
20 public void setMensajeBean(MensajeBean m){
21 this.mensajeBean=m;}
22 }
```

"Model"–IV

```
1  package holamundoMIC;
2  import java.io.Serializable;
3  import javax.faces.bean.ManagedBean;
4  import javax.faces.bean.ManagedProperty;
5  import javax.faces.bean.RequestScoped;
6
7  @ManagedBean(name="mensaje",eager=true)
8  @RequestScoped
9  public class MensajeBean implements Serializable {
10 private String mensaje="Hola todo el mundo y parte del extrajero
       !";
11 public String getMensaje(){
12 return mensaje;
13 }
14 public void setMensaje(String mensaje){
15 this.mensaje=mensaje;
16 }
17 }
```

"View"

JSF page creation

It takes place by creating a `home.xhtml` page (in `src/main/webapp/`) that can be browsed by an XML compatible explorer

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head>
5      <title>JSF Tutorial!</title>
6  </head>
7  <body>
8      #{holaMundo.mensaje}
9  </body>
10 </html>
```

IDEs

Definition

Visual application for building software applications from components

Elements of an IDE

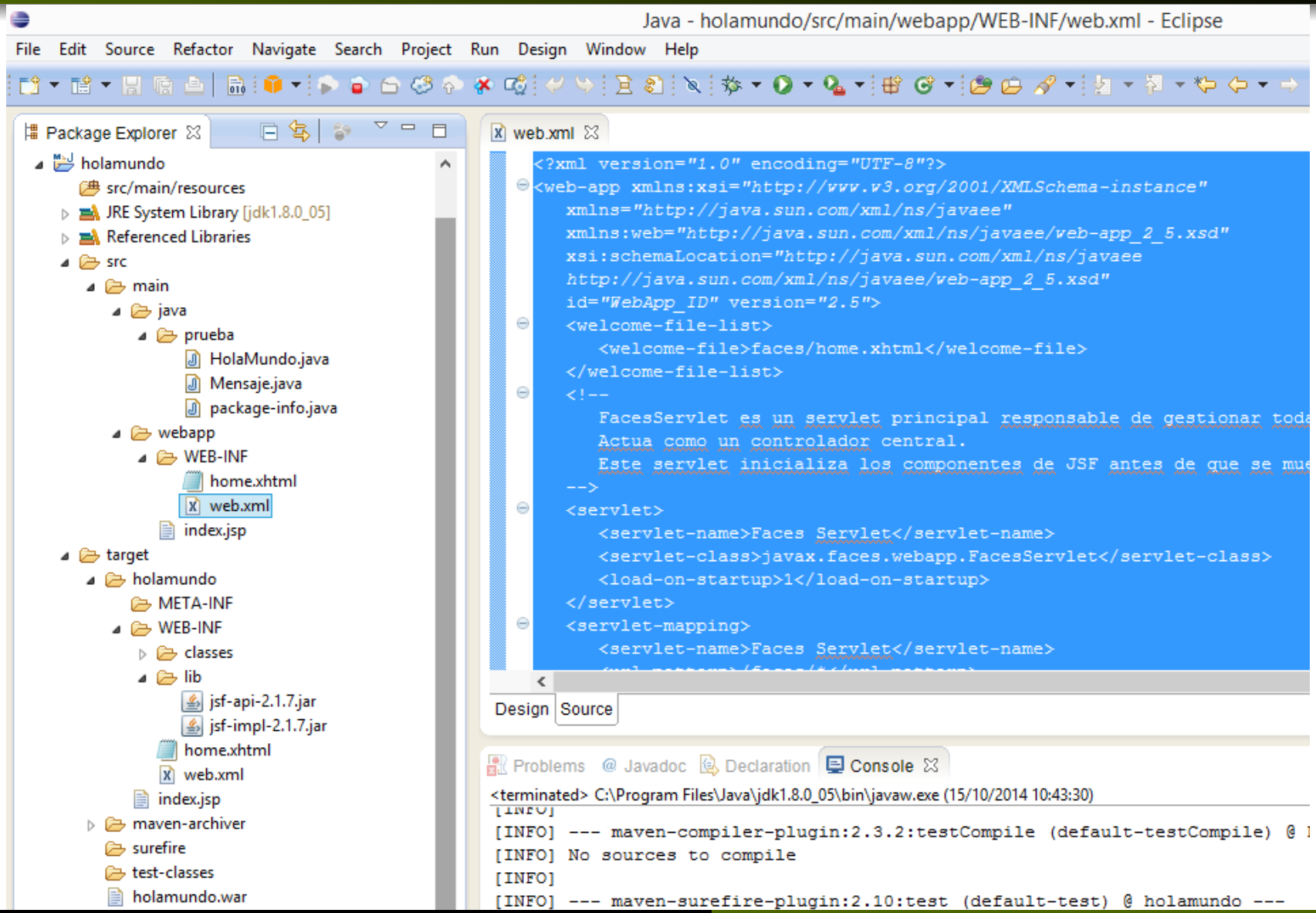
- Canvas or container
- Editors for configuring and specializing components
- Viewers and browsers
- Directories of components
- Tools for component-based development (compilers, debuggers, unit proofs, etc.)
- Project management and control access
- Support for CSCW

Current IDEs examples

Name	Builder
Visual Studio	Microsoft
Visual Age	IBM
Visual Cafe	Symantec
Eclipse	ESF
NetBeans	Oracle

Complemented with configuration languages: JavaScript, VBScript, etc.

Eclipse project-files browser structure



References

- JSP Tutorial: http://www.jsptut.com/
- EJB Tutorial: http://www.tutorialspoint.com/ejb/
- Java Server Faces in Wikipedia : http://en.wikipedia.org/wiki/JavaServer_Faces
- Java WAR files : http://www.yolinux.com/TUTORIALS/Java-WAR-files.html
- Maven Repository : http://mvnrepository.com/artifact/org.apache.commons
- Maven POM Reference : http://maven.apache.org/pom.html