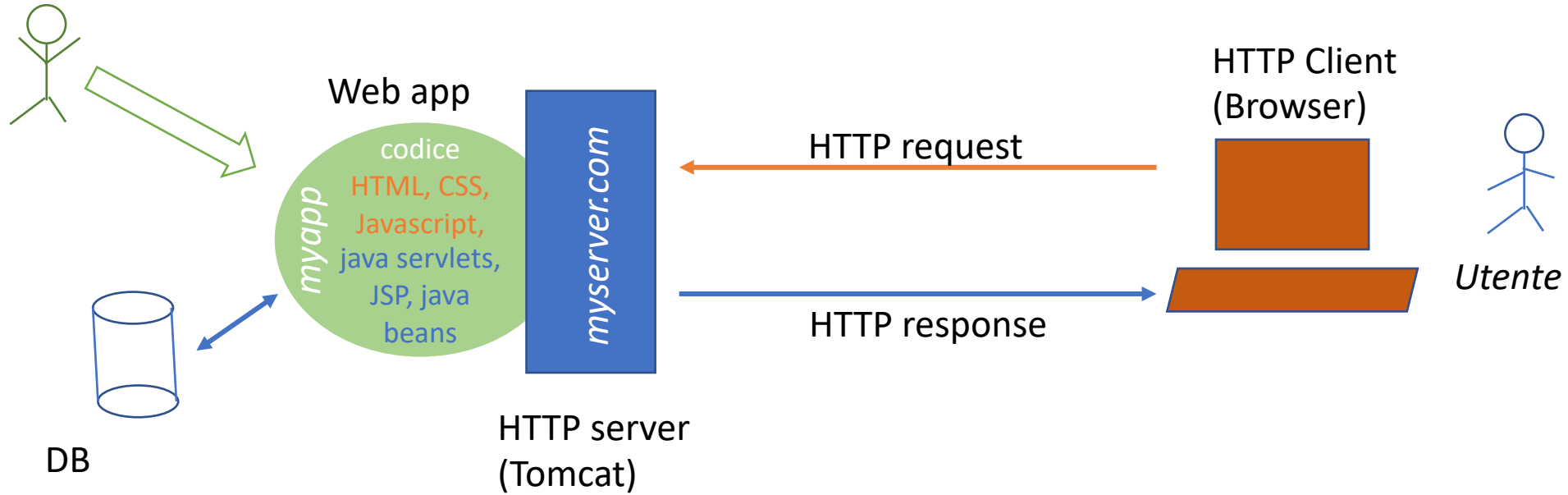


Il modello MVC per le applicazioni web

Architettura Web

Tu - Programmatore



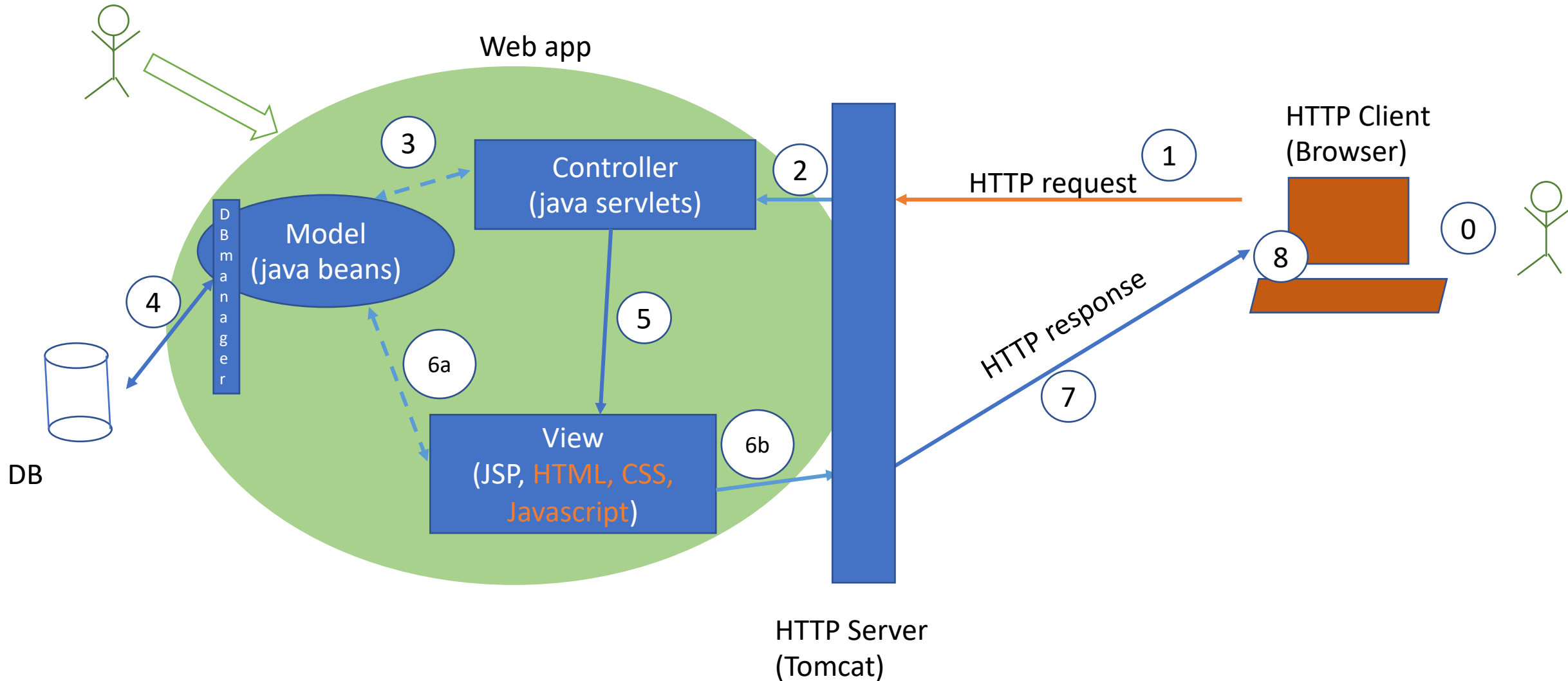
<http://myserver.com/myapp>

Contesto di un'applicazione web

- Un'applicazione web MVC è deployed (istallata) su un **Application Server** o semplicemente un **Servlet Container** come Tomcat. Sia un Application Server che un Servlet Container sono HTTP servers.
- Tramite Tomcat è in grado di ricevere messaggi HTTP di richiesta e di inviare messaggi HTTP di risposta da/a un browser (HTTP client).
- L'applicazione interagisce con un database (server) per aggiornare/interrogare dati

Web app: *modello base*

Model-View-Controller (MVC)



Componenti di un'applicazione web secondo il modello MVC (Model-View-Controller)

- Controller
 - Servlets: classi java che utilizzano la **libreria servlet-api.jar** della distribuzione di Tomcat per interagire con i messaggi di Request e Response (e tanto altro).
Leggono i parametri inviati dal browser, aggiornano/interrogano il Model e mandano i risultati di ciò alla View.
- Model
 - Classi java sia per contenere dati (javabeans) che per gestire la logica dell'applicazione. Queste di solito aggiornano/interrogano la base dati tramite il DB manager e restituiscono i risultati al Controller
- View
 - JSP: particolari servlet che si presentano come template di pagine web (HTML).
Vengono invocate dal Controller che dà loro anche accesso ai javabeans contenenti i dati risultanti dal Model. I template vengono istanziati con questi dati per formare le pagine finali (in HTML, CSS, Javascript, etc.) da inserire nella Response.

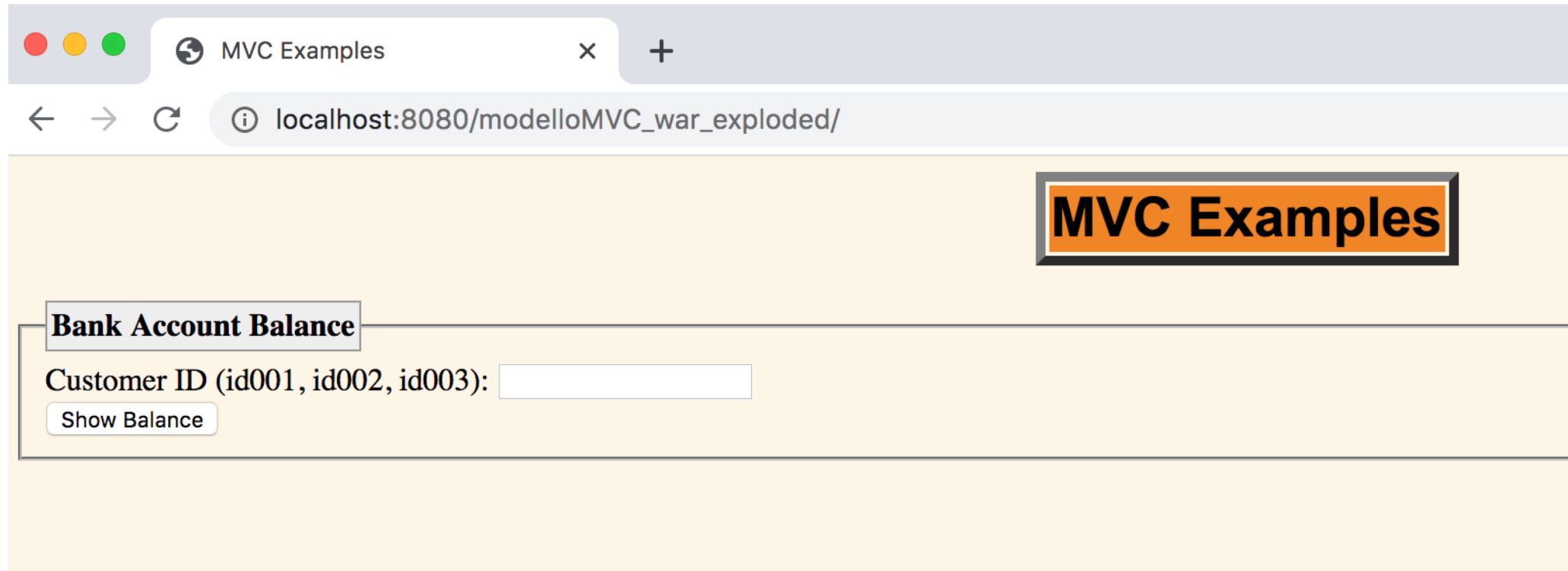
Comportamento di un'applicazione MVC

L'utente richiede una risorsa attivando una URL (cliccando su un link, inserendo un indirizzo web nel browser, cliccando submit in un form, etc.)

0. L'utente dopo aver ottenuto* la homepage dall'app (magari con un form) inserisce dati e fa il submit del form.
1. Il browser costruisce una HTTP Request con informazioni prese dalla URL prodotta dal form e l'invia, tramite internet, al server HTTP interessato (Application server) che la traduce in oggetto java e la passa all'applicazione MVC ed in particolare alla sua componente Controller.
2. Il Controller riceve dall'Application server la HTTP Request tradotta dall'Application Server in oggetto Java.
3. Il Controller legge l'oggetto di richiesta HTTP, ed utilizza i dati estratti per invocare il codice costituente il Model. In particolare, predispone oggetti java che dovranno contenere le risposte del Model (javabean) ed invia la sue richieste al Model.
4. Il Model, utilizzando i parametri ricevuti, interroga la base di dati a cui è collegato e restituisce i risultati nei javabean.
5. A seconda di tali risultati il Controller invia i javabean ad una particolare componente della View e gli passa il controllo.
6. La View legge i dati dal javabean (a), li formatta in una pagina web, inserisce tale pagina nell'oggetto di risposta e invia questa all'Application Server (b).
7. L'Application Server traduce l'oggetto java di risposta ricevuto dalla View in un messaggio HTTP Response e lo invia al browser richiedente.
8. Il browser legge il messaggio HTTP Response, interpreta il codice HTML, CSS, javascript, etc. della pagina web e mostra a video il risultato finale.

*Si noti che al passo 0. affinché l'utente ottenga la homepage dall'app deve esserci necessariamente un altro giro di request/response

Un esempio : ShowBalance



MVC Examples

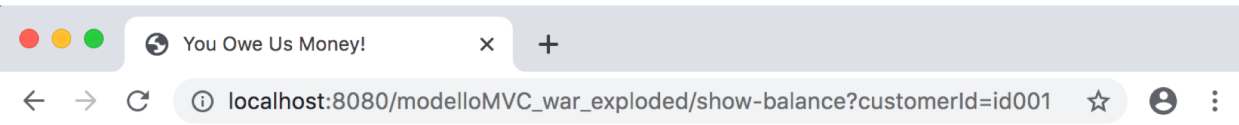
Bank Account Balance

Customer ID (id001, id002, id003):

Show Balance

Si usi IntelliJ per aprire il progetto modelMVC ottenuto dal file modelMVC.zip

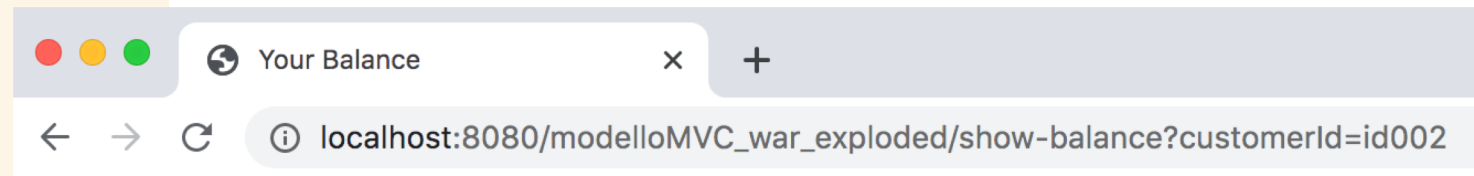
ShowBalance output su 'id001' e 'id002'



We Know Where You Live!

Watch out, Harry, we know where you live.

Pay us the \$3.456,78 you owe us before it is too late!



Your Balance



- First name: Codie
- Last name: Coder
- ID: id002
- Balance: \$1234.56

ShowBalance output su 'id003' e 'pippo'

