Esercizio 4 versione pure HTML

Luca Lain - Sergio Lupo

Esercizio 4: trasferimento denaro

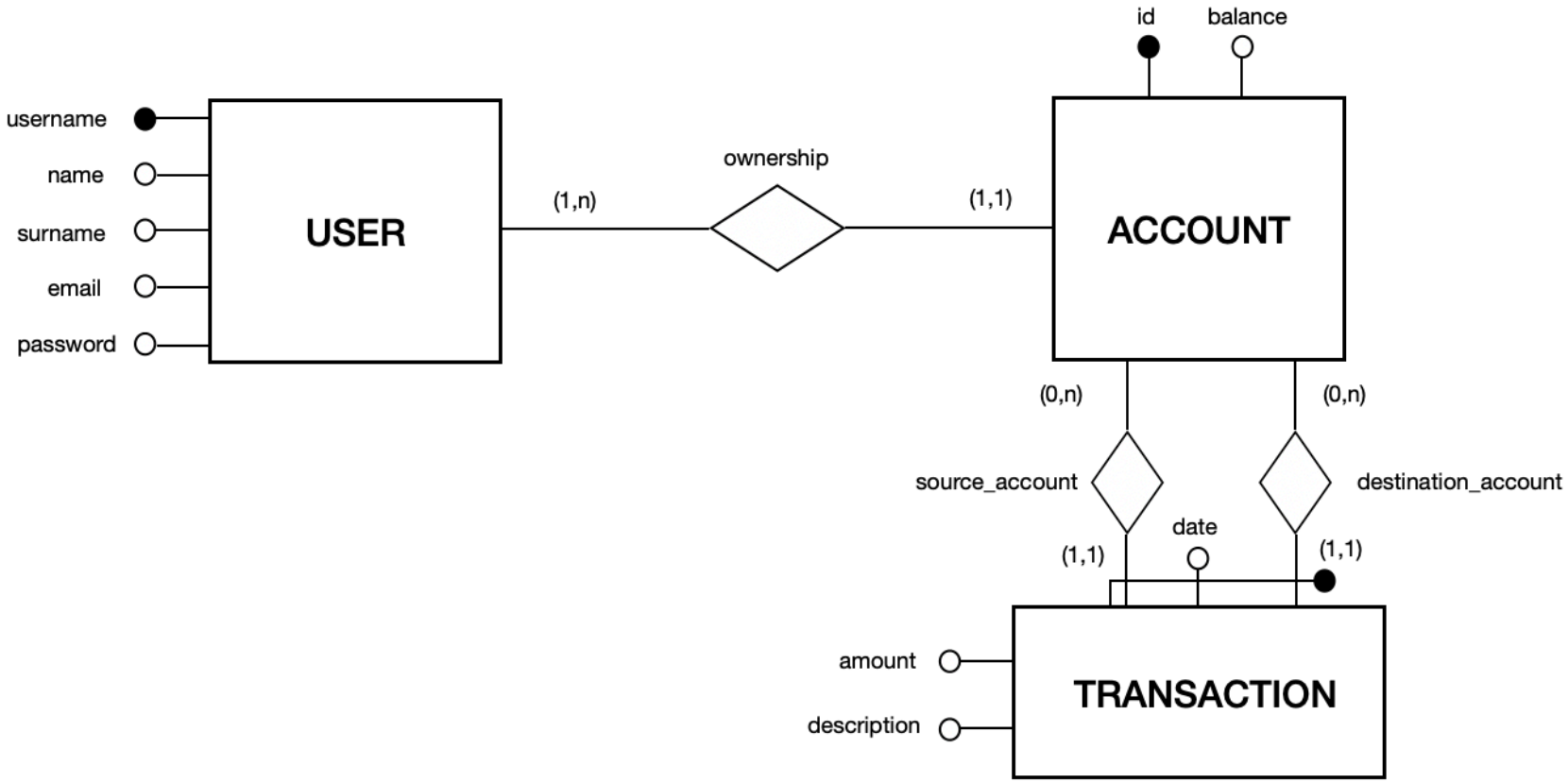
Un’applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. L’applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell’indirizzo di email e l’uguaglianza tra i campi “password” e “ripeti password”. La registrazione controlla l’unicità dello username. Un utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l’utente accede all’applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all’autenticazione dell’utente appare l’HOME page che mostra l’elenco dei suoi conti. Quando l’utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All’invio della form con il bottone INVIA, l’applicazione controlla che il conto di destinazione appartenga all’utente specificato e che il conto origine abbia un saldo superiore o uguale all’importo del trasferimento. In caso di mancanza di anche solo una condizione, l’applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l’applicazione deduce l’importo dal conto di origine, aggiunge l’importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell’importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L’applicazione deve garantire l’atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L’applicazione consente il logout dell’utente.

Analisi dati per database

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username. Un utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

entities, attributes, relationships

Database Design



Database Schema (1/2)

```
CREATE TABLE `user` (
  `username` varchar(255) NOT NULL,
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB;
CREATE TABLE `account` (
  `id` int NOT NULL AUTO_INCREMENT,
  `balance` float NOT NULL,
  `idUser` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idUser` (`idUser`),
  CONSTRAINT `idUser` FOREIGN KEY (`idUser`) REFERENCES `user` (`username`) ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Database Schema (2/2)

```
CREATE TABLE `transaction` (
  `sourceAccount` int NOT NULL,
  `destinationAccount` int NOT NULL,
  `id` int NOT NULL AUTO_INCREMENT,
  `date` date NOT NULL,
  `amount` float NOT NULL,
  `description` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `sourceAccount` (`sourceAccount`),
  KEY `destinationAccount` (`destinationAccount`),
  CONSTRAINT `destinationAccount` FOREIGN KEY (`destinationAccount`) REFERENCES `account` (`id`) ON UPDATE CASCADE,
  CONSTRAINT `sourceAccount` FOREIGN KEY (`sourceAccount`) REFERENCES `account` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Analisi Requisiti Applicazione

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username. Un utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

pages(views), view components, events, actions

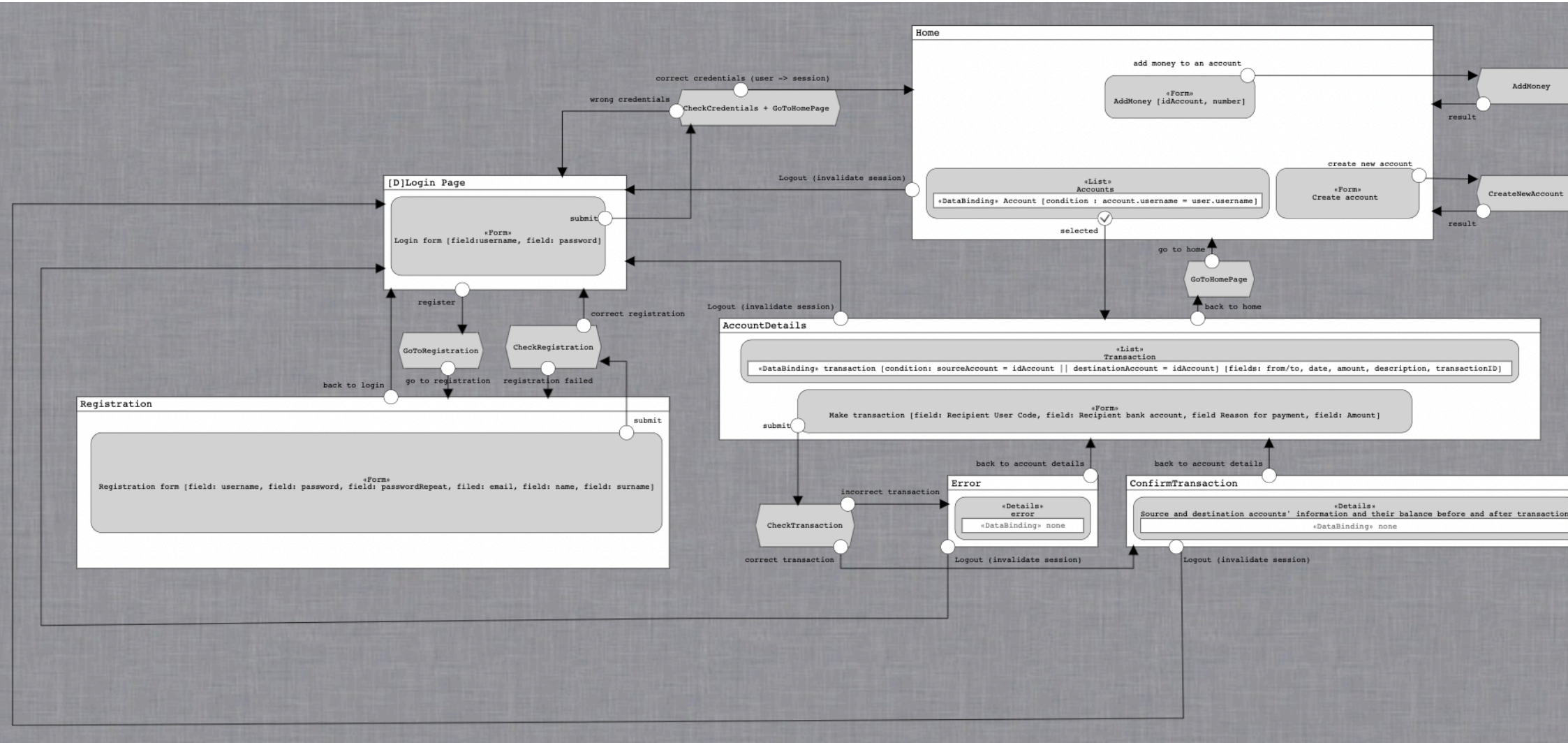
Components

- **Model objects (Beans)**
  - User
  - Account
  - Transaction
- Data access object (DAO)
  - UserDAO
    - InsertUser(User user,String password) : boolean
    - checkLogin(String usr, String pwd) : User
  - AccountDAO
    - getAccountsFromIdUser(String idUser) : ArrayList<Account>
    - getAccountFromId(int id) : Account
    - updateAccounts(Account source, Account destination,Transaction transaction) : boolean
    - insertNewAccount(User user) : boolean
    - addMoney(int id, float num) : boolean
  - TransferDAO
    - getTransactions(int idUser) : ArrayList<Transaction>
- Controllers(Servlets)
  - Logout [loggedUser]
  - GoToRegistration [notLoggedUser]
  - GoToHomePage [loggedUser]
  - GetAccountDetails [loggedUser]
  - CreateNewAccount [loggedUser]
  - CheckTransaction [loggedUser]
  - CheckRegistration [notLoggedUser]
  - CheckCredentials [all]
  - AddMoney [loggedUser]
- Views(Templates)
  - index.html (the login page)
  - Registration.html
  - Home.html
  - Error.html
  - ConfirmTransaction.html
  - AccountDetails.html
- Filters
  - LoginChecker
  - UserNotLogged

Completamente delle specifiche

- Ogni utente viene identificato dal suo username.
- Per credenziali di login si intendono username e password.
- Se l'utente è loggato, indipendentemente dalla pagina in cui si trova, può fare logout e tornare quindi alla pagina di login.
- Se l'utente è loggato e tenta di accedere alla pagina REGISTRATION, verrà reindirizzato automaticamente alla HOME. Invece se l'utente non è loggato e tenta di accedere ai dati dei suoi conti o fare transazioni su di essi, verrà reindirizzato alla pagina di LOGIN.
- Se in un qualsiasi momento si verificano azioni impreviste (attributi della richiesta invalidi o tentato accesso a pagine per cui non ha l'autorizzazione), l'utente viene indirizzato a una pagina di ERRORE contenente il motivo dell'errore.
- Dalla pagina di login, è possibile spostarsi nella pagina REGISTRATION, dove inserendo nome, cognome, username, email, password e la ripetizione della password in una form, è possibile creare un utente; se l'username inserito è già in uso l'utente viene riportato alla pagina REGISTRATION in cui gli si mostra un messaggio di errore, altrimenti viene indirizzato alla pagina di LOGIN.
- Non è possibile fare transazioni in cui il conto origine sia uguale al conto destinazione.
- L'importo del trasferimento deve essere sempre positivo.
- Dalla pagina HOME, l'utente inoltre può creare un nuovo conto e aggiungere soldi ad un conto già esistente.

Design Applicativo (IFML)



Sequence diagrams

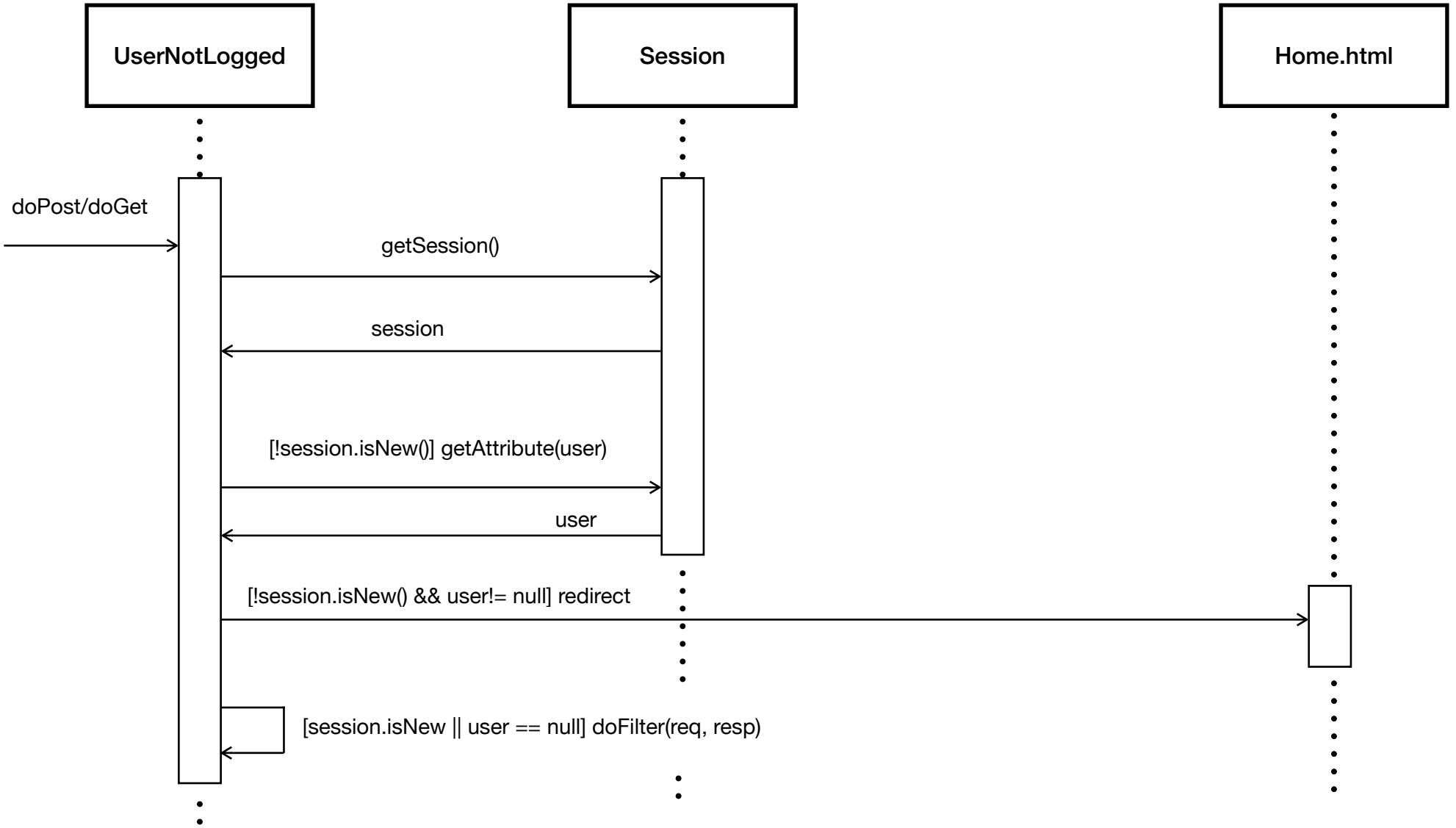
Di seguito sono riportati i sequence diagram che rappresentano gli eventi principali dell'applicazione Web.

Alcuni dettagli minori sono stati tralasciati, ovvero gli errori interni del server, di accesso al database e l'inserimento di parametri nulli.

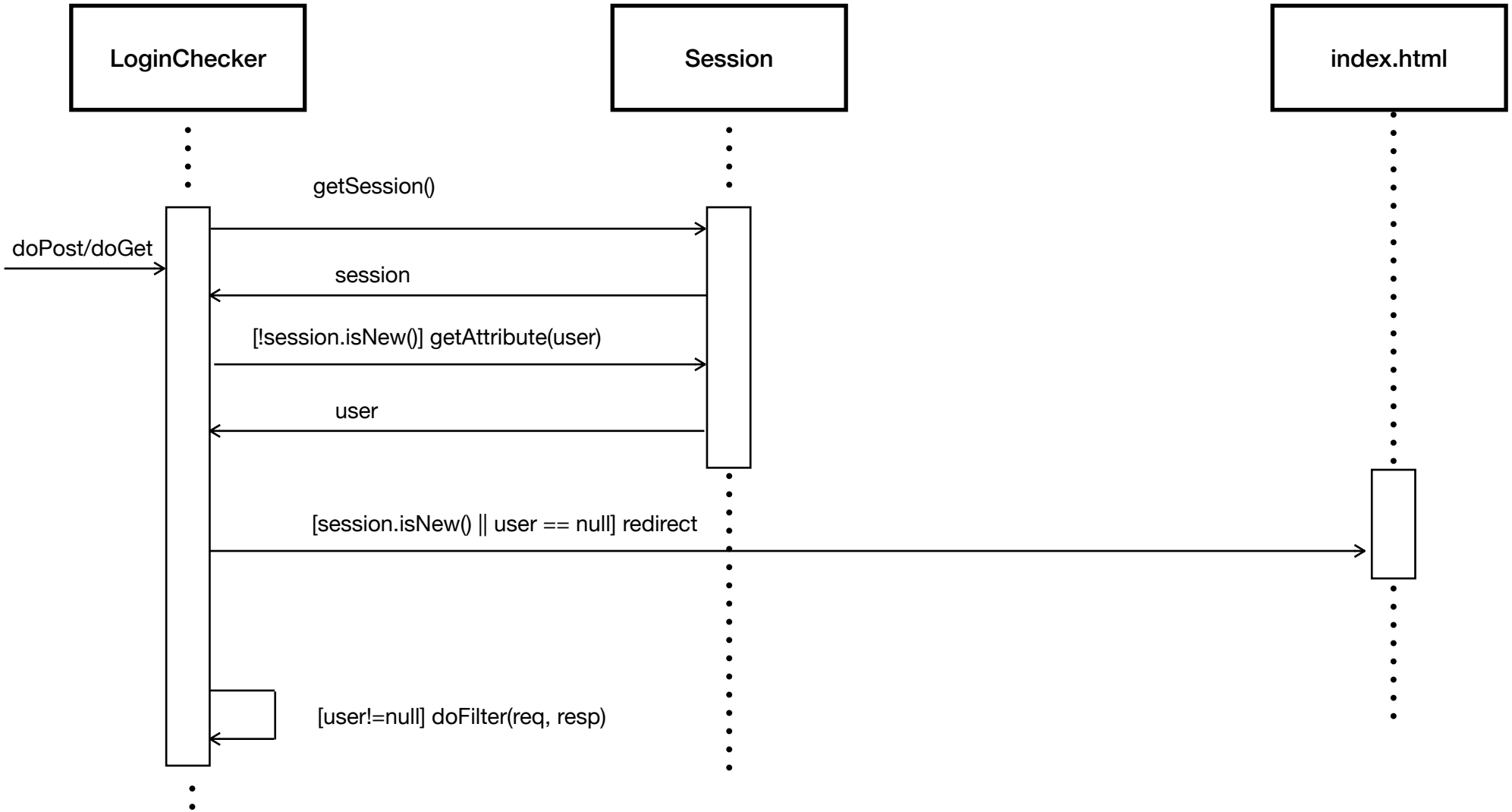
Per semplicità i controlli dei filtri sono rappresentati nei primi schemi e sono omessi in quelli successivi.

Per ogni sequence diagram è riportato in alto il nome dell'evento che rappresenta.

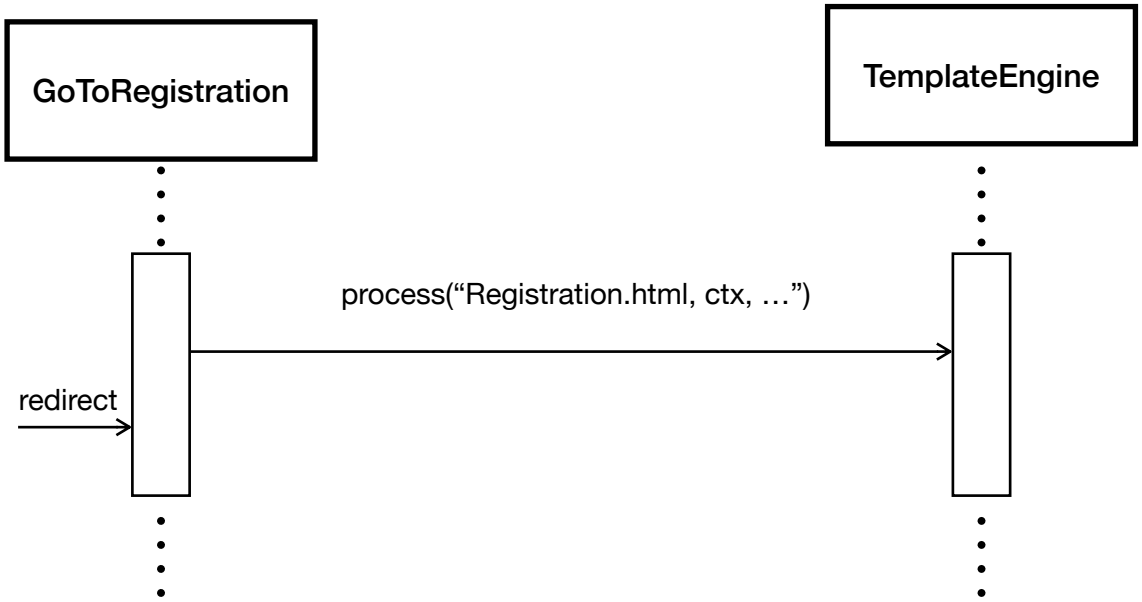
User Not Logged (Filter)



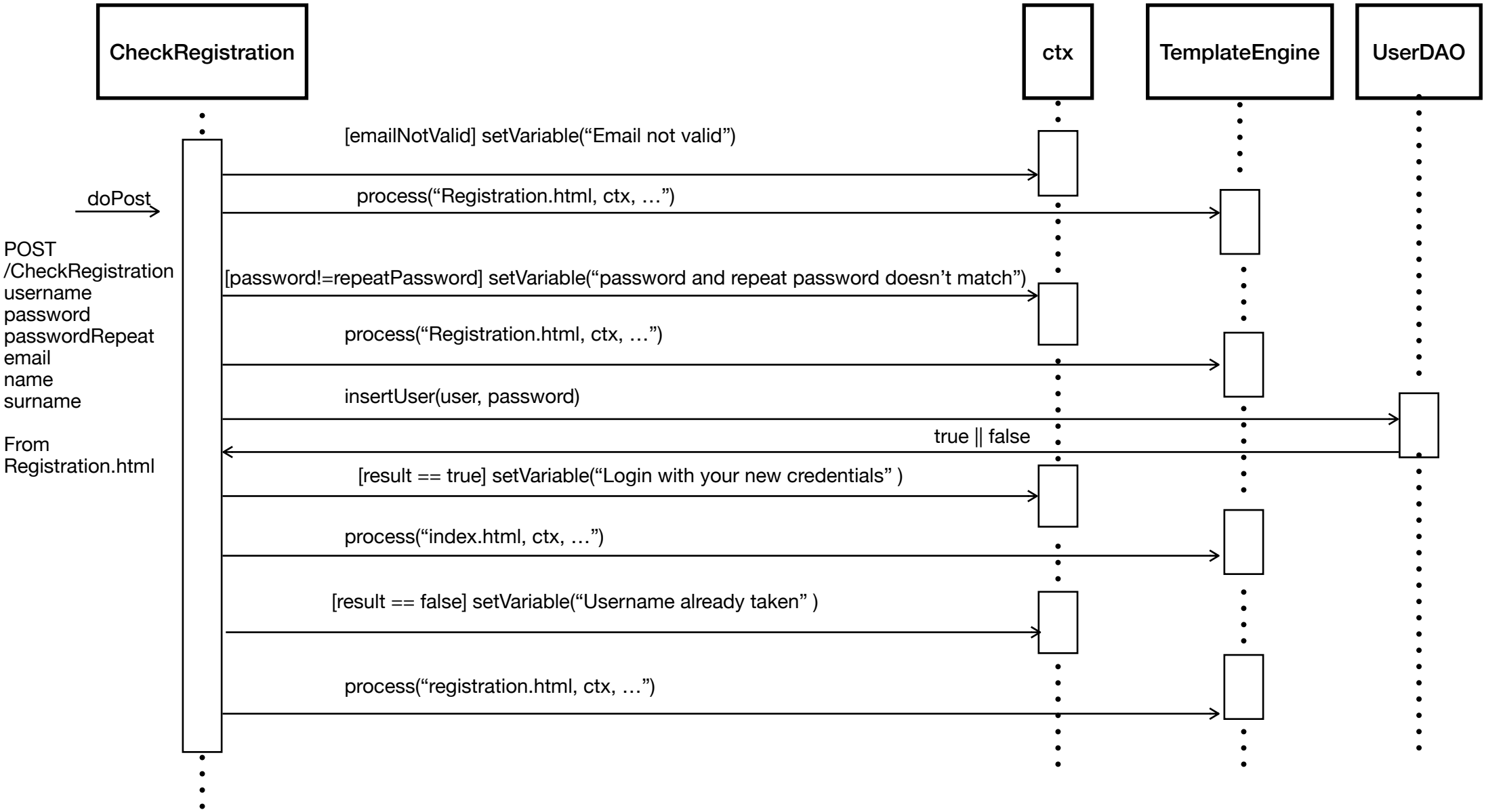
Login Checker (Filter)



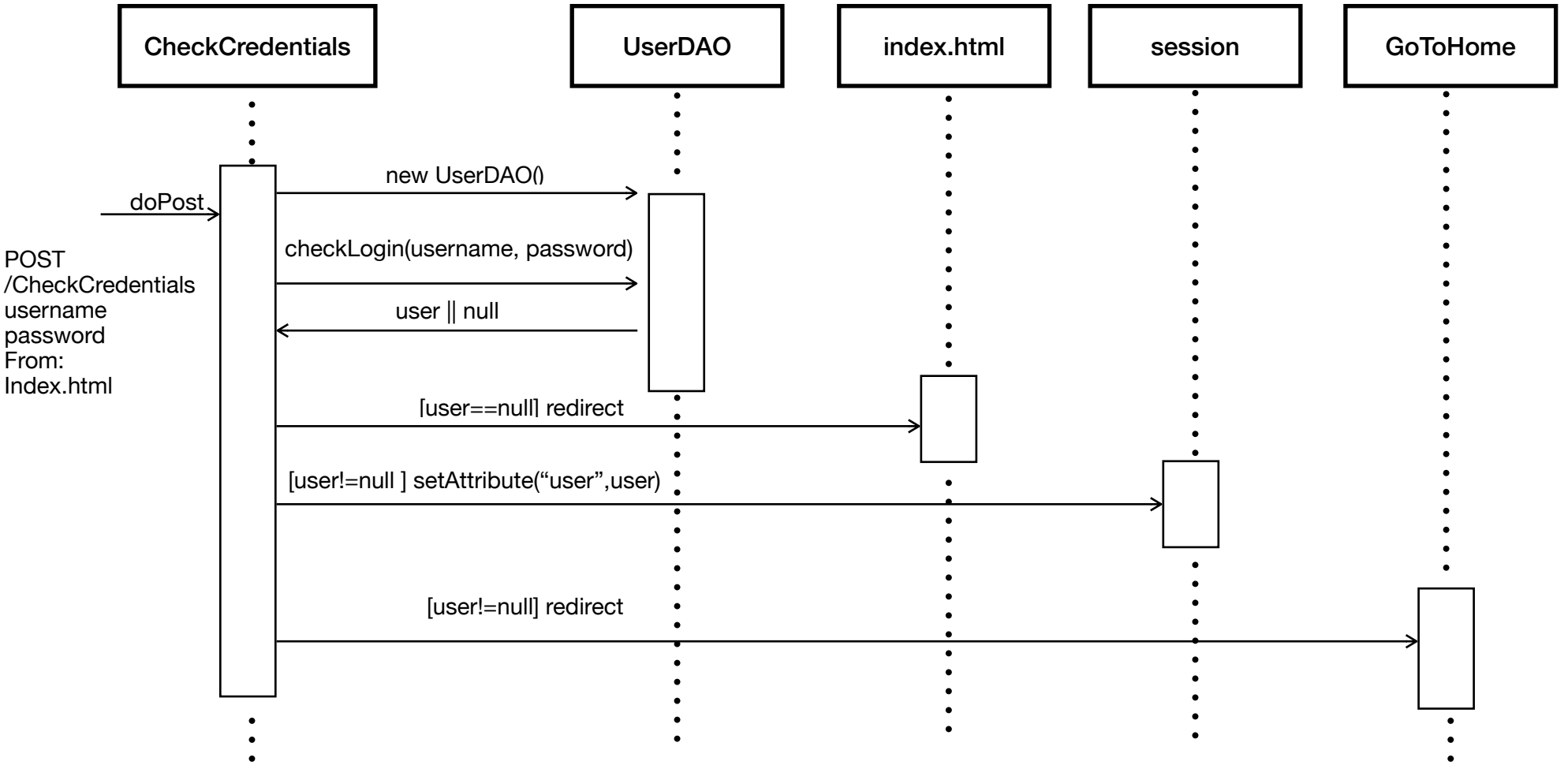
Go To Registration



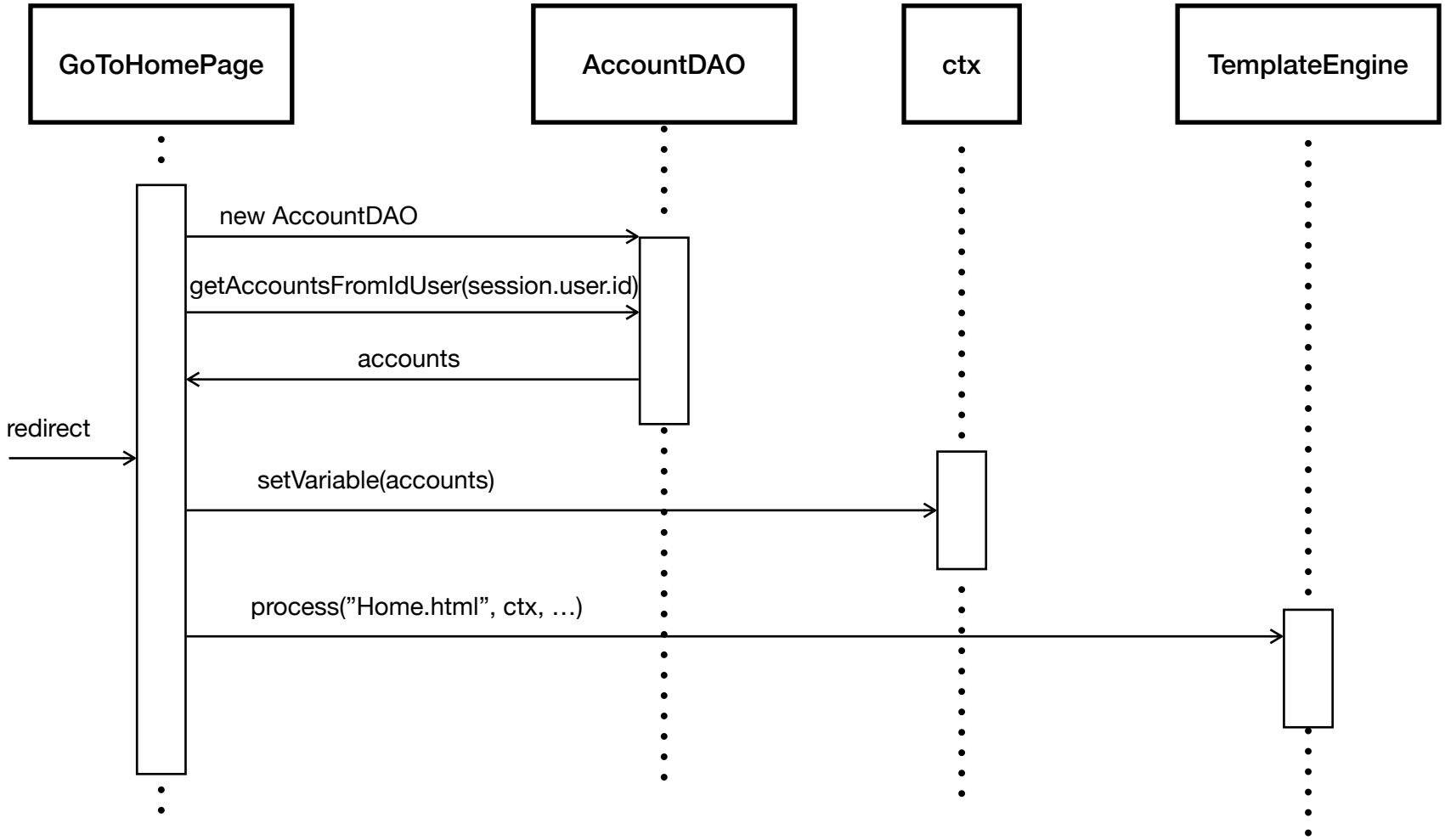
Check Registration



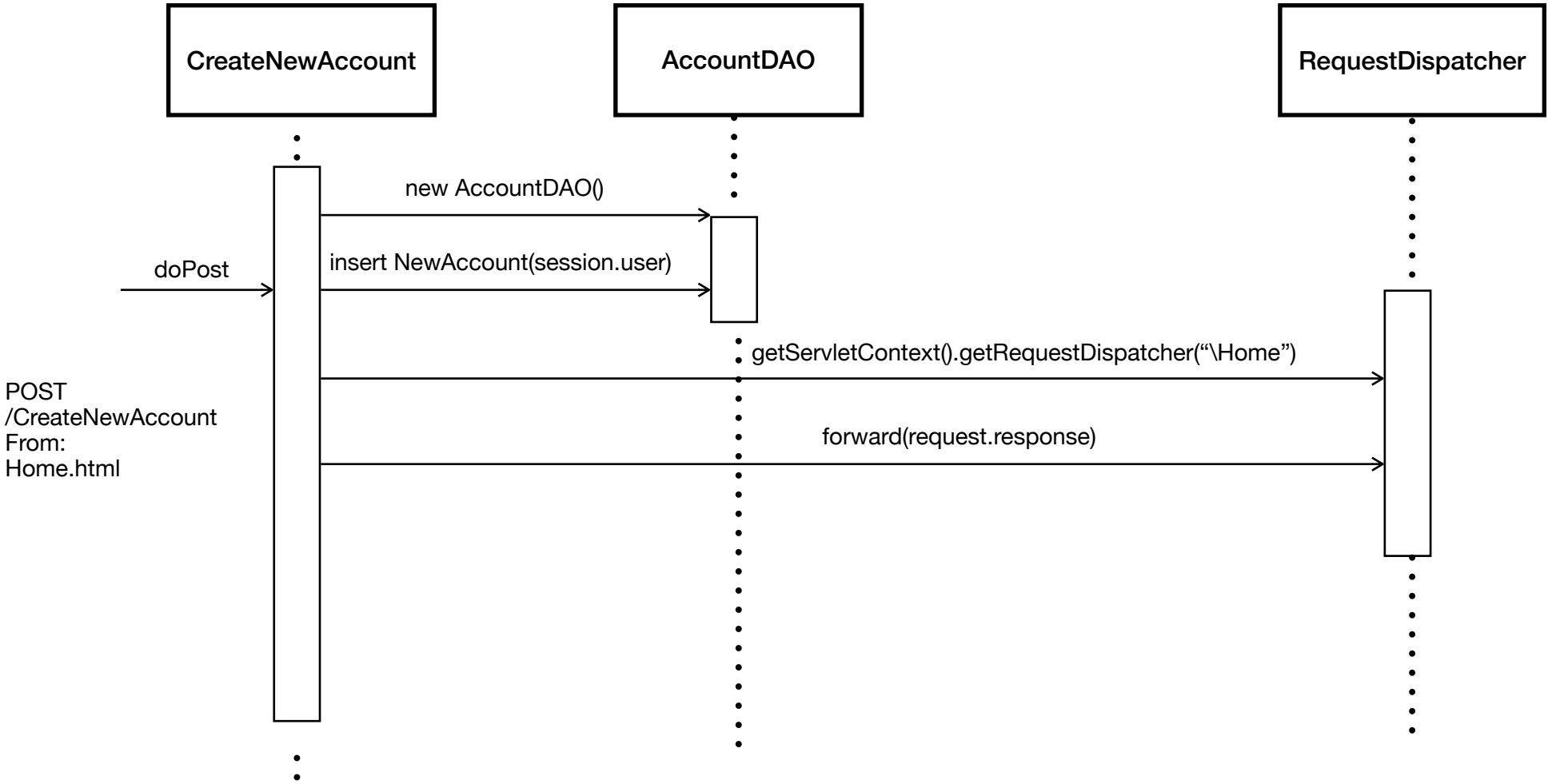
Login


CheckCredentials
UserDAO
index.html
session
GoToHome
doPost
POST
/CheckCredentials
username
password
From:
Index.html
new UserDAO()
checkLogin(username, password)
user || null
[user==null] redirect
[user!=null] setAttribute(“user”,user)
[user!=null] redirect

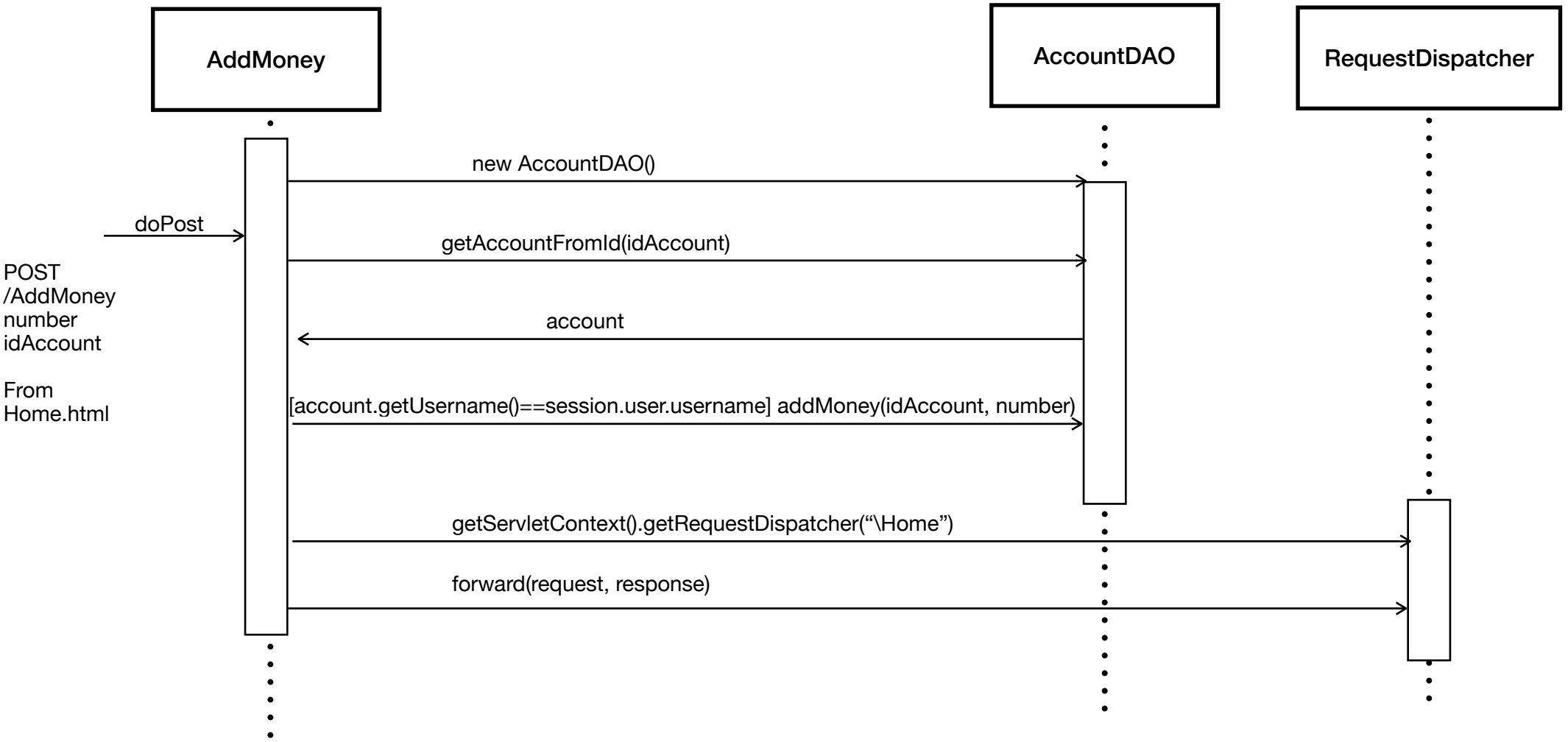
Go to Home


GoToHomePage
AccountDAO
ctx
TemplateEngine
new AccountDAO
getAccountsFromIdUser(session.user.id)
accounts
redirect
setVariable(accounts)
process(”Home.html”, ctx, …)

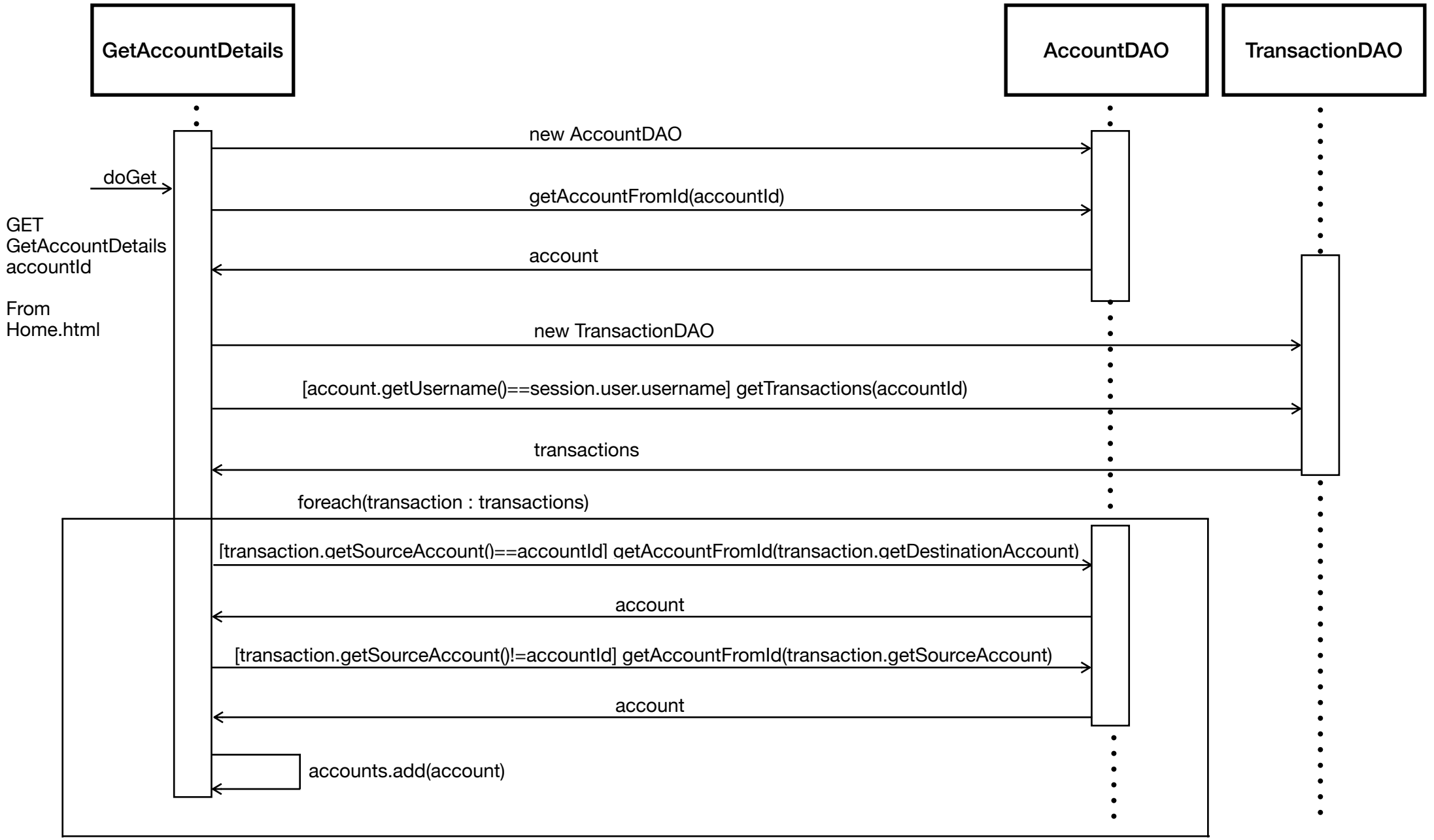
Create New Account

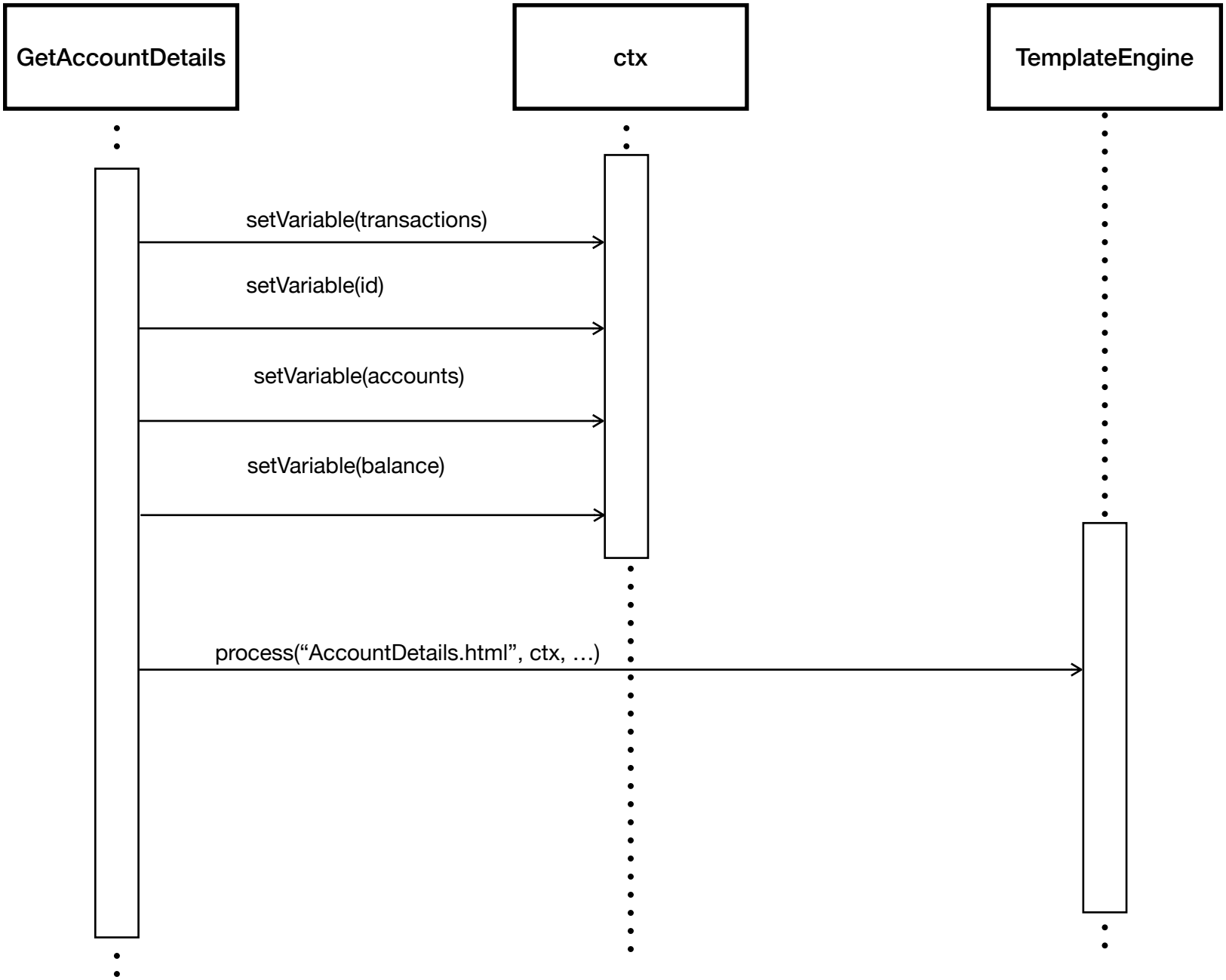


Add Money

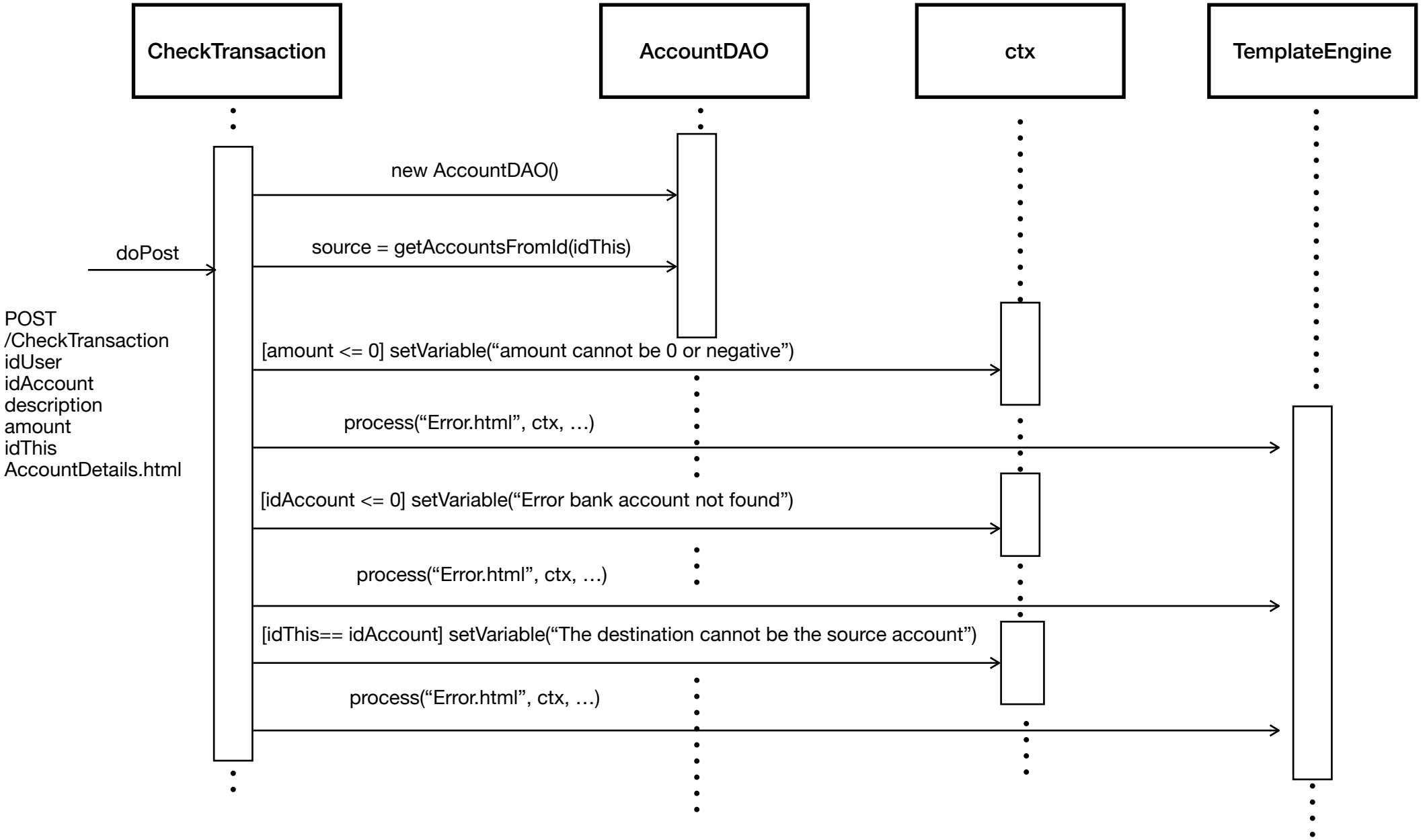

AddMoney
AccountDAO
RequestDispatcher
new AccountDAO()
doPost
getAccountFromId(idAccount)
POST
/AddMoney
number
idAccount
From
Home.html
account
[account.getUsername()==session.user.username] addMoney(idAccount, number)
getServletContext().getRequestDispatcher("\Home")
forward(request, response)

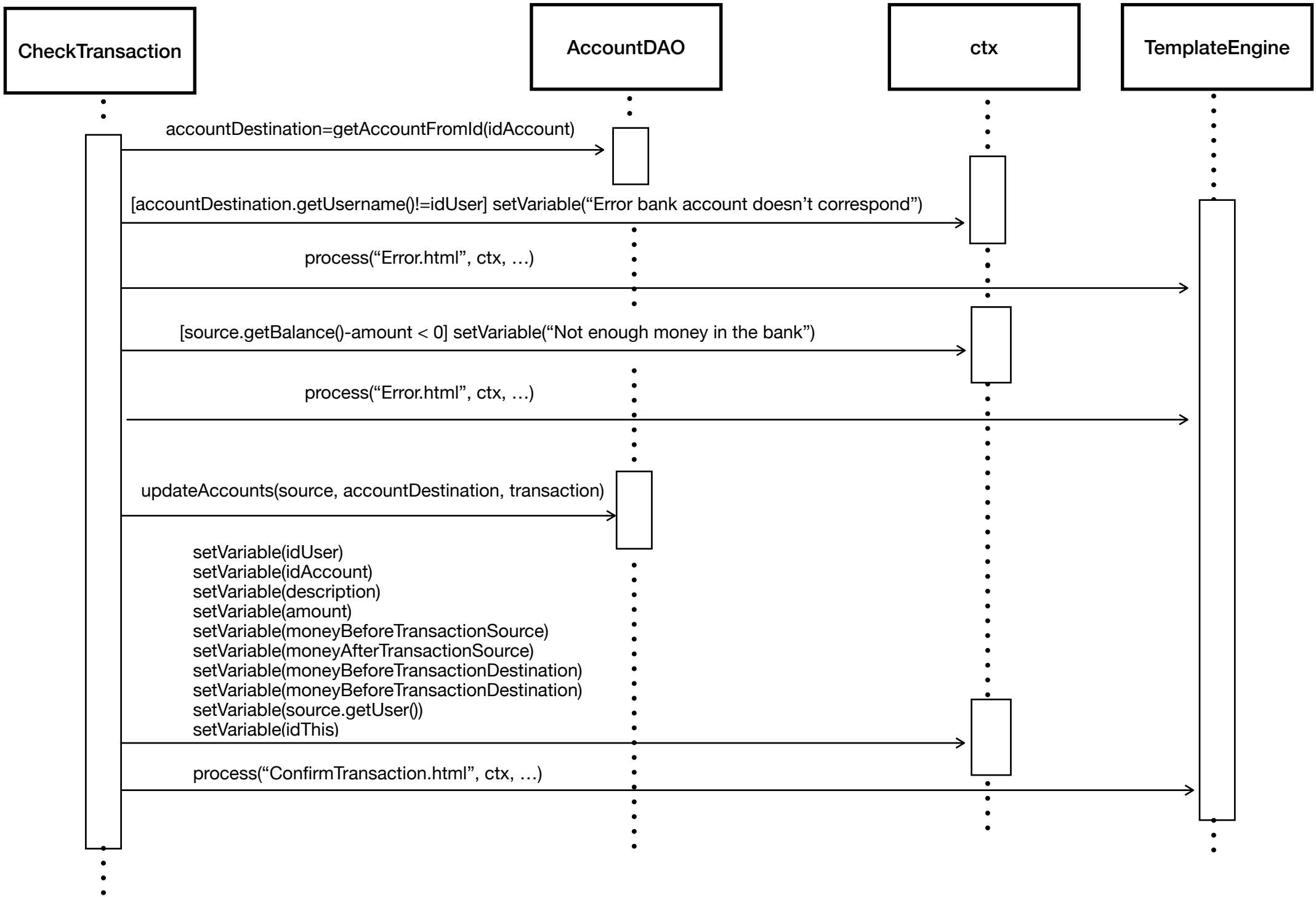
Show account details



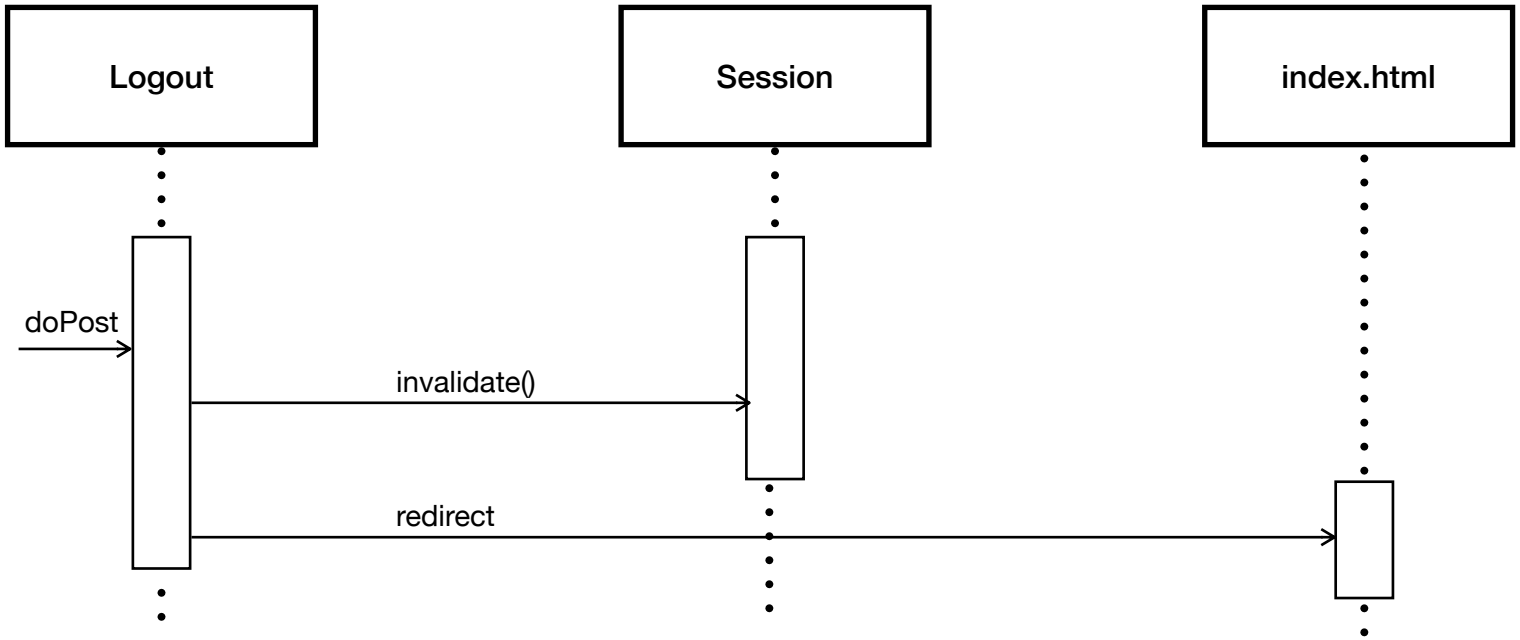

GetAccountDetails
ctx
TemplateEngine
setVariable(transactions)
setVariable(id)
setVariable(accounts)
setVariable(balance)
process(“AccountDetails.html”, ctx, …)

Check Transaction




CheckTransaction
AccountDAO
ctx
TemplateEngine
accountDestination=getAccountFromId(idAccount)
[accountDestination.getUsername()!=idUser] setVariable("Error bank account doesn't correspond")
process("Error.html", ctx, …)
[source.getBalance()-amount < 0] setVariable("Not enough money in the bank")
process("Error.html", ctx, …)
updateAccounts(source, accountDestination, transaction)
setVariable(idUser)
setVariable(idAccount)
setVariable(description)
setVariable(amount)
setVariable(moneyBeforeTransactionSource)
setVariable(moneyAfterTransactionSource)
setVariable(moneyBeforeTransactionDestination)
setVariable(moneyBeforeTransactionDestination)
setVariable(source.getUser())
setVariable(idThis)
process("ConfirmTransaction.html", ctx, …)

Logout



POST
/Logout

From
AccountDetails.html
ConfirmTransaction.html
Error.html
Home.html