Cordell Andersen and Erica Wiens
CS-389 Advanced Web and Cloud Applications
Nate Williams
April 30, 2018

Doctor Who Episode Generator

The goal of this project was to create a web app where users could create their own doctor who episodes, featuring a selection of companions and Doctors with their choice of episode name and plot description and the ability to view past episodes that they have created. At this time, we have succeeded in creating a web app that will store user names for episodes, Doctors, companions, and plots. Future work will limit Doctor and companion selection to those that have been released in the actual story and add styling to the app to make it more visually appealing.

How it works:

The application begins by setting up an application context. The context is then told to use the configurations from the configs package. Using the spring framework for dependency injection the dependencies set up in the configs are made into transactionals that can be used in other classes without having to include all of the dependencies. More dependencies and dependency versions are specified in build.sbt. Information for the database configuration is in *application.conf* and information for logging is in *logger.xml*. The *routes* file holds the url information for specific actions and pages of the application.

The app opens to the login screen (*index.scala.html*) which provides the user the option of logging in or switching to another screen to create a new user *LogInApplication.java*.
When the human user chooses to create a new username/password log in combination they are redirected to the new user screen (*createuser.scala.html*) where they can create a new user. This page is controlled by *UserApplication.java* which validates the input and persists the new user to a table in the database using the *UserPersistenceService*.
When the human user logs in with a valid username/password combination, they are redirected to the episode input form (*enterdata.scala.html*). The functionality for this form is in *FormApplicaion.java* which validates the episode input and persists the episode using *EpisodePersistenceService.java*.

Configs:

AppConfig.java : Tells the application where to look for components “controllers” and “services”

DataConfig.java: Sets up the entity manager factory, the database, and the transaction manager. These services are used in the persistence services.

Controllers:

FormApplicaion.java : This controller holds the functionality for the form for entering new episodes. It requires the service *EpisodePersistenceService*, the model *Episode*, and the view *enterdata*.

LogInAppliction.java : This controller holds the functionality for logging users in. If a user successfully enters in a known username and password combination then the user is logged in, otherwise they are given an error message. The user also has the option of clicking a button that will redirect them to *UserApplication.java* where they can create a new user. It requires the service *UserPersistenceService*, the model *User*, and the views *index* and *enterdata.*

UserApplication.java: This controller holds the functionality for creating a new username and password. When a user clicks on the submit button the username is checked to see if it is a duplicate and if it does not contain certain characters. The password is also checked to make sure that it does not contain certain characters. Requires the service *UserPersistenceService*, the model *User*, and the views *createuser* and *enterdata.*

Services:

EpisodePersistenceService.java : This service persists new episodes to the episodes table in the database. It checks to make sure that every field is non-empty and does not contain the characters ";", "$", "#", "-", ">". This service requires the model *Episode* as well as *javax.persistence.EntityManager*, *javax.persistence.PersistenceContext*, *and javax. transaction.Transactional*. The entity manager is set up in *DataConfig.java.*

UserPersistenceService.java : This service persists new users to to the user database and checks the database for username/password combinations. This service validates the usernames and passwords to make sure that they do not contain the characters ";", "/", "~", "`", "#", "-", ">", ",". This service will also check to make sure that every username is unique. It requires the model *User*, and the *javax.persistence* services *EntityManager* and *Persistence Context*. The entity manager is set up in *DataConfig.java.*

Models:

Episode.java: Creates the Episode object that is used for the table of episodes. Requirements:

Name	Required	Min Length	Max Length
episodeName	Y	3	30
doctorName	Y	4	20
companionName	Y	4	30
plotDescription	Y	20	500

User.java: Creates User object that sets up the requirements for the table of users. Requirements:

Name	Required	Min Length	Max Length
username	Y	4	256
password	Y	4	256