Presentatie Groepswerk

Inleiding

Per pagina(servlet) hebben we de functionaliteit besproken op bezoeker en auteur niveau.

Besproken welke frameworks we gaan gebruiken( hibernate, mySQL, js, Html,css,..)

Diagram gemaakt om de onderliggende relaties tussen de servlets in kaart te brengen.

Uitwerking

Model:

We zijn begonnen met het uitschrijven van de versillende Entiteiten: Author, Blog en Comment. Elk met hun verschillende eigenschappen en methoden. Getters en setters en toString. We hebben ook een *EntityInterface* aangemaakt met één methode: *cloneFrom()* die we in elke klasse implementeren.

Data:

Voor de connectiviteit met de database hebben we met hibernate gewerkt. Voor deze verbinding hebben we een *EntityManagerFactoryProvider* klasse aangemaakt.

Om geen Dao van elke entiteit afzonderlijk te moeten schrijven heeft manuel een *GenericMapper* geschreven waarbij we bij elke methode een object kunnen meegeven. Deze klassen bevat de volgende methoden: addObject, editObject, removeObject, getObject.

Listener:

Voor onze database werken we met mySQL. We hebben onze database reeds voorzien van een aantal authors, blogs, comments om zo tests te en de demo te kunnen uitvoeren. Ulas heeft een aantal fictieve authors, etc geschreven die we door middel van een *Initialiser* in de database laden. Dit is een klasse die *ServletContextListener* implementeerd. Deze bevat een methode *contextInitialized()* waardoor al onze data automatisch in de database geladen wordt zonder dat we dit manueel in de main hoeven te doen. We hebben hiervoor enkel de initialiser door middel van een listener toegevoed aan de web.xml.

In deze map hebben we ook een SessionListener voozien die de klasse HttpSessionListener implementeerd. Deze is voorzien van een sessionList waarin alle sessions worden bijgehouden. –*sessionCreated()* is een methode die ervoor zorgt dat wanneer er een nieuwe sessie is deze wordt opgeslagen in de sessionList. -*sessionDestroyed()* zorgt ervoor dat wanneer een user offline gaat de sessie wordt verwijderd uit de lijst. -*amountOfActiveSessions()* geeft het aantal actieve sessions weer. Deze methode gebruiken we later nog in onze sidebar om het aantal actieve users weer te geven.

Servlets:

Get

In onze Servlet map hebben we de verschillende pagina’s(servlets) gemaakt. We hebben deze map verdeeld in 2 submappen get en post. In de get folder zitten de servlets die de doGet methode implementeren. *- HomeServlet*: Dit is onze homepage waar we 6 blogs laten zien.

- *LoginServlet:* Aan de hand van een sessie en de methode isLoggedIn van de SessionControler(zie later) gaan we na of een user al ingelogd is. Indien dit zo is krijgt de user een alertbox met dit bericht. We gaan hier ook nog nagaan of de user de matching username/password ingeeft. Hiervoor gebruiken we de methode isInvalidCredentialsFlagset die na gaat of de persoon/pw bestaat.

*- RegiserServlet:* Hier wordt nagegaan of de user al ingelogd is. Zoniet gaat hij naar het registerform.

*- SearchServlet:* Zoeken naar Blogs/authors op tags, titels,etc ...

- *BlogdetailServlet:* Hier geven we een gedetailleerde versie van die blog weer. Op basis van het unieke blod-id gaan we op zoek in de database naar de blog.

- *AuthorServlet:* Op basis van de unieke author name gaan we op zoek in de database naar de overeenkomstige author. Hier geven we een lijst weer van de blogs geschreven door deze auteur.

Post

In de post folder vinden we de LoginPostServlet, LogoutServlet, RegisterPostServlet, DeleteCommentServlet en de SendCommentOnPost. Deze servlets implementeren de doPost methode.

- *LoginPostServlet:* Hier gaan we de parameters username en password opvragen. Aan de hand van de username gaan we op zoek naar de overeekomstige auteur in de database en kijken of zijn password matched. We gebruiken hiervoor een instantie van *GenericMapper* die we authormapper noemen. Met de methode *getObject* van *GenericMappe*r gan we op zoek in de database naar de overeenkomstige auteur. Met de methode *checkPw* gaan we na of the pw dat de user ingeeft overeenkomt met het password uit de database.

- *LogoutPostServlet:* Hier gaan we de boolean *isLoggedIn* op false zetten.

*- RegisterPostServlet:* Hier vragen we alle gegevens van de user en laden we deze in de database door middel van de addObject() methode van de GenericMapper. Vervolgens zetten we de user op Logged in en sturen we die terug naar de pagina vanwaar hij kwam.

- *SendCommentOnPost:* Deze servlet zorgt ervoor dat de user een comment kan posten op een blog. We gaan de comment opslaan in de database met de overeenkomste blogID.

- *DeleteCommentServlet:* Door middel van het commentID gaan we de comment in de database verwijderen.

In onze Servlet map hebben we nog een extra folder, de help folder. In deze hebben we 2 servlets namelijk de *AlreadyLoggedInServlet* en de *InvalidCredentialsServlet.*

Deze 2 servlets verwijzen naar een jsp pagina waar die een alert tonen. Already logged in en Invalid credentials wanneer de user verkeerde username/password ingeeft.

Exceptions

Exceptions everywhere. Om onze exceptions op te vangen hebben we 3 klassen gemaakt namelijk *AuthorNotFoundException, AuthorNotLoggedInException en PasswordInvalidException*. Deze geven author not found in the database; User is not logged in, unable to retrieve logged in user en invalid password.

Tools

*- BCrypter*: In onze applicatie maken we gebruik van een database waar onder andere passwords opslaan. Om deze veilig op te slaan maken we gebruik van een Bcrypter klasse. Hierin hebben we 2 methodes. De eerste is de hashPassword waar me een String password meegeven. Deze zorgt ervoor dat het password gehashed word en voegt er ook nog een salt aan toe. Dit is een random reeks nummers die uniek is voor elk password waardoor het bijna onmogelijk is om het originele password te achterhalen. We hebben nog een tweed methode, namelijk de checkPassword methode. Hier geven we de String password en het

*- JavaScriptGenerator:*De klasse gebruiken we voor onze alert messages en redirect. We hebben een methode generatedAlertBoxAndRedirect waar we een string message en string link meegeven. De message bv. Already logged in stuurt de user terug naar de pagina vanwaar hij kwam.

*- SessionController:* Tot slot hebben we een klasse waarin we alle methodes omtrent de sessie implementeren. AddNewPageToSessionHistory waarin we de lastPage en currentPage definiëren. Wanneer we vanaf een lege pagina/direct naar de login gaan wordt de homepage automatisch op lastPage gezet. Wanneer de user op de login page of register page terechtkomt, en deze met succes invult gaat hij terug naar de pagina vanwaar hij komt.

- isLoggedIn is een boolean die we in een aantal servlets gebruiken zodat we weten of de user logged in is of niet.

Verder vinden we in deze klasse nog een aantal statische hulpmethoden die ons helpen om onze code gebruiksvriendelijk en leesbaarder te maken.