Architecture: Seedy Music Club E-Com Site

CSI5380 – Team Seedy Music Club – Fall 2017

Version History

Editor	Date (YYYY-MM-DD)	Comments
Karim Sultan	2017-10-03	First draft of this document.
Karim Sultan	2017-11-07	Updated document with the changes proposed by Saad and Bisi.

Table of Contents

Team Goals

Knowledge

- Answers what is an e-com site?
- What is a JEE based site architecture?
- What are best practices for e-com site creation?

Learning Opportunity

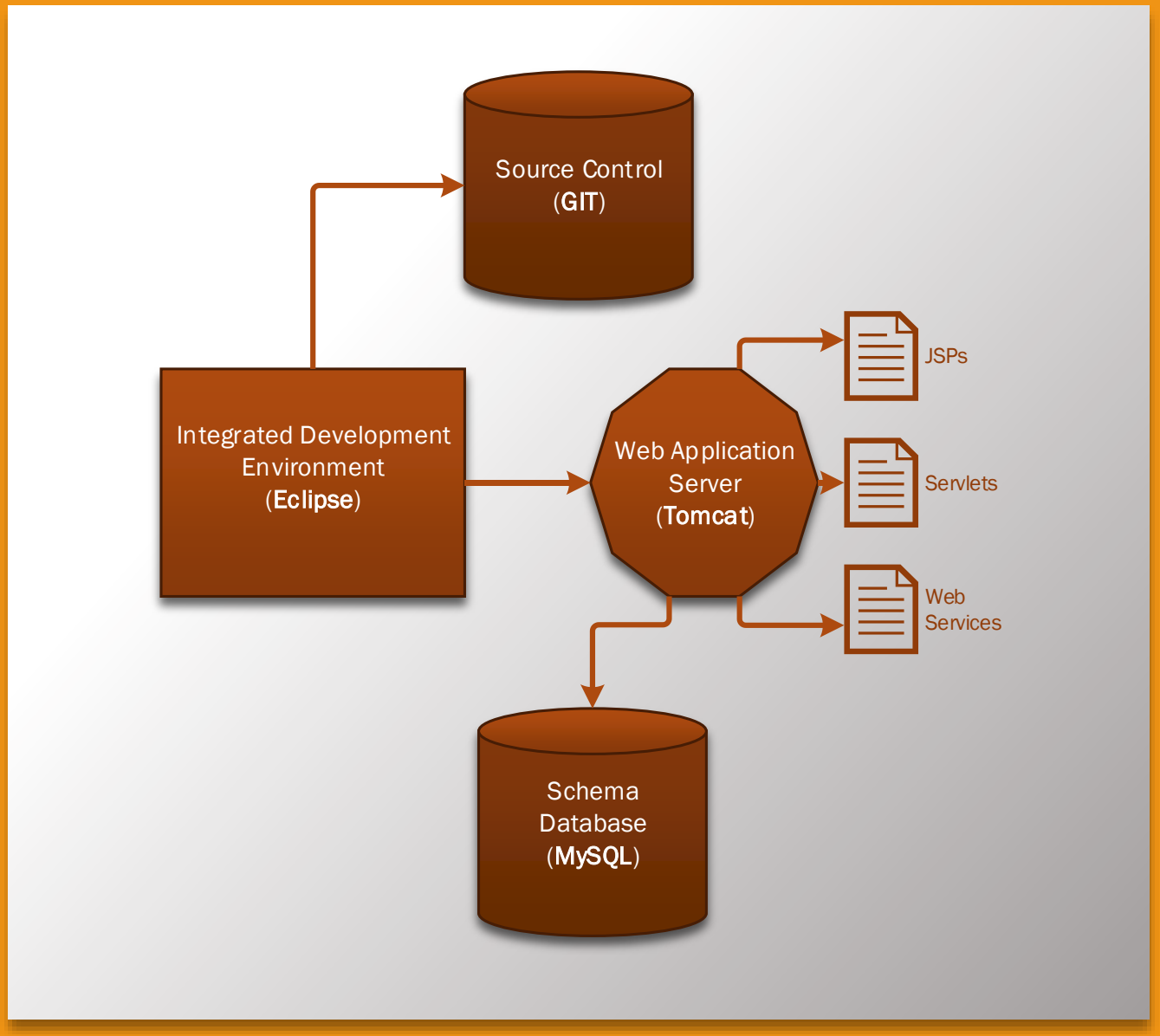
- Chance to learn about e-com sites
- It’s only a Simulation – don’t stress!
- Good shared knowledge in group – don’t be afraid to ask questions!

Experience

- Everyone can try any aspect

Process

- Eclipse IDE will use a `Dynamic Web Project` project type
- Source version control is done using Git
  - This way we all share the same code!
  - Cookbook updated to show Git config
- Eclipse runs code locally in Tomcat
  - Using https on port 8443
  - In production, we use port 443
- Web apps use local database
  - We will all need the same populated DB
  - An SQL schema creation and population script will be required to initialize and rebuild DB
  - We will use Tomcat's DB Connection Pooling
  - In production, we use master DB



Architecture

CSI5380 – Team Seedy Music Club – Fall 2017

Architecture Goals

01

To create a simulated e-com site for a fictional CD store, using JEE

02

To use an MVC design pattern

03

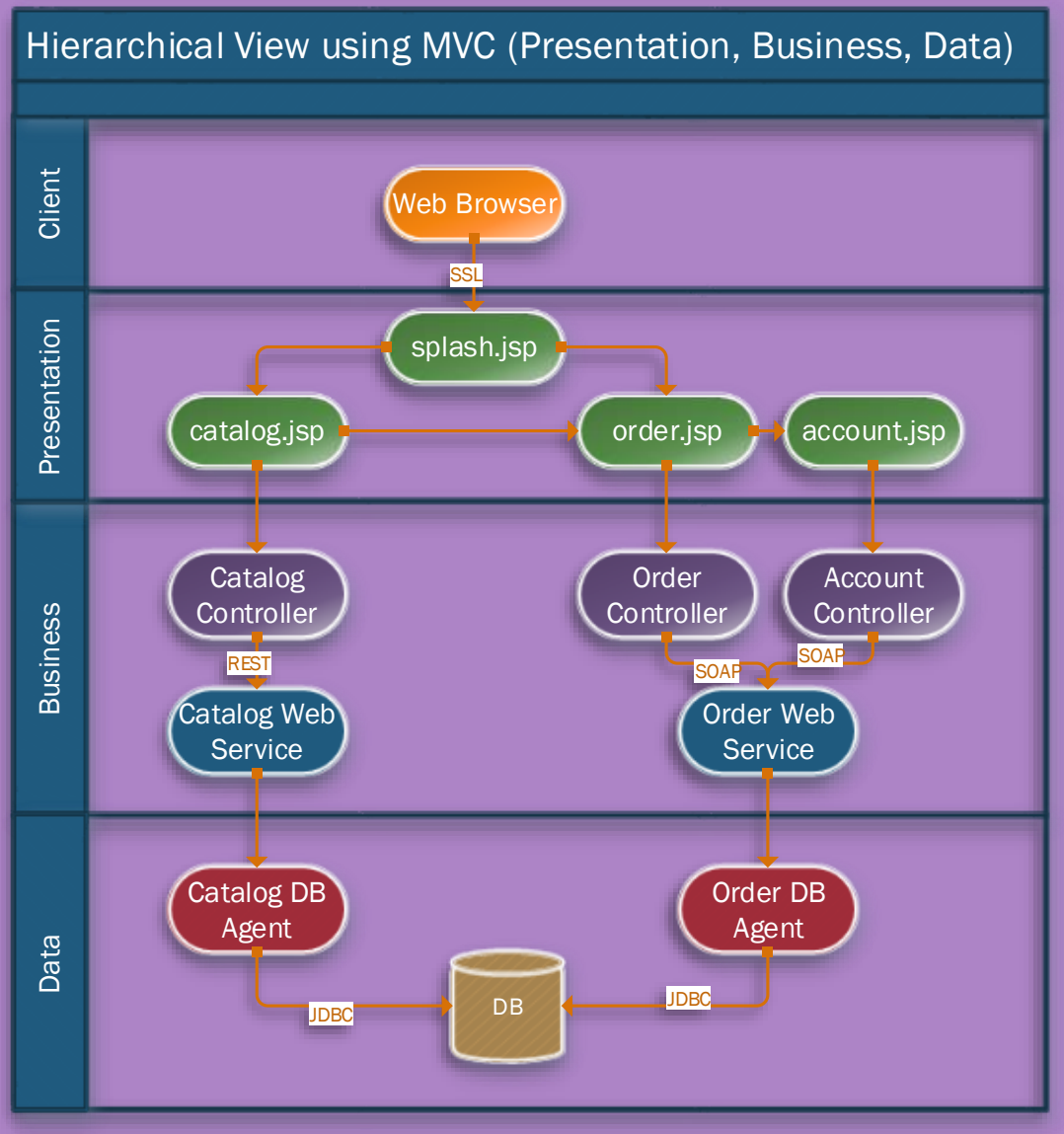
To accomplish a Minimum Viable Product (MVP) prior to enhancements

04

To use best practices in design, development, testing and deployment

Components

- Client Layer
  - SSL Enable Browser
- Presentation Layer
  - 4 JSP files handle all views
- Business Layer
  - 3 Controller servlets decide on actions
  - 2 Web Services (one SOAP, one REST) provide logic and data interface
- Data Layer
  - 2 DB agents abstract and interact with DB
  - Technically, still part of Business Layer but easier to visualize in Data Layer
  - Will use XML query files and Transfer objects

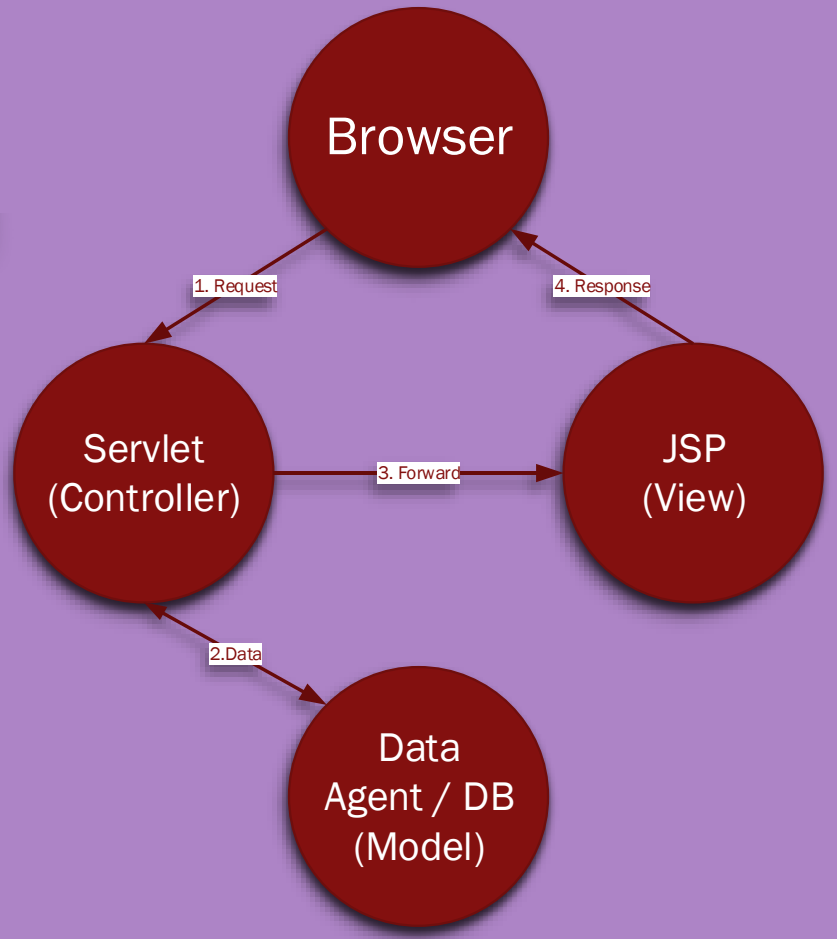


Components

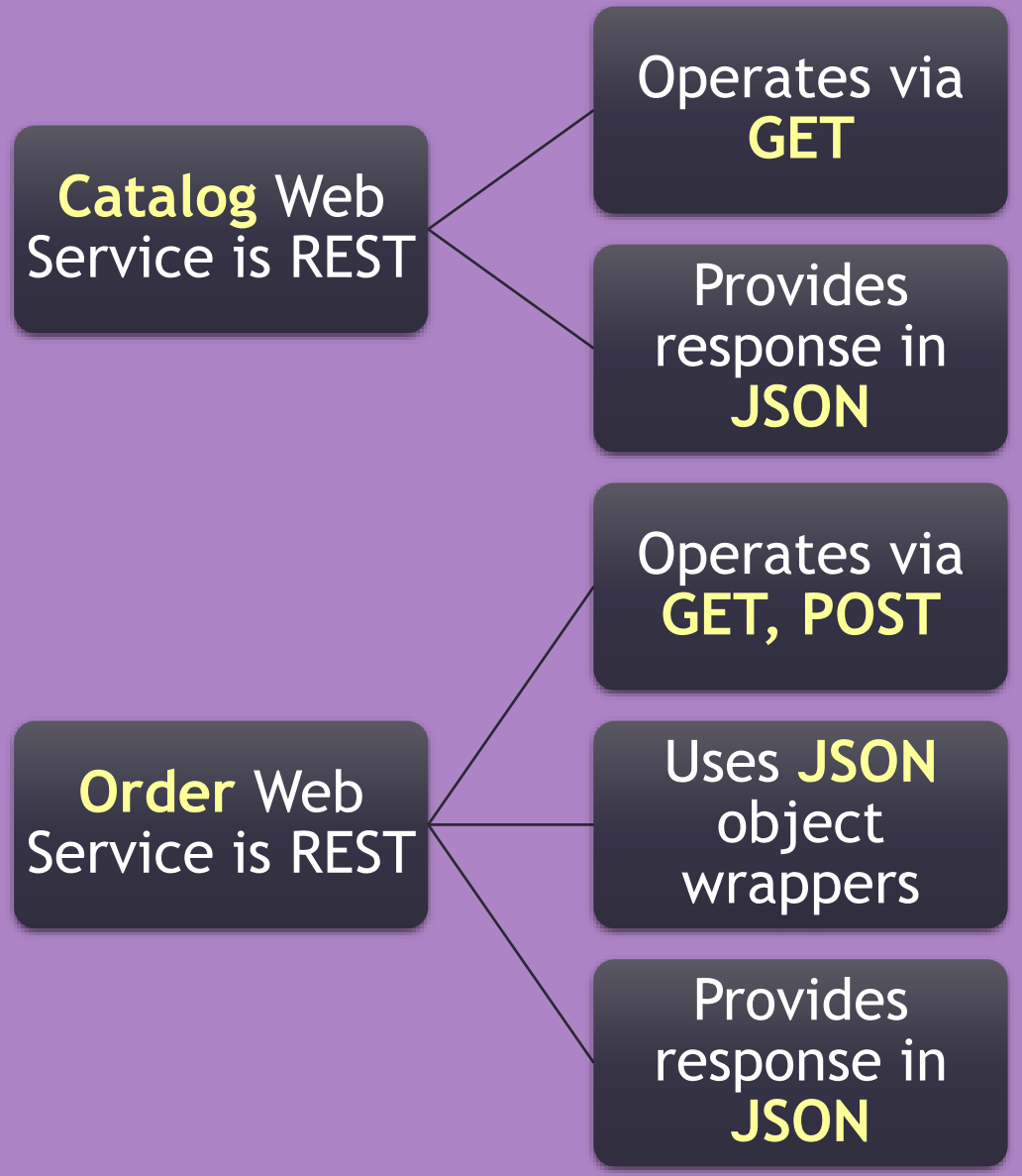
Component	Purpose
Splash	This is the `index.jsp` landing page of the store, no logic; starts the web app UI
Catalog	An MVC 2 pattern that calls a Catalog controller (servlet), interacts with the data layer (REST web service, DB access and Transfer object) and publishes a view (JSP)
Order	An MVC 2 pattern that displays the cart items, total cost, and manages the checkout process using an Order controller (servlet), data layer interface (REST web service, DB agent, and Transfer object) and publishes a view (JSP)
Account	An MVC 2 pattern that handles user login and new user registration using an Account controller (servlet), uses the Order SOAP web service, and publishes a view (JSP) ***[EDIT] The Account controller will be integrated into the Order controller as per spec.***
Schema	A MySQL DB schema and script with table schema and stored procedures for SQL queries, pre-populated with 200 CDs and some dummy users accounts

MVC Model 2 Architecture

- This is a best practice; aka MVC 2
- Browser interacts with a servlet (Controller)
- Servlet does logic and accesses data (Model)
- Servlet then places data in session and forwards to a JSP page (View)
- JSP retrieves data from session and displays
- Chicken or egg: what comes first, JSP or servlet?
  - Often a splash screen (JSP) provides the initial UI; AND/OR a JSP page receives the request and immediately forwards to controller;
  - Alternatively, via servlet filters, we can have a Filter servlet intercept the initial request and redirect it…



Web Services



Security Goals

1

All traffic is encrypted

- Dev env uses a self-signed certificate
- Production env obtained a valid digital certificate from a recognized Certificate Authority

2

Access to the order screen is blocked unless logged in

- Users cannot jump to payment screen directly

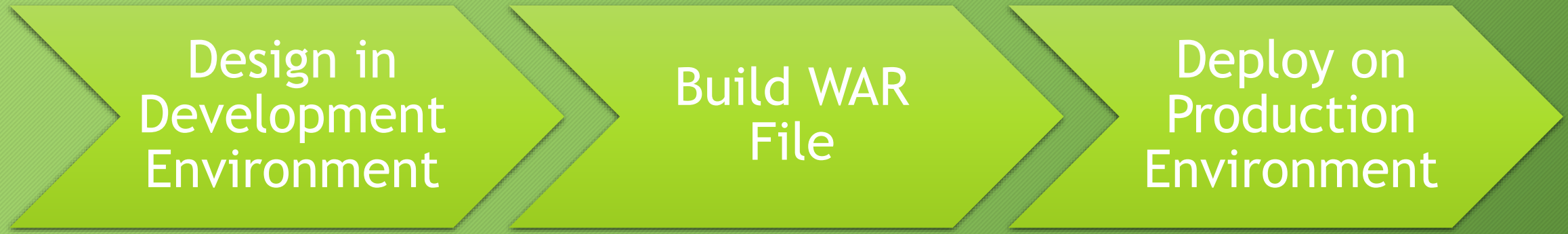
3

No CC information will be stored

- All payment details will be fake for simulation purposes

Deployment

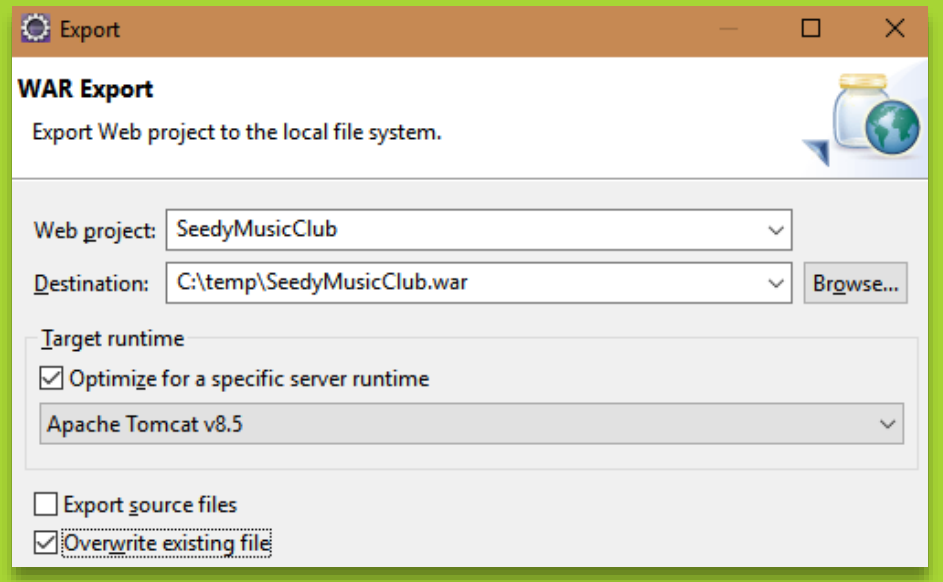
CSI5380 – Team Seedy Music Club – Fall 2017



Deployment Process

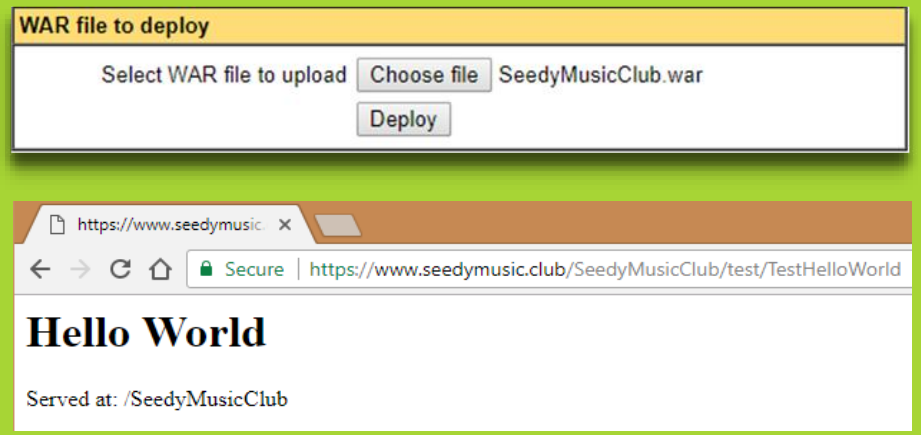
How to Deploy [1/2]?

- Code and test your additions/changes
- Check-in your source to the GIT repository
- Build a **`WAR`** (**W**eb **AR**chive) file:
  - Eclipse: **`File | Export | Web… | WAR`**
  - The Web Project is the name of application
  - Destination is a local directory
  - Optimize it for Tomcat 8.5
  - Do NOT export source file!
  - Select overwrite to avoid annoying alert



How to Deploy [2/2]?

- From browser go to:
  - https://www.seedymusic.club/manager
  - Login with provided credentials (see **Cookbook**)
  - Section "**War file to deploy**"
  - Choose the **WAR** file you exported from Eclipse
  - Hit **Deploy**
- That's it! You're live on the Web:
  - https://www.seedymusic.club/SeedyMusicClub/
  - (URL is based on name of your WAR file)



Further Resources