Exploring Commercial Catch: Creating a Responsive Florida Fisheries Web GIS Using ASP .NET, the Esri JavaScript API 4.x, and Calcite Maps

by

Philipp J. Conner

A Thesis Presented to the
Faculty of the USC Graduate School
University of Southern California
In Partial Fulfillment of the
Requirements for the Degree
Master of Science
(Geographic Information Science and Technology)

May 2018

To my life long marine science lab partner and wife, Lacie

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to acknowledge the outstanding support provided by Dr. Elisabeth Sedano and Dr. Jennifer Swift while working on this thesis. A large thank you to Geographic Solutions Inc. and the Seminole Tribe of Florida for working with me while I strived to complete this Thesis. I would also like to give a special thank you to Dr. Jennifer Gage for not only her help with this project but also her guidance and support throughout my entire graduate program.

List of Abbreviations

ASP	Microsoft Active Server Pages
ASP.NET	ASP Application using Microsoft .NET Framework
CSV	Comma Separated Value
FIPS	Federal Information Processing Standard
FWC	Florida Fish and Wildlife Conservation Commission
GIS	Geographic Information Systems
IDE	Integrated Development Environment
IIS	Internet Information Services (a Microsoft web server application)
MS	Microsoft
MSA	Magnuson-Stevens Fisheries Conservation and Management Act (1976)
MSSQL	Microsoft SQL Server
NMFS	National Marine Fisheries Service
NOAA	National Oceanic at Atmospheric Administration
RDBMS	Relational Database Management System
SDE	Spatial Database Engine
SQL	Structured Query Language
TFS	Team Foundation Server
TTP	Florida Marine Fisheries Trip Ticket Program
USCB	United States Census Bureau

Abstract

The state of Florida has access to vast marine life resources. The Florida marine commercial fishery plays a significant part in the state economy. Hundreds of thousands of commercial fishing trips occur each year totaling hundreds of millions of dollars in dockside value. Because of the weight of the commercial fishery in the state's natural resources, sustainable management is critical. In 1976, the Magnuson-Stevens Marine Fisheries Management and Conservation Act outlined a federal plan for the management of the United States marine resources. The Gulf of Mexico Fisheries Management Council worked with Florida to develop the Florida Marine Fisheries Trip Ticket Program (TTP), to fulfill the mandate to record all licensed commercial landings in the state. TTP landings summaries are available to the public via a web application hosted by the Florida Fish and Wildlife Conservation Commission. These summaries contain a wealth of data in tabular format.

This thesis aims to create a web GIS (The Landings Explorer) of the landings summaries data, to increase accessibility and provide spatial and temporal insight to fishery stakeholders and the general public. By creating widgets and user controls in the ASP.NET Landings Explorer and the Esri JS API v4.x, a user is able to select any species and year from the landings summaries data and render a web map with classes derived from each of the data attributes within these summaries. The application strives to provide fishermen, seafood processors, restaurateurs, and other stakeholders with insights about the commercial fishery to aid in economic and conservatory decision making. By developing the Landings Explorer with the responsive web frameworks, Calcite Maps and Bootstrap, the application seeks to allow users to access this data on any internet connected device from dockside to the kitchen.

Chapter 1 Introduction

The state of Florida has some of the most beautiful and bountiful marine resources in the United States, and their management is critical to the economic and ecologic future of Florida. The Southern Atlantic and the Gulf of Mexico are home to many species and specialized ecosystems that provide a highly diverse fishery. It is because of this unparalleled marine habitat that the commercial fishery plays a significant part in the state economy. One way Florida records the productivity of its fisheries is through recording commercial landings, tallying the pounds of fish caught at time of the vessel offloading. According to the commercial fisheries landing summaries (the total reported poundage brought to shore), more than 380,000 commercial fishing trips occurred in 2015, valued at over $244 million (all values in US dollars) (FWC 2016). Individual catch yields for some species, like Red Grouper (*Epinephelus morio*; Figure 1) or Pink Shrimp (*Farfantepenaeus duorarum*), can easily reach into the millions of pounds annually.

Figure 1. Red grouper adult (*Epinephelus morio*). Photo credit: NOAA

With millions of dollars and pounds of seafood at stake in Florida fisheries, understanding the commercial impact can create informed producers and consumers, and increase awareness of sustainable catch. The goal of this project is to create a web GIS application to display commercial fisheries landings data to a wide range of stakeholders, across multiple platforms. This chapter provides a background of commercial fishing in Florida, as well as a brief description of the intended users of this project and the application development process.

1.1. The Florida Fishery

Florida is not only home to one of the most populous commercial fisheries, but also one of the most diverse. Small, flat-bottomed, "john boats" will prowl the inland waters for red fish (*Sciaenops ocellatus*), while large, steel hull, long liners (Figure 2) steam out of a channel to track swordfish (*Xiphias gladius*) in the Gulf Stream. Some vessels use nets (hand-thrown, seined, or pursed) to catch smaller fish like Shad (*Alosa sapidissima*), Spot (*Leiostomus xanthurus*), or Striped Mullet (*Mugil cephalus*). Spongers call Tarpon Springs home with a unique blend of Greco-American maritime traditions. In the fall, stone crab season starts with a series of festivals along the coast, powered by one of the nation's only inherently sustainable fisheries (a single claw is removed, the crab is returned to the water to regrow the lost claw). Each of these fisheries has an economic impact in the state, and each has real people who depend on these marine resources for a living.

Figure 2. Longline fishing vessels in Port Orange, FL

1.2. Regulatory Background

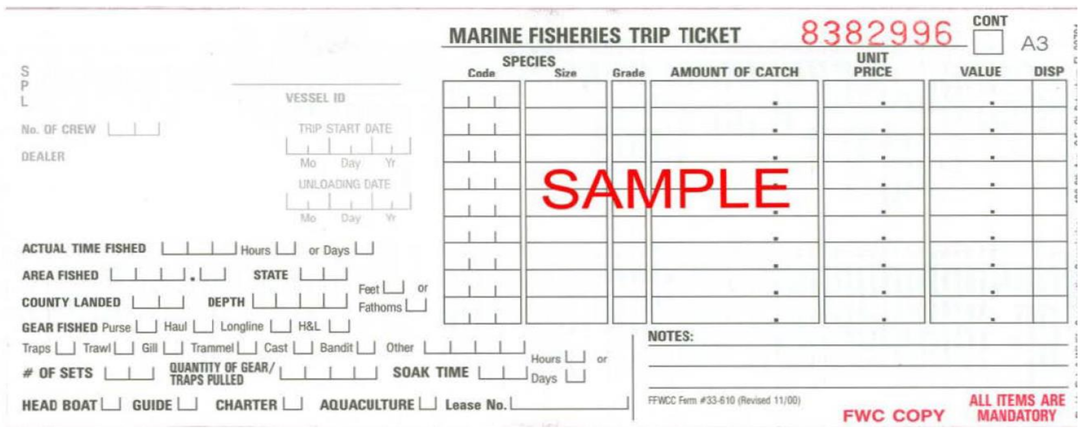
The history of fisheries management is relatively young and can be complex, like most environmental regulations drafted in the 1970s. With multiple stakeholders, including fishermen, seafood processors, retailers, and restaurants, regulation is often a point a contention. This is exacerbated by the fact that both the federal government and state government have a share in the regulation of marine resources. Aside from commercial fisheries regulation, the recreational fishery is also closely monitored and regulated. This subsection seeks to provide context for regulation and management.

1.2.1. Magnuson-Stevens Fisheries Management and Conservation Act

Enacted In 1976, the Magnuson-Stevens Fisheries Management and Conservation Act (MSA), outlines a federal framework for the management of US marine resources. The MSA mandates that regional councils oversee the conservation and regulation of the fisheries with industry partners and management professionals. This management seeks to maintain sustainable commercial fishing by developing stock assessments and providing scientifically driven catch limits (U.S. Congress 1976, 2007). These regional councils are often comprised of multiple states and coasts. Florida is no exception, with the Gulf and Southern Atlantic Councils working with Florida to develop the Florida Marine Fisheries Trip Ticket Program (TTP). The Florida Fish and Wildlife Conservation Commission (FWC) is responsible for collecting TTP records and creating federal reports for the regional councils and National Marine Fisheries Service (NMFS) of the National Oceanic and Atmospheric Administration (NOAA). In part, the TTP was created in order to fulfill the stock assessment mandate. While the TTP summaries provide the greatest overview of collected fisheries data, other data contributes to the stock assessment. This additional data includes fisheries biologists doing dock-side sampling, research cruises, and contracted fishing, to name a few.

1.2.2. Florida Trip Ticket Program

Under the TTP, at the time of offload (landing), commercial fishermen are required to complete a triplicate paper trip ticket (Figure 3). Traditionally the landing procedure is as follows. The catch is weighed by individual species and is totaled. The seafood buyer, usually a seafood processor (“fish house”) or wholesaler, then decides on a price, often informed by the supply and demand of the species, and the going rate for the species. At the time of transaction, the fisher and the buyer fill out the triplicate trip ticket in tandem, listing the species poundage, time and gear fished, fishing area, landing location, and other fishing method data. After completion, one slip is sent to FWC, and the other two to be retained by the captain (or vessel owner if the captain is hired) and buyer, respectively. However, as more fish houses enter the high-speed internet age, an electronic solution was added to allow the faster receipt of the landings information. Currently, a desktop application, VESL, created by Bluefin Data manages electronic reporting to state and federal agencies.



MARINE FISHERIES TRIP TICKET 8382996 CONT A3

S
P
L

VESSEL ID

No. OF CREW

TRIP START DATE Mo Day Yr

DEALER

UNLOADING DATE Mo Day Yr

SPECIES Code	Size	Grade	AMOUNT OF CATCH	UNIT PRICE	VALUE	DISP
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	

SAMPLE

ACTUAL TIME FISHED Hours or Days

AREA FISHED . STATE

COUNTY LANDED DEPTH Feet or Fathoms

GEAR FISHED Purse Haul Longline H&L

Traps Trawl Gill Trammel Cast Bandit Other

OF SETS QUANTITY OF GEAR/ TRAPS PULLED SOAK TIME Hours or Days

HEAD BOAT GUIDE CHARTER AQUACULTURE Lease No.

NOTES:

FFWCC Form #33-610 (Revised 11/00)

FWC COPY

ALL ITEMS ARE MANDATORY

Figure 3. A Florida Trip Ticket, Image Credit: FWC

FWC offices receive the paper trip tickets monthly in batch alongside the data submitted in the VESL DB. The trip tickets are tallied, and the data directly informs the population assessment, which in turn impacts fisheries management through catch limits and moratoriums. Once are certain poundage is met, the fishery may close completely prohibiting catch, or management plans may be revisited.

The trip tickets contain the names of the owner and captain, the area fished, the price paid, and the value of the catch. Due to the sensitive nature of the data, the TTP data set is confidential. Some fishermen resent the data collection and management, citing it as a violation of privacy. This is understandable as the trip ticket has financial information and geographic information. While individual trip tickets are not publicly available, the trip ticket summary data is available to the public and can be downloaded in a CSV via the FWC website (Table 1).

Table 1. Sample Trip Ticket summary data

Year	County	Species Name	Pounds	Trips	Avg. Price per Pound (USD)	Est. Total Value (USD)
2010	PINELLAS	SNAPPER, RED	185,846	526	$3.54	$657,130
2010	PINELLAS	SNAPPER, SILK	15,494	38	$3.12	$48,347
2010	PINELLAS	SNAPPER, VERMILION	18,711	248	$1.96	$36,649
2010	PINELLAS	SNAPPER, YELLOWTAIL	212	17	$1.73	$367
2010	PINELLAS	SPOT	437	18	$1.21	$531
2010	PINELLAS	SWORDFISH	5,252	5	$3.17	$16,667
2010	PINELLAS	TILAPIA (NILE PERCH)	37,664	163	$0.58	$21,980
2010	PINELLAS	TILEFISH (GOLDEN)	12,233	23	$2.05	$25,133
2010	PINELLAS	TILEFISH, BLUELINE (GRAY)	4,001	9	$0.91	$3,626
2010	PINELLAS	TRIGGERFISH	2,393	155	$1.01	$2,427
2010	PINELLAS	TUNA, BLACKFIN	582	12	$0.68	$397
2010	PINELLAS	TUNNY, LITTLE (BONITO)	1,255	30	$0.63	$791
2010	PINELLAS	WAHOO	688	15	$1.8	$1,238
2010	PINELLAS	CONCH (WHELK, HELMET)	312	2	$0.15	$47
2010	PINELLAS	CRAB, BLUE (HARD)	307,661	2,131	$1.58	$485,972
2010	PINELLAS	CRAB, BLUE (SOFT)	79	5	$9.14	$718
2010	PINELLAS	CRAB, STONE, JUMBO CLAWS	17,345	663	$12.18	$211,352
2010	PINELLAS	CRAB, STONE, LARGE CLAWS	76,912	1,771	$11.04	$849,341
2010	PINELLAS	CRAB, STONE, MEDIUM CLAWS	75,889	1,805	$6.82	$517,819

Source: Florida Commercial Fisheries Landing Summaries

1.3. Motivation

After spending time with many parties within the commercial fishery in Florida, the author believes that a way to intuitively display fisheries landings summaries should be available. Some fishermen are not aware that these summary data exist, despite the very relevant economic insights the data provides. The same can be said for the other interested parties: seafood wholesalers and retailers, restaurants, and tourism interests. The federally managed fisheries are always a hot topic dockside, considering that the economy of the fish itself passes from the fisherman to the fish house, and then to the public via direct retail, or by restaurant preparation. Each party has a valid stake in the management of the fishery, not only in its goal of ecological sustainability but the fisheries' direct economic impact. A dime less per pound of fish can be significant when wholesaling thousands of pounds. Changes in the management of a specific fishery can cause menu mainstay species to become rare, high ticket dishes. The Landings Explorer web GIS application was created to aid in these kinds of situations.

The FWC website does provide the data, which means the requirement that the public has access to these data has been fulfilled, but not the same regarding access to the insights *within* these data. It is important that we provide a means to consume this data, both for economic and ecological reasons. With the creation of the Landings Explorer, this project hopes to provide the tools for enlightened decision making for stakeholders and expand education for the general public.

1.4. Application Description

The Landings Explorer web GIS application allows users to dive into the complete TTP landings summary dataset. The user is presented with a responsive ASP.NET (Microsoft Active Server Pages with the .NET libraries) application. JavaScript user controls via the Esri JavaScript

(JS) API allow the user to alter the map in many ways. The user can interact with the map to look at changes to catch over time, to see how catch has increased or decreased. This can allow insight into impacts caused by events such as the Deepwater Horizon disaster, or a major regulatory change. The user can also view the map from each species landed since the start of the TTP, seeing how catch varies from species to species. There are also the inherent geographic insights that come with a web GIS. A user can view regional fishery strength or the abundance of a species with regards to an abiotic factor gradient, for example.

1.4.1. Intended User Groups

The commercial fishery has many different stakeholders that could benefit from the Landings Explorer. The commercial fisher and wholesalers could both glean valuable economic data from the application. The fisher could easily see where they could get the best price for their fish or the wholesalers could find where they best profit could be made by viewing the same prices. Those within the service and tourism industries can also benefit from the landings data, as the rarity of specific species and price fluctuations can drive demand, and affect menu pricing. The seafood consumers are also a targeted user group as they drive consumer demand for fisheries products. For example, if one wanted to purchase spiny lobster, they may be able to find it not only fresher but also cheaper in the southern Florida counties as opposed to where it would need to be shipped to the panhandle. The application can also make certain regional products easier to find, such as oysters in Apalachicola as opposed to trying to find them in Miami.

1.4.2. Study Area

The study area for this thesis is the state of Florida with its counties as the primary division (Figure 4). Florida was chosen because of its large marine fishery, and the readily available commercial landings data. The landings summaries are described by county, therefore county municipal boundaries provided by the US Census Bureau (USCB) are the primary divisions of the study area. Statewide data is available at monthly time intervals but was not used because of the lack of area divisions. However, including state data is considered for future versions.

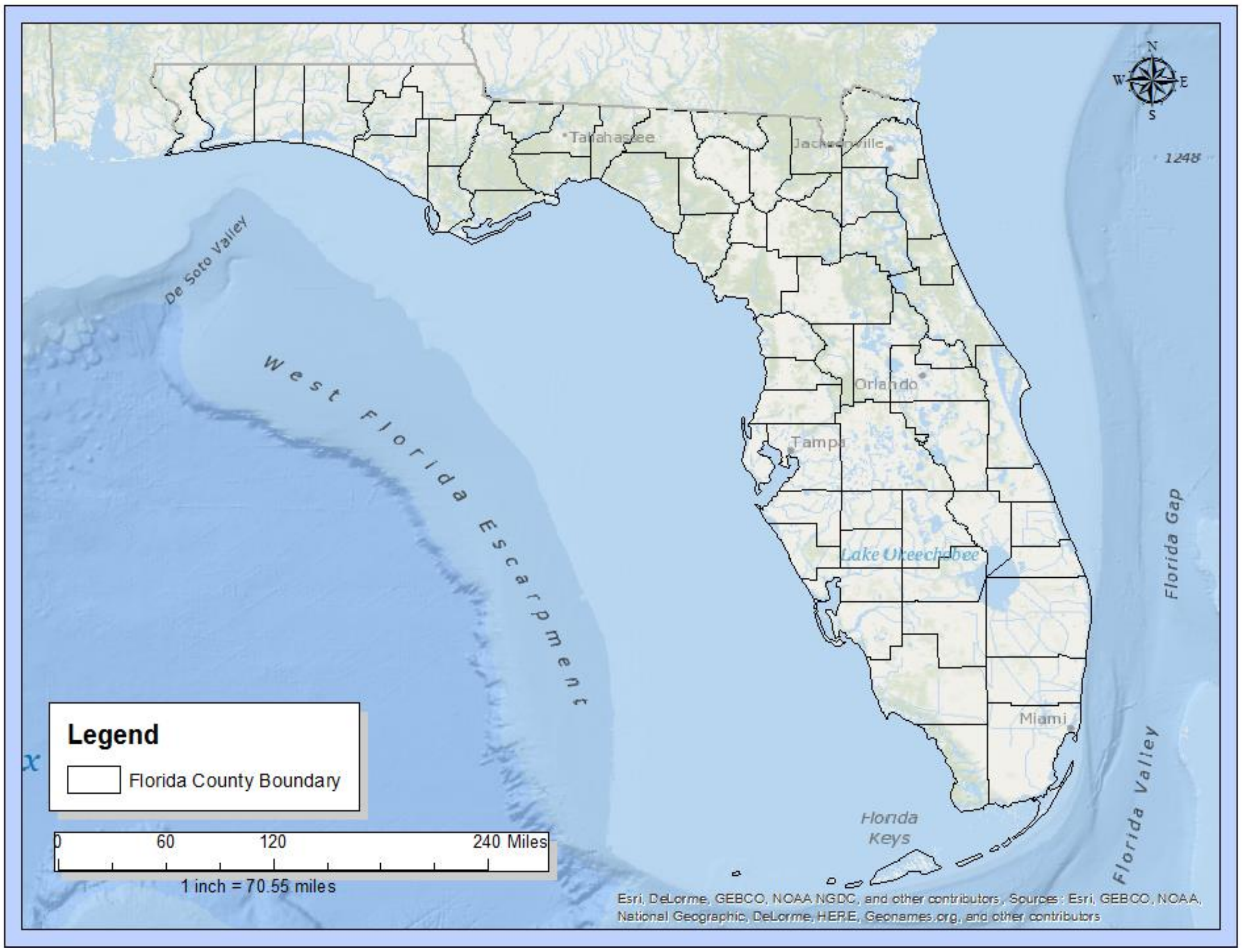


Figure 4. The study area: The state of Florida

1.4.3. Example Use Cases

The economics of a trip can be complex, including fuel, ice, bait, gear, provisions, and other expenses. Most trips begin thousands of dollars in debt before catching a single fish. Therefore, finding a good price for your catch can be the difference between turning a profit or not. Often fishers will call their regular fish house and see what the dock rate is for their catch that day. If they do not like the price, they can take their catch to another fish house for a better price. The maps in the Landings Explorer can be used to see how the paid price fluctuates seasonally or spatially. A captain could quickly identify locations that may be having a shortage of a certain species and choose to offload there to get the most out of the demand. Inversely, if the fish house is short of a specific species, they may be able to determine where the bulk of the catch is and call other fish houses in that area, if not the vessels themselves.

Seafood consumers and the service industry could use the application for different means. Ecological and sustainability considerations are coming to the fore as understanding about fisheries management grows. Many restaurants in the Tampa Bay area have had to change signature dishes due to weak local catches or exorbitant price increases. A common phrase on seafood menus is "Grouper's Cousin." This, often tilapia or other similar species has displaced groupers in local fare due to rising costs and management impacts. The consumer or chef could see how ecological or management events drive price over time. Landings Explorer can also help these people find more sustainable alternatives to fish already being consumed.

1.5. Thesis Structure

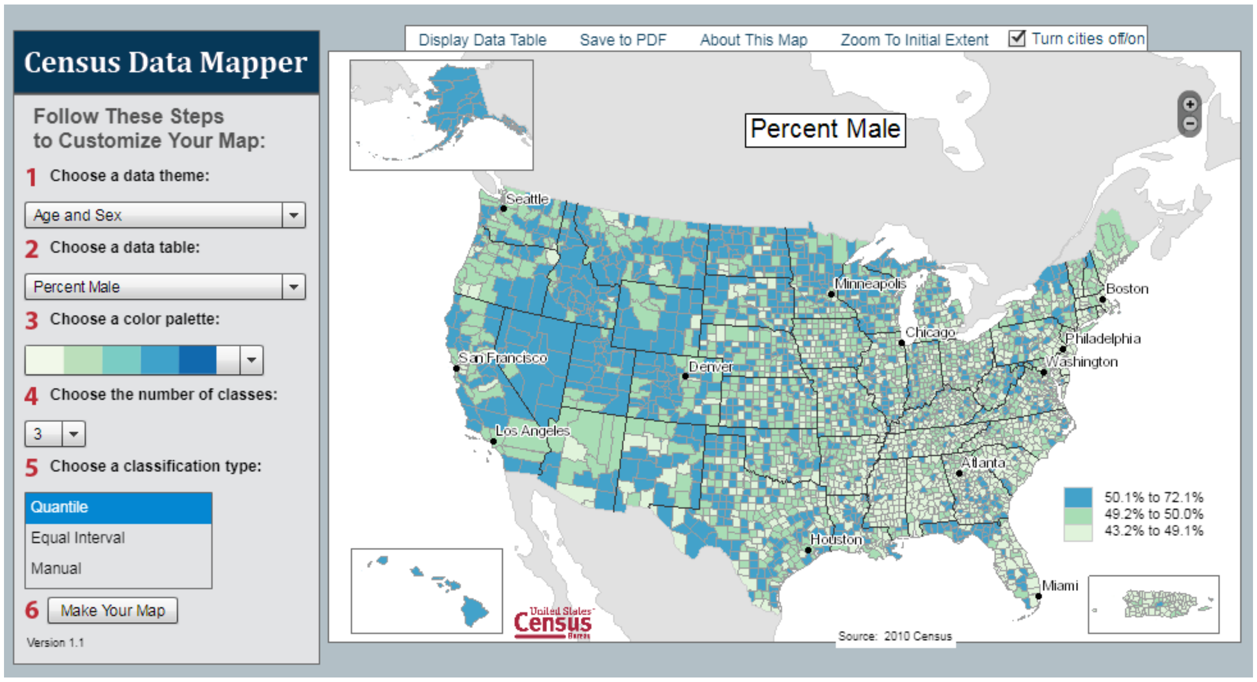
The rest of this thesis is constructed from individual sections touching on different aspects of the application, its conceptualization, and its development. In Chapter 2, relevant literature is examined with parallels and inspiration being explored. Chapter 3 discusses the application requirements, going into detail about the design and use cases of the application. Development of the application is outlined in Chapter 4, with specifics about data sources, architecture, and coding patterns. Chapter 5 displays the final application and discusses each control in the application. Lastly, Chapter 6 discusses the conclusions of the project as well as future development, hurdles, and a retrospective on design and development choices.

Chapter 2 Related Work

Utilizing a web GIS to display data for consumption is a relatively new idea in the lifetime of GIS. A web GIS uses websites and other technologies to create an internet connected GIS. Often times this consists of a client server interaction. Some early GIS web apps relied on now deprecated technologies, while others are still in production today. With legacy technologies falling out of favor and new technologies being developed constantly, the variety of web GIS applications continues to grow. Just as web GIS technologies vary, so do their uses. Some focus on educating a specific user group like drone pilots or school children (Blee 2016; Burrows et al. 2013; Kim et al. 2013), while others aid in decision making (Dowling 2014; Karnatak 2007; Piarsa et al. 2012). Even a cursory glance at the work in this field shows that there are many ways to solve many different problems, from leveraging existing enterprise packages to “ground up” development (Dragićević 2004; Zavala-Romero et al. 2014). This thesis will provide a broad, but detailed view of the literature related to this web GIS project in three distinct sections: Web GIS Applications, Fisheries Management, and Application Insights.

2.1. Web GIS

Since the first web GIS was created by the Palo Alto Reach Center (PARC) in 1993, we have seen an explosion of GIS applications moving to the Web (Kraak 2004; Fu and Sun 2011). From county GIS portals to the Census Data Mapper (Figure 5), web mapping is quick becoming ubiquitous in the GIS world. Everyday uses of the web are now featuring some form of GIS. Shopping for a house (Dowling 2014), booking a hotel room, finding a local restaurant, and even finding a date for Saturday night utilize the “science of where.” However, making things easy to find or navigating on your vacation are just a couple of examples of the benefits provided by web GIS.


Census Data Mapper

Follow These Steps to Customize Your Map:

1 Choose a data theme: Age and Sex

2 Choose a data table: Percent Male

3 Choose a color palette:

4 Choose the number of classes: 3

5 Choose a classification type: Quantile

6 Make Your Map

Version 1.1

Figure 5. The US Census Data Mapper

There are many benefits to creating a GIS on the web. A web GIS application can be deployed once, reach a broad public outside of the project's locale, and can even be viewed on a smart phone (Kraak 2004; Fu 2015). Simultaneous collaboration can be done with users with a web GIS. Data sources can be kept up to date and can be accessed by disparate software and languages. A web GIS data service can be written in a specific language on a specific operating system (C# on Windows for example) and accessed by another application on a different platform and language (Java on an Android device) (Fu and Sun 2011). A relatively recent use of web GIS is in volunteer GIS (VGIS) (Fu and Sun 2011). VGIS combined with the web can allow thousands of users to contribute GIS data freely in ways never before imagined. Professionally, field surveyors can record data and upload to the home office on the fly via a cellular network without having to return home to copy and process their data. As technology moves forward, new capabilities will continue to be developed and implemented.

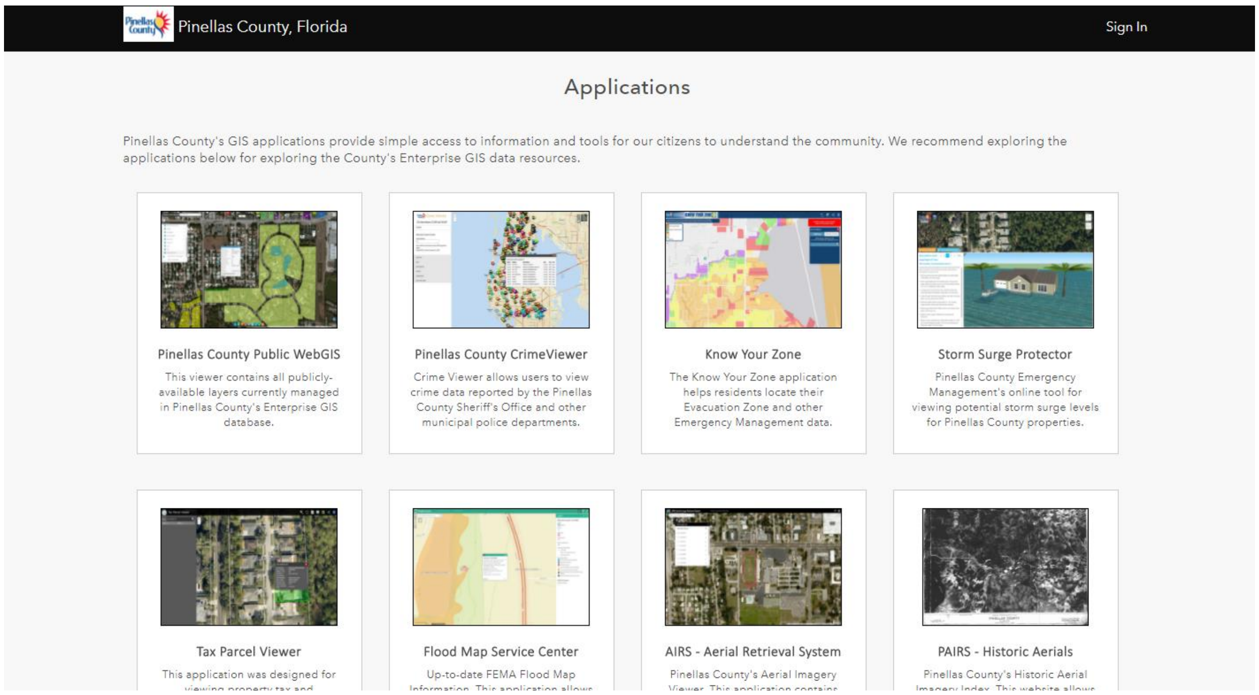
Web GIS applications come in many different forms and can utilize many different technologies. Some rely on large, proprietary software suites, while others choose open source platform solutions. With many options available for the web developer, finding a web GIS platform that fits the project needs is also easier than ever. Perhaps one of the most obvious platforms to the average user is Google Maps.

Not only is Google Maps popular with the average user, but many developers have also chosen this platform for web development (e.g., Blee 2016). For his Master's Thesis, Blee developed a web GIS powered by Google's API to display drone regulatory areas with a MySQL backend. Kamel Boulos (2005) used the same API to develop a web GIS that displayed strategic heath authorities for the web user. In 2011, Kamel Boulos went on to even develop touchless interfaces for Google Earth, adding even more depth to the Google Platform (Kamel Boulos 2011).

Another popular platform is the Esri ArcGIS API and ArcGIS Server (Berry et al. 2011; Piarsa et al. 2012; Huang, Du, and He 2013; Norjou and Thomas 2016). The Esri platform provides a front to back enterprise web solution, paring Microsoft or Oracle database technologies with an Esri GIS server, managing multiple applications with a web GIS portal. Berry et al. (2001) developed an Esri powered web GIS that combined government interaction, VGIS, and 3D technology to drive community engagement in the planning process for wind farm development. Others opt for open source solutions where licensing is a limiting factor or access to the native source is required (Zavala-Romero 2014). Web GIS is now even available across many diverse hardware platforms, and as with software, there are many choices with benefits and shortcoming for each.

Developers can use Java to unify applications, not only across multiple platforms but multiple hardware configurations. Your car can report traffic or emergency data to a Linux server hosting Apache Tomcat, to be consumed by a mobile device running iOS. If a large corporation has the Esri ArcMap suite already in their enterprise application stack, other Esri software can be paired to streamline the workflow, minimizing effort in bringing data to a production web environment. A startup with limited funding could go completely open source and pair a QGIS desktop workflow with the QGIS server or a standalone WMS server to minimize costs (Fu and Sun 2011; Zavala-Romero 2014). If one does not have the hardware or infrastructure to host a web GIS oneself, one could always use one of the recent cloud platforms like AWS (Norjou and Thomas 2016). Cloud solutions allow users to move their entire GIS workflow to the web, from desktop authoring to python geoprocessing (Fu and Sun 2011; Norjou and Thomas 2016). Another option is an enterprise GeoPortal. A geoportal is a website that acts as an interface for web GIS applications, GIS web services, and other forms of GIS data. The Esri GeoPortal bundled with ArcGIS Enterprise can handle user access, web application development, interfacing with an ArcGIS server, and more. The one thing all these solutions have in common is digital cartography on the web. These solutions are not just limited to commerce and enterprise development, though.

Perhaps one of the most useful applications of web GIS is in eGovernment. eGovernment is the modern movement from traditional paper means of government interactivity to building a web presence to facilitate interaction (Fu and Sun 2011). This could be from renewing your driver's license on a Department of Motor Vehicles website to filing your taxes on line. GIS powered eGovernment is relatively new, but continuously growing as the government is valuable a source of information for all its constituents. In Pinellas County, Florida, this is especially true. With growing concerns about climate change and increasingly intense weather events, The Pinellas County Geo Portal (Figure 6) provides information about flood zones, hurricane tracking and evacuation routes, and storm surge modeling.



Applications

Pinellas County's GIS applications provide simple access to information and tools for our citizens to understand the community. We recommend exploring the applications below for exploring the County's Enterprise GIS data resources.

Pinellas County Public WebGIS
This viewer contains all publicly-available layers currently managed in Pinellas County's Enterprise GIS database.

Pinellas County CrimeViewer
Crime Viewer allows users to view crime data reported by the Pinellas County Sheriff's Office and other municipal police departments.

Know Your Zone
The Know Your Zone application helps residents locate their Evacuation Zone and other Emergency Management data.

Storm Surge Protector
Pinellas County Emergency Management's online tool for viewing potential storm surge levels for Pinellas County properties.

Tax Parcel Viewer
This application was designed for viewing property tax and

Flood Map Service Center
Up-to-date FEMA Flood Map information. This application allows

AIRS - Aerial Retrieval System
Pinellas County's Aerial Imagery Viewer. This application contains

PAIRS - Historic Aerials
Pinellas County's Historic Aerial Imagery Index. This website allows

Figure 6. The Pinellas County GeoProtal

Beyond environmental concerns, an eGovernment portal provides other vital services to the taxpayers. The Pinellas portal provides county residents with crime maps, administrative boundaries, parcel data sets, and aerial imagery; even a means to provide feedback on future development (Berry, 2011). Developing a government geoportal is of great personal interest to the author. Aerials can tell a story and highlight the history of an area. They can also show how a road construction project is progressing. Parcel data can aid in neighborhood development, and administrative boundaries can show which police force holds jursidiction where. eGovermnet in conjuction with GIS portals have great value. However, the value of GIS in eGovernment does not end here. Incorporating easily consumable fisheries data in an already existing eGovernment platform is one of the major goals of this project. GIS services continue to be integrated into eGovernment and will likely be for the foreseeable future.

2.2. Fisheries GIS and the Web

Existing fisheries applications are primarily the domain of regulatory agencies that have a mandate to provide data services to the public. These applications focus on displaying maps related to the fisheries, however the author has yet to find one that shares the fisheries landings themselves. The Gulf of Mexico Fishery Management Council (GoMFMC) hosts a geoportal. The GoMFMC geo portal hosts multiple individual mapping applications like the Fish Habitat Mapper (Figure 7).

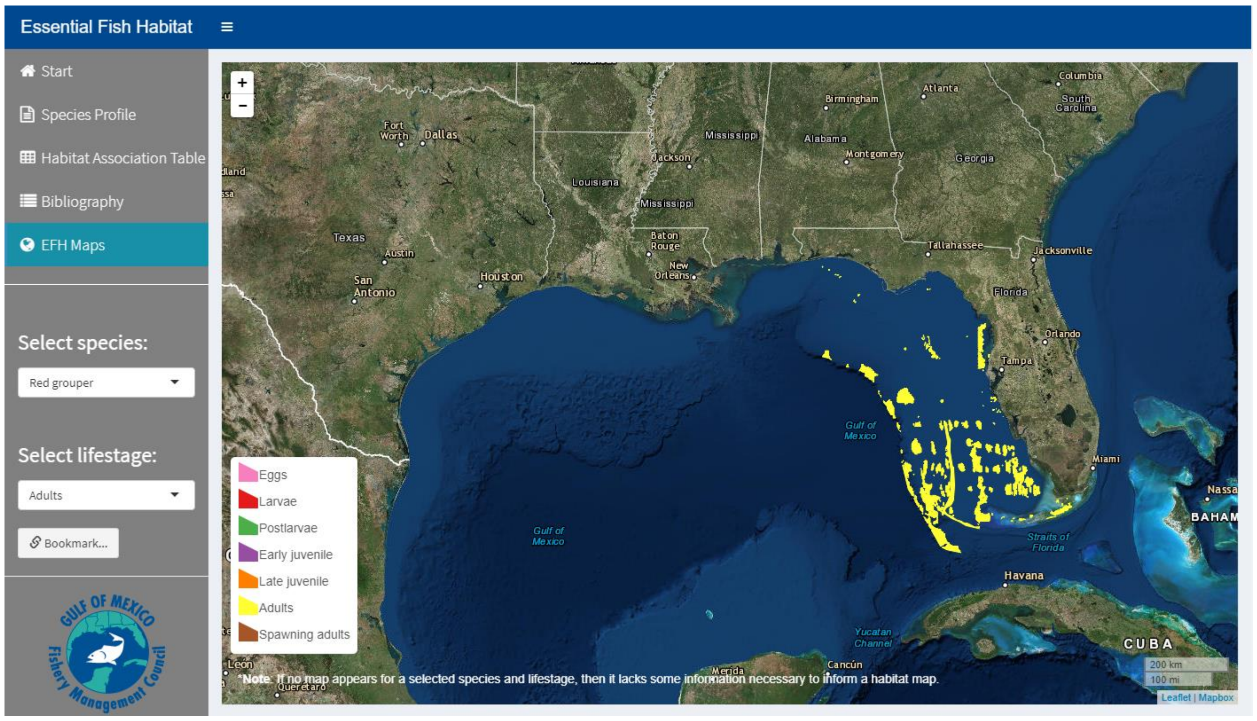
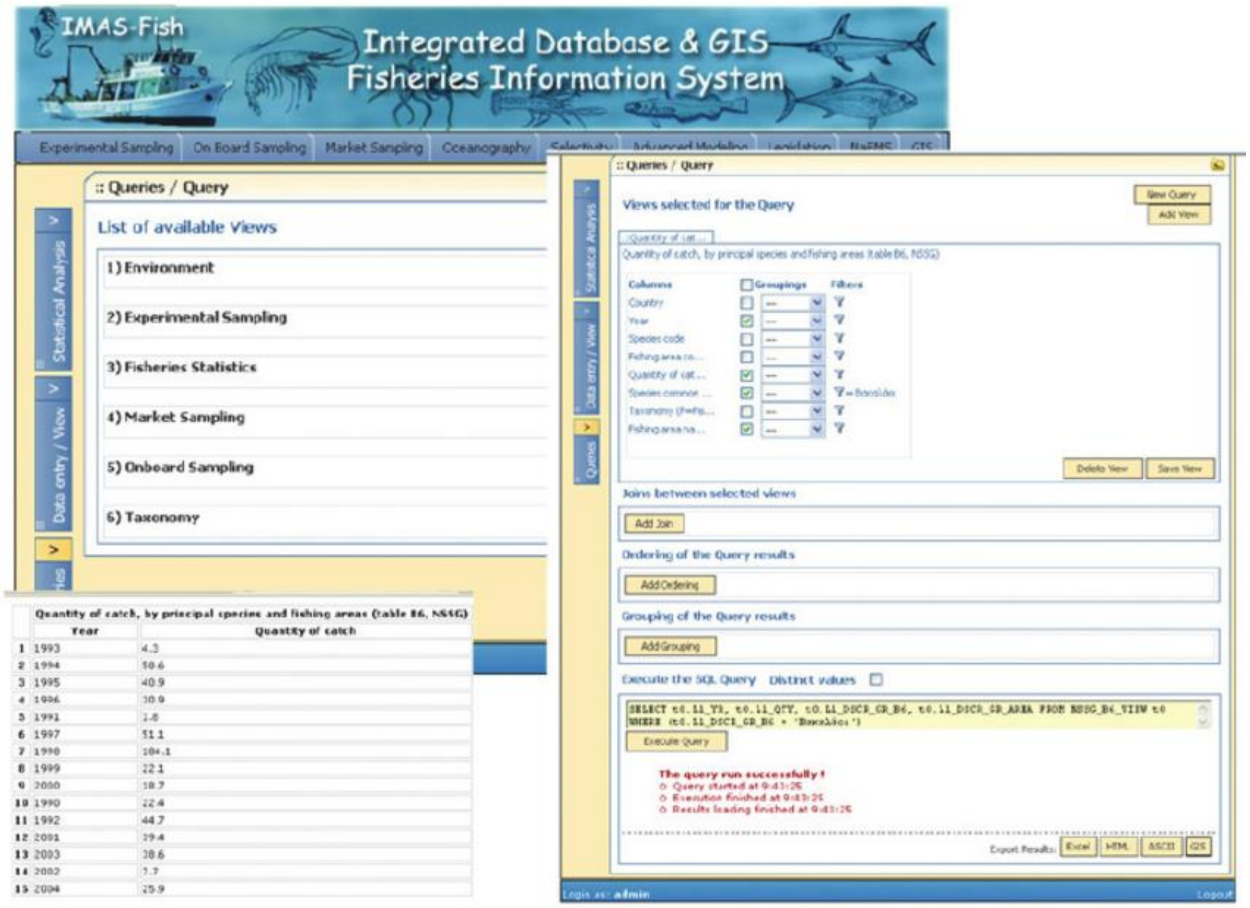


Figure 7. Essential Fish Habitat Map Application by the Gulf of Mexico Fishery Management Council

These applications display ecological data, highlighting areas of sensitivity of different species. A future goal of this research is to incorporate this application within a similar geoportal or ASP application, as visualizing multiple datasets on the same map can provide even more insight. Other fisheries web GIS focus on a system for management professionals to enter, run models and processes, and retrieve GIS Data. Kavadas et al. developed the IMAS-Fish application (Figure 8) (Kavardas et al. 2013). The IMAS-Fish application was created for Hellenic Centre for Marine Research to manage interdisciplinary fisheries data via a relational database management system (RDBMS) combined with a GIS data portal. A relational database consists of multiple tables joined together by a related field key. While this application focuses on aiding the management professional, the Landings Explorer operates at a more modular level.



Quantity of catch, by principal species and fishing areas (table B6, NSSG)

Columns	Groupings	Filters
Country	☐	
Year	✓	
Species code	☐	
Fishing area co...	☐	
Quantity of cat...	✓	
Species common ...	✓	= Βακαλάος
Taxonomy (Fam...	☐	
Fishing area na...	✓	

```
SELECT t0.11_YR, t0.11_QTY, t0.11_DSCR_GR_B6, t0.11_DSCR_GR_AREA FROM NSSG_B6_VIEW t0
WHERE (t0.11_DSCR_GR_B6 = 'Βακαλάος')
```

The query run successfully !
- Query started at 9:43:25
- Execution finished at 9:43:25
- Results loading finished at 9:43:25

Quantity of catch, by principal species and fishing areas (table B6, NSSG)

	Year	Quantity of catch
1	1993	4.3
2	1994	50.6
3	1995	40.9
4	1996	30.9
5	1991	1.8
6	1997	51.1
7	1998	104.1
8	1999	22.1
9	2000	18.7
10	1990	22.4
11	1992	44.7
12	2001	29.4
13	2003	38.6
14	2002	7.7
15	2004	25.9

Figure 8. IMAS- Fish Application

2.3. Application Insight

2.3.1. Data Patterns

There are two main data patterns in modern web GIS. The first is to consume GIS data inherent in the web application. This is exemplified by developing a web GIS with Google technologies. The data is stored in Keyhole Markup Language (KML). This eXtensible Markup Language (XML)-like dataset often exists in a file along site the site itself, though this can also be consumed as a service. This leads to the second data pattern, consuming data through a Representational State Transfer (REST) service. This provides the data on a secondary or tertiary computer system that can act in an inter-operational way, meaning the data is served, decoupling the data source from the web site host (Fu and Sun, 2011). In the case of ArcGIS server, these

REST services can even be supplied through a direct RDBMS query. The data can be hosted in SQL for example, then queried by ArcGIS server, before offering the service to a browser on a third client machine. Many RDBMS products are currently available. Some of the most common options are Microsoft SQL (MSSQL) Server (e.g., Berry et al. 2011 and Dowling 2014), Oracle (e.g., Kavadas et al. 2013), PostgreSQL, and MySQL. However, some GIS suites are limited to the type of database that can be used. Esri's ArcMap has significant support of SQL Server and Oracle RDBMSs, but limited support for MySQL, for example.

2.3.2. Application Patterns

Application patterns can vary depending on users, use cases, and the data required. Some web GIS applications require the ability to create new features (Kavadas et al. 2013). Other are deployed to display data about a specific topic, and the user takes on the role of consumer (Blee 2016; Burrows et al. 2013). The application may not even require a local RDBMS, but rather relies on consuming *external* web GIS services (Djamaluddin 2015). An application may be a VGIS, collecting user-submitted data, and displaying the results as a community service, meaning that the focus is on layer writability and QA/QC, paired closely with editable features in a database. Others rely on web-hosted geoprocessors (Dragićević 2003; Huang, Du, and He 2013; Fu 2015).

Chapter 3 Requirements

This project seeks to provide an engaging, easy-to-use experience for fishery stakeholders to view data from the Trip Ticket Commercial Fishery Landings Summaries. To ensure that this overarching goal is met, multiple requirements have been chosen to help drive development and user experience of the Landings Explorer application. In this chapter, four sections are provided. The first section describes the overarching goals of the application. The second outlines the user requirements for the application. User requirements lay out how the user will interact with the temporal, geographic, and economic information of the data and features. They are enumerated and tested via use cases. The third section describes the technological requirements. These are points of determination for design strategies and development methodology, and include specific coding, architectural, and database system considerations. The fourth section outlines the specific use cases in the project. Lastly the hardware and software utilized are discussed.

3.1. Application Goals

The primary goal of the application is to allow the fisheries stakeholders to glean insight from the landings data set in an easy-to-use web GIS, accessible from any internet connected device regardless of physical platform (device) or operating system. Specifically, the commercial fishing industry stakeholders are the fishermen, the seafood wholesaler or processor, the retail seafood establishment owner, the restaurateur, and the seafood consumer. Each of these shareholders can find value in different information, but economic information is of primacy to all. It is with this in mind that the following requirements were identified.

A secondary goal of the application is to develop the application in such a way that it if desired, could be added seamlessly to an Esri Enterprise GIS environment within a Microsoft ASP.NET application. Many government agencies use an Esri Enterprise environment, with their web pages being developed in ASP.NET. The technological requirements were chosen with this goal in mind, making the Landings Explorer application modular.

3.2. User Requirements

The first requirement is that the user, regardless of the type of stakeholder, should be able to control which map is rendered by selecting a species and year combination. The data spans over 20 years and across more than 120 individual commercially regulated species, and the user should be able to display any combination of these. Secondly, the user should be able to click on a feature in the map (a county in this case) and display each field in the landings summary dataset. The third requirement is that the user should be able select the data attribute within the landings summaries to be visualized in the map. This means that different color schemes should be selectable for total pounds caught, landing price per pound, number of trips, and total estimated value of the catch.

3.3. Technological Requirements

The first technological requirement is that the Landings Explorer application be developed as a ASP.NET web application, so that it may be modular for potential future integration with existing server applications. ASP is a Microsoft server application that permits the execution of server-side code. The .Net part of this technology is often a part of modern ASP applications due to Microsoft interoperability and greater features. ASP.NET is discussed in greater depth in Chapter 4. Currently, both the NOAA and the FWCC main websites are ASP.NET applications. It is because of this that the ASP.NET application was chosen as the server application.

The second requirement is that the application must consume REST services and that no data be stored locally. This is to mimic a true web deployment in a high security high isolation production environment. Third, the REST service must host the landings data and the county layer. The application should utilize a table service from the ArcGIS server that was generated directly from the SQL deployment. This requirement was chosen as it most closely matches the data stream in a larger enterprise ASP application. It was also chosen because future development could utilize direct SQL query services.

Lastly the application must be developed using Responsive design principles. Responsive design is a way of building web applications that permits the web page to dynamically shift to meet the dimensions of the screen being displayed. Using CSS classes and a responsive framework like Bootstrap, the website reads the screen size then adjusts fonts, pop-ups, widgets and other parts of the page to create a pleasing layout. This save the publisher from having to develop a separate native mobile application or a separate mobile web page.

3.4. Use Cases

To aid in determining the success of the application, three use cases have been designed to represent expected uses of the application by fishery stakeholders. These were selected to provide a variety of users from the fishery stakeholders as well as questions asked and data sought. The first use case centers around a fisherman trying to find the best price for his fish. The second use case looks at a fish house looking for a species of fish its boats do not target, and that they must import. The final case looks at a restaurant owner seeking to add a new species of fish to the menu, but they would like to find something local that is affordable and also readily available.

3.4.1. Fishermen Use Case

The first use case involves fisherman trying to determine the best county in which to sell his catch. For this example, an independent fisherman is used, as contracted or “house” fisherman would likely be obligated to sell at their dock of origin (if the fish house owns the boat, it will almost always sell to the owning fish house). This fisherman will be searching for a place to off load a catch typical to the bandit boat fleet of the western Gulf coast of Florida. This catch typically consists of Red Grouper (*Epinephelus morio*), Gag Grouper (*Mycteroperca microlepis*), and Red Snapper (*Lutjanus campechanus*). The figures examined will be for the year 2015, as the fisherman is only interested in the most recent prices paid, and the county must reside on the west Gulf coast of Florida. The geographic constraint in this example use case is due to the cost of travel to offload a catch. It makes little economic sense to spend thousands of dollars on fuel to start a trip in Tampa Bay to then sail all the way to Jacksonville in the northeast corner of the state to offload the catch.

3.4.2. Fish House / Processor Use Case

This use case while similar to the above use case, could be considered the mirror opposite. The fish house or processor is looking to secure Dolphin Fish / Mahi Mahi (*Coryphaena hippurus*) for a client. As their hypothetical fleet lacks pelagic (open water, or surface fishing) vessels, they therefore, must find a county where an affordable wholesaler may be located. For this use case the year 1995 was chosen for variation. The fish house is located in Panama City Beach in this example and is not limited by distance because this catch will be transported by truck to their location, and the distance-based cost would be nominal. However, closer would mean less shipping costs, and will be factored into their decision.

3.4.3. Restaurant Use Case

In the final use case is a restauranteur looking to add a new species of fish to the menu and would like to see what the price and availability of a few species of fish within the county. While they know that the prices in the Landings Explorer do not indicate the price they will be paying for fish fillets from a seafood processor, they know that they are relative to the dock price, meaning, in general, the pricier the fish at the dock, the pricier the fillets. For this use case the restauranteur is looking at three species of fish: Atlantic Swordfish (*Xiphias gladius*), Amber Jack (*Seriola dumerili*), and Golden Tile (*Lopholatilus chamaeleonticeps*). For this use case, Miami-Dade County and the year 2010 were chosen.

3.5. Hardware and Software Used

The base GIS software used in this project is the Esri ArcGIS desktop suite. The project initially was begun with version 10.3 of the suite, but now employs v10.5. The ArcGIS server product is at v10.4 at time of writing. The ArcGIS JS API development began at version 4.1 and is currently at 4.4.x. The data layer foundation of the software utilizes Microsoft SQL server 2012, with interoperability via SDE in ArcMap 10.5. The project IDE (integrated development environment) and source control are provided by Microsoft Visual Studio Community 2015 and Visual Studio Online - Team Foundation Server respectively.

To bring the application more in line with intended deployment scenarios, ASP.NET was chosen to be the server application. Rather than handling functionality in traditional HTML, ASP is “intelligent” enough to take a user input, process a response on the server itself, and return dynamic content. ASP allows for functionality like password guarded sections of a site, user account creation, report generation, and much more. This is the reason many large websites are developed using ASP. The second half of this technology is the .NET framework. This Microsoft developed framework allows greater interaction with other aspects of the Windows server environment. Another useful benefit of an ASP application is that HTML can be dynamically added.

For example, the current ASP application has a drop-down list for the user to select a species to display on the map. This user control is currently hard coded with over 120 lines of HTML markup, one for each species. This data could be retrieved from a SQL database tied to a built-in ASP.NET list user control and populated dynamically, and is slated for future development.

With the requirements and software described, the next chapter focuses on the development of the Landings Explorer application. Specific sections are dedicated to widgets and aspects of the application. Functionality of the application is also described and examples from the code are used to illustrate the JavaScript patterns. Chapter 4 shows how all the pieces of the Landings Explorer come together.

Chapter 4 Application Development

This chapter discusses the development of the application, including design considerations, technologies, and languages. Section 4.1 examines the high-level flow of the application and goes into greater detail about the RESTful design pattern. The second section outlines the specific software, platforms, and enterprise environments used while developing the Landings Explorer. Section 4.3 is focused on data processing and preparation of the database. Section 4.4 goes into depth about the development of the web services and their consumption within the application. JavaScript development with the Esri JS API is discussed in Section 4.5. Lastly, ASP.NET and HTML development are discussed in the final section.

4.1. Application Flow

The application features RESTful design (Figure 9), meaning the application consumes data through a REST web service. A REST web service is a data-providing endpoint that can be called within the application to serve up data. In this case, the data served are the landings summaries and the county layer. The data provided by the web services are formatted in JSON (JavaScript Object Notation, a standard data transfer format often used by RESTful web services) which is then used by the ArcGIS JS API to render the map with the attribute data. The services are first created in the ArcMap desktop and published to the server. The data are received in a CSV “flat file.” The data are imported into SQL server 2012; then a table is created in ArcMap by using the SDE connection to the SQL server. The table and county shapefiles are then published to the ArcGIS Server. This process creates the two distinct services described in Section 4.4 Web Services. The map view is then created in the ASP.NET application and the ArcGIS JS API is used to create a MapImageLayer from the joined services. This layer is then added to the map view and displayed to the user.

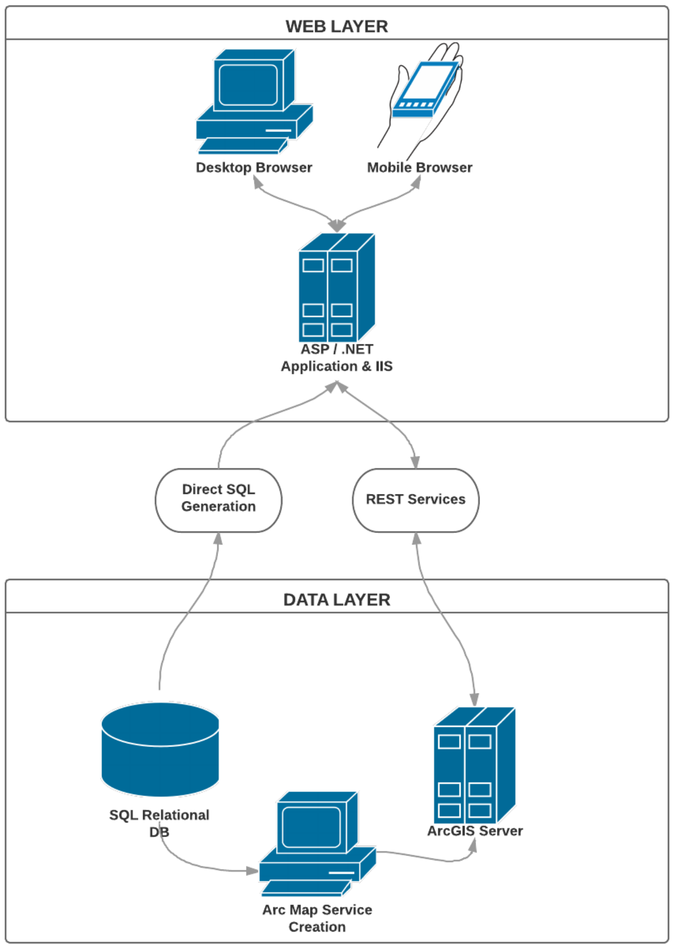


Figure 9. Abstracted flow chart

4.2. Data Processing

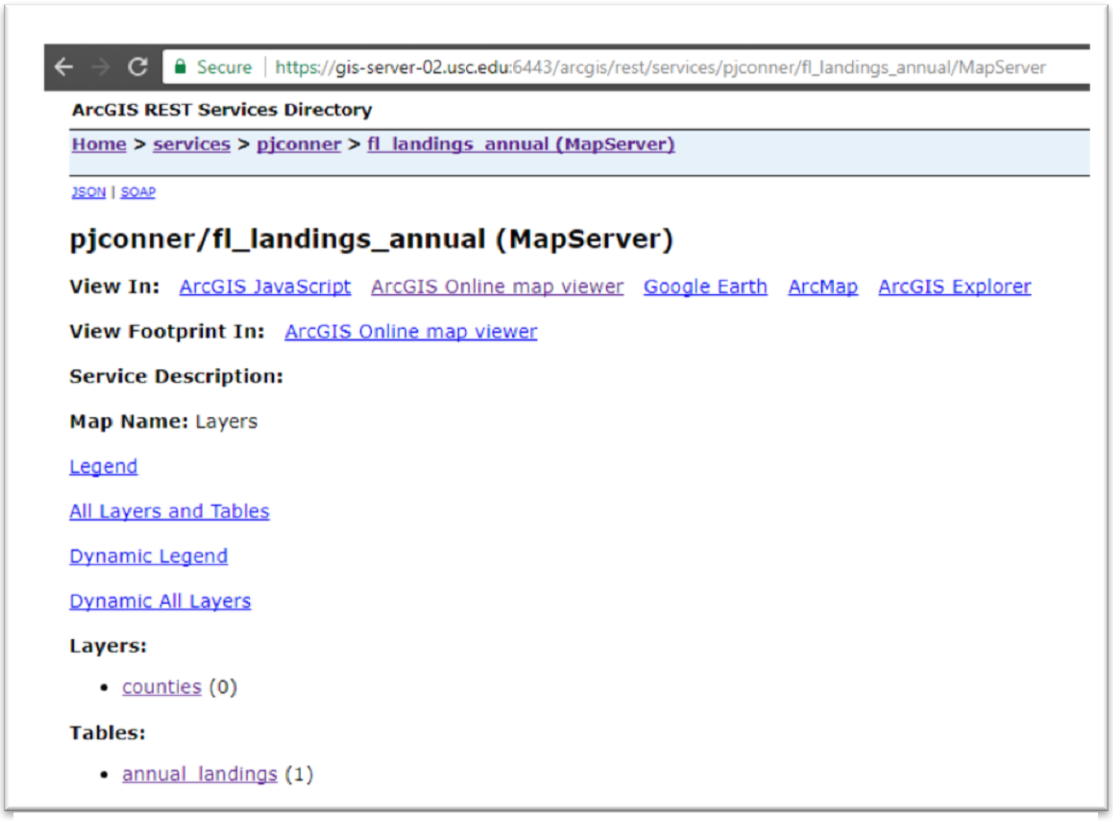
Data processing completed for this project were minimal. For the application, the data consists of all landings from the start of the TTP in 1984 to the last finalized annual data committed in 2016 (FWC 2016). The landings data were imported to MSSQL as-is, so the original data are served through the application. The county shapefile obtained from the USCB had a single column added to unify the county name formatting between the two data sets to permit the data table join explained in Section 4.5. On the front end, formatting was applied using the built-in ArcGIS JS API formatting for specific map elements, in the popup template, for example.

In a normal GIS enterprise environment, the attribute data would be stored within the feature classes organized into feature datasets. With the Landings Explorer, the attribute data is decoupled from the spatial data. Storing years of attribute data within a county polygon feature class is an inefficient data storage methodology. The fact that the landings data is most likely stored in a separate database within the FWCC system architecture supports this decision. This data may not even be stored on the same server. Because of this reason, the application was developed with decoupled attribute and spatial data. However, a web service could be published with the relationship added between the landings attribute tables and the spatial county layer.

With fisheries data being constantly created, the RDBMS data source could provide dynamic queries to provide the most up-to-date data as possible. If this application were to be incorporated into a larger enterprise application, no underlying data structure changes would be required. Rather a single updated query could populate the table service, or a direct query service could be used to bypass the entire table service.

4.3. Web Services

The core of the application utilizes two distinct web services (Figure 10). First, by using the Publish Web Service dialog, the USCB county boundaries were published via ArcMap. A feature service was published for the county layer. Because they are a federal standard, the included FIPS codes were retained.. Next, a table service of the landing summaries was published with the entirety of the summaries being published together. To provide the attribute data by county, the table service was then joined to the county service. More information about the join will be provided in the following section.


Secure | https://gis-server-02.usc.edu:6443/arcgis/rest/services/pjconner/fl_landings_annual/MapServer

ArcGIS REST Services Directory

Home > services > pjconner > fl_landings_annual (MapServer)

JSON | SOAP

pjconner/fl_landings_annual (MapServer)

View In: ArcGIS JavaScript ArcGIS Online map viewer Google Earth ArcMap ArcGIS Explorer

View Footprint In: ArcGIS Online map viewer

Service Description:

Map Name: Layers

Legend

All Layers and Tables

Dynamic Legend

Dynamic All Layers

Layers:

- counties (0)

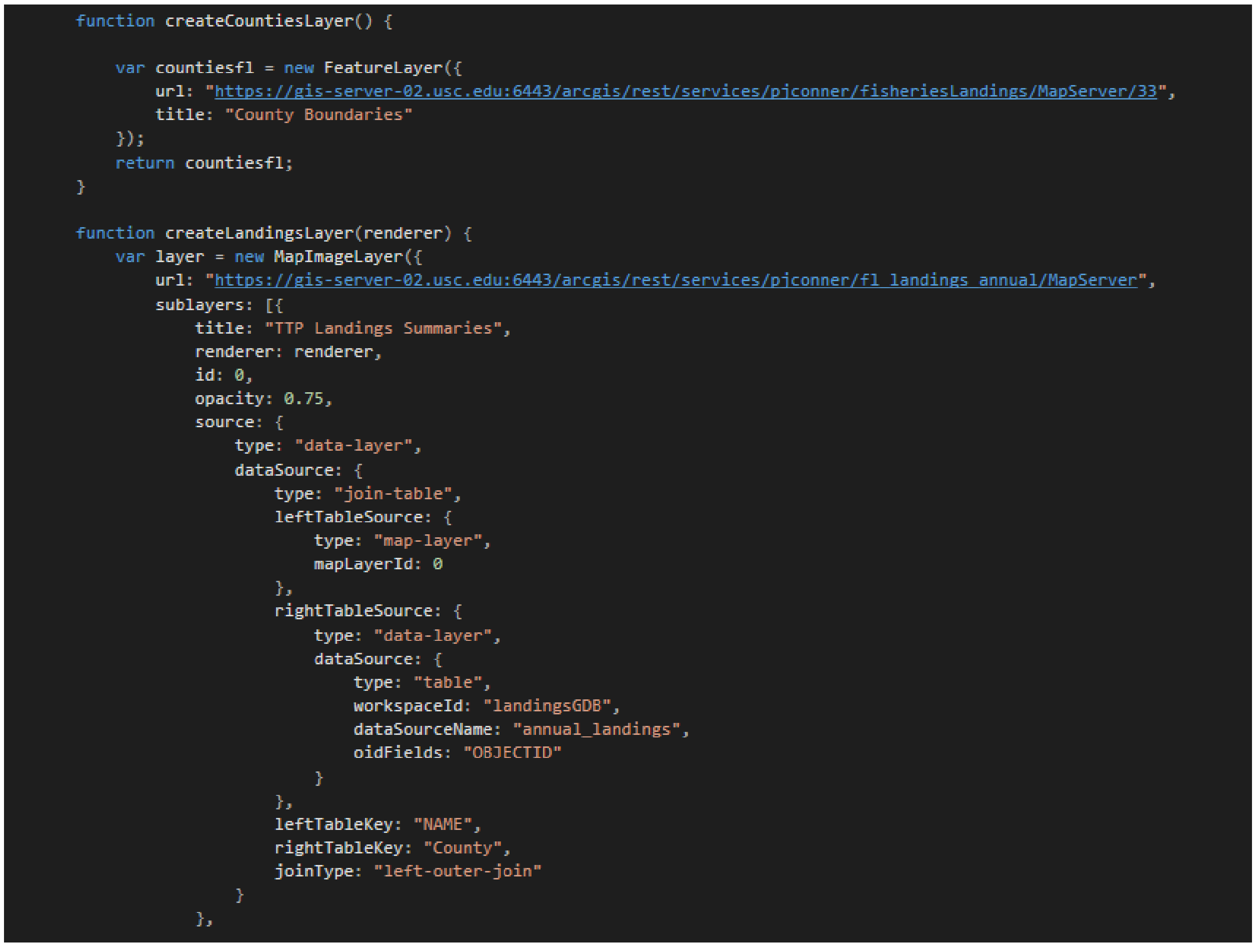
Tables:

- annual_landings (1)

Figure 10. Web service for the Landings Explorer

4.4. Esri JavaScript API 4.X

The JavaScript API is at the heart of this application. The API consumes the web service and renders the map, and offers functionality that may be utilized in future enhancements to this project. Future uses of the API will be discussed in the results, Section 7.4 Future Development. The application joins the landings data to the county shapefile by creating a MapImageLayer with a dynamic data layer (Figure 11). A left join is done on county name, creating a one-to-many relationship.



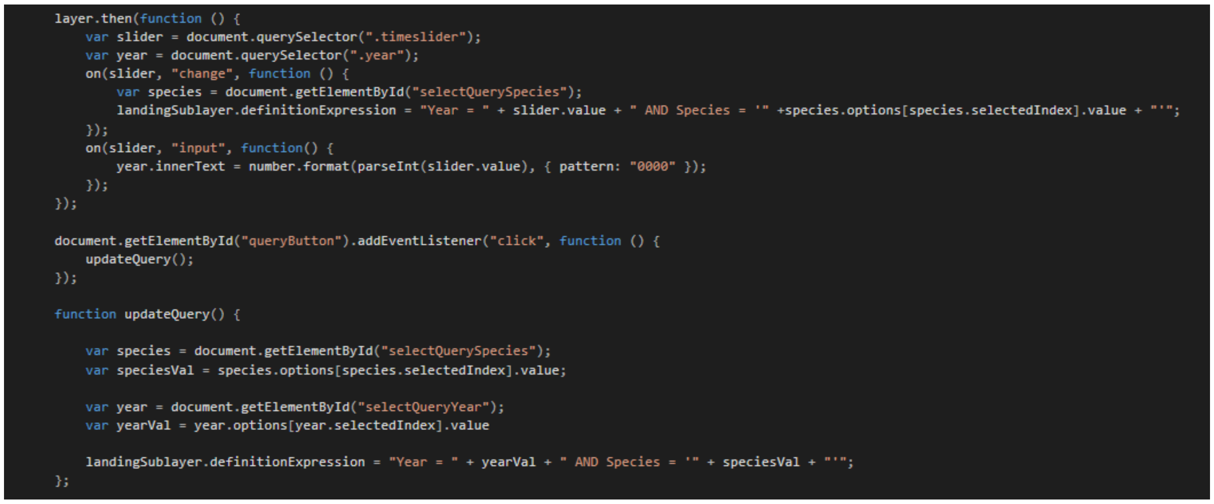
```
function createCountiesLayer() {

    var countiesfl = new FeatureLayer({
        url: "https://gis-server-02.usc.edu:6443/arcgis/rest/services/pjconner/fisheriesLandings/MapServer/33",
        title: "County Boundaries"
    });
    return countiesfl;
}

function createLandingsLayer(renderer) {
    var layer = new MapImageLayer({
        url: "https://gis-server-02.usc.edu:6443/arcgis/rest/services/pjconner/fl_landings_annual/MapServer",
        sublayers: [{
            title: "TTP Landings Summaries",
            renderer: renderer,
            id: 0,
            opacity: 0.75,
            source: {
                type: "data-layer",
                dataSource: {
                    type: "join-table",
                    leftTableSource: {
                        type: "map-layer",
                        mapLayerId: 0
                    },
                    rightTableSource: {
                        type: "data-layer",
                        dataSource: {
                            type: "table",
                            workspaceId: "landingsGDB",
                            dataSourceName: "annual_landings",
                            oidFields: "OBJECTID"
                        }
                    },
                    leftTableKey: "NAME",
                    rightTableKey: "County",
                    joinType: "left-outer-join"
                }
            },
```

Figure 11. JavaScript code sample of a table service join

A definition expression was applied to reduce the landings data set to a specific year species combination, bringing the relationship one to one. Red Grouper and 2016 were selected to be the default selection. The layer is only joined once and then updated as the user interacts with the map and changes the species or year. The JavaScript waits for the layer to finish loading then acts upon it (Figure 12). Future development may utilize jQuery for asynchronous handling, but initial development was done without.



```
layer.then(function () {
    var slider = document.querySelector(".timeslider");
    var year = document.querySelector(".year");
    on(slider, "change", function () {
        var species = document.getElementById("selectQuerySpecies");
        landingSublayer.definitionExpression = "Year = " + slider.value + " AND Species = '" +species.options[species.selectedIndex].value + "'";
    });
    on(slider, "input", function() {
        year.innerText = number.format(parseInt(slider.value), { pattern: "0000" });
    });
});

document.getElementById("queryButton").addEventListener("click", function () {
    updateQuery();
});

function updateQuery() {

    var species = document.getElementById("selectQuerySpecies");
    var speciesVal = species.options[species.selectedIndex].value;

    var year = document.getElementById("selectQueryYear");
    var yearVal = year.options[year.selectedIndex].value

    landingSublayer.definitionExpression = "Year = " + yearVal + " AND Species = '" + speciesVal + "'";
};
```

Figure 12. User control logic within the map JavaScript

4.5. Esri Widgets

There are a trio of built-in Esri widgets that were deployed in the project as-is. These are the legend widget, the layers widget, and the print widget. These were included in the JavaScript file by calling the module alongside the application's other modules. These widgets worked out of the box, with Calcite Maps / Bootstrap providing native support for styling. Calcite maps are the respsonsive design frameworks used in the Landings Explorer and will be discussed more in sections 4.8.1 and 4.8.2. The only manual coding required was defining the layer and legend DIVs within the HTML and ASP pages. Calcite Maps handled styling by managing the CSS classes. This allowed the proper theming, responsive behavior, and other functionality without any additional coding on behalf of the developer.

4.6. Custom Widget Development

Aside from the built-in widgets described above, the remaining widgets were developed as custom controls. The JavaScript event listeners and functional code were included in the map JavaScript file. These widgets were created because the functionality desired had not been developed for the newer version of the JavaScript API or were informational, and a widget was needed to inform the users. The former reason led to the development of the Landings Species Year widget, Landings Over Time widget, and time slider widgets. The introduction and about widgets were developed for the latter reason.

4.6.1. Landings Species Year Widget

Perhaps the most important widget in the application is the Landings Species and Year widget. This widget was the first developed. It is controlled by an event listener on the “query” button. When the button is pressed, the species and year and pulled from the drop-down lists and the definition query is updated. A JS function then refreshes the map with the new definition expression is applied. The where clause of the query is static with only the new year and species being dynamic.

4.6.2. Landings Over Time Widget

This widget was also developed as a simple JS event listener that would read the year value of the selected year chosen on the slider and updated the definition query like the query widget. The code used for both is almost identical, though instead of reading the year from the drop-down control in the query widget, the slider widget year is read instead for updating the definition.

4.6.3. The Pounds, Trips, Value widget

The Pounds, Trips, Value widget allows the user to change the visible renderer. Four renderers were created, one for each of the data attributes in the landings table, except for average price per pound (see Section 6.1.4 for more information). Each renderer was manually created in the map JS file, by defining the class breaks for the data and the color values associated with each. The pounds per trip renderer had field normalization applied, making it the only custom attribute renderer, rather than a renderer from a native attribute from within the table. The control was developed as a simple on off toggle. A series of If/Then statements checks the currently selected rendered when the drop-down list is changed. The previous renderer is removed from the map, then the selected renderer is added.

4.6.4. Introduction and About Widgets

The Introduction and About widgets are mostly informational. The Introduction widget uses a generic widget template copied from existing widgets. As there is no interacting functionality with the map there was not code hooks into map controls. However, there was the need to handle opening and closing of other widgets. The widget required JavaScript code to manage the opening and closing of the panels via button in each section. Since these are all DOM objects with bootstrap classes, normal HTML link buttons could not be used. Incorporating the JavaScript in the map file helped in minimizing control code in the front-end HTML. The controls are event listeners that add and remove the collapse and open panel classes tying directly into the built-in Calcite / Bootstrap functionality. The About widget is built in the same fashion. Basic HTML tags were used in conjunction with the custom widget template to create a list of items. Only the unordered list, bold, div, and paragraph tags were used. This is the least sophisticated of the widgets in the Landings Explorer, requiring no JS at all.

4.7. ASP.NET Development

The application uses .NET framework version 4.0 and ASP.NET version 4.5. The map view is a named HTML division (div) tag, where the division tag occupies all the screen space, except the Calcite Maps menu bar and its related div tags. It is embedded within an ASP.NET page as a standalone page, meaning the map view is the entirety of the APS page. Currently, no

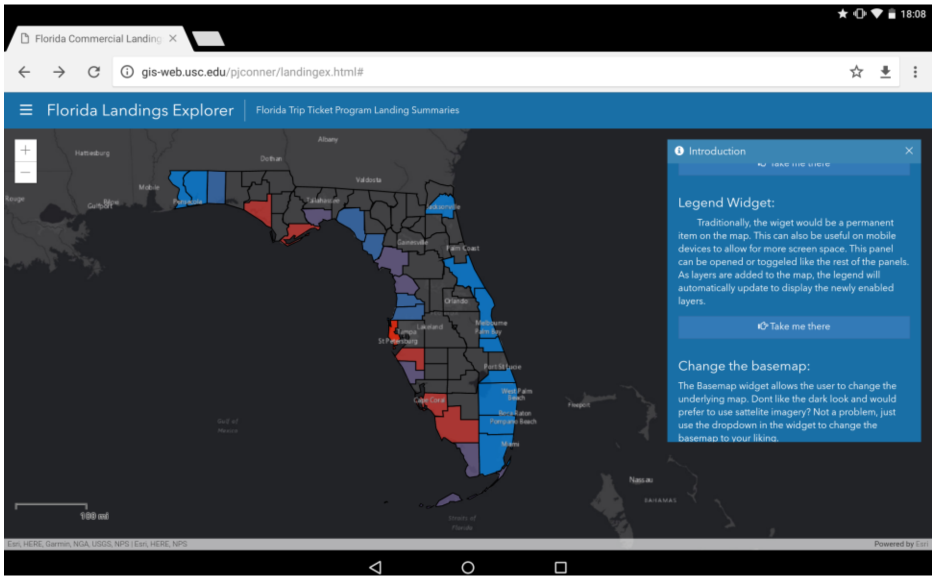
ASP.NET user controls are used in the application as the JS code, and the built-in map widgets handle all existing functionality, however greater ASP.NET integration is scheduled for future development to allow the application to respond to changes in the dataset easily. The ASP application works perfectly when run locally (on the author's desktop), however also due to security concerns, the application is not running on the USC Microsoft Internet Information Services (IIS) server.

4.8. Responsive Design

Initially, a native Android application was considered for the Landings Explorer. Time constraints and maintenance considerations led to a responsive solution to mobile development. The proposed Android application would have been platform specific (Apple users would be excluded from the mobile client) therefor responsive design would bring the application to all platforms able to browse the web on a mobile device. Since the development of the native android application would still consume the web services, no functionality was lost by moving to browser-based clients. When Calcite Maps was found, the decision to develop a responsive application was reinforced.

4.8.1. Bootstrap

Bootstrap frees the developer from managing CSS classes, some basic JS functionality, and creating a specific theme for the application. This allows the user to focus on developing map functionality without having to manage responsive media levels for example. In Figure 13 below, the Landings Explorer renders on a tablet with minimal issue out of the box. Normally the developer would need to specify at what screen size the application would need to change layout and format to better fit a smaller screen. The styling of the application was greatly enhanced by the use of Bootstrap.


Florida Commercial Landing
gis-web.usc.edu/pjconner/landingex.html#
Florida Landings Explorer
Florida Trip Ticket Program Landing Summaries
Introduction
Take me there
100 mi
Powered by Esri

Legend Widget:

Traditionally, the wiget would be a permanent item on the map. This can also be useful on mobile devices to allow for more screen space. This panel can be opened or toggeled like the rest of the panels. As layers are added to the map, the legend will automatically update to display the newly enabled layers.

Change the basemap:

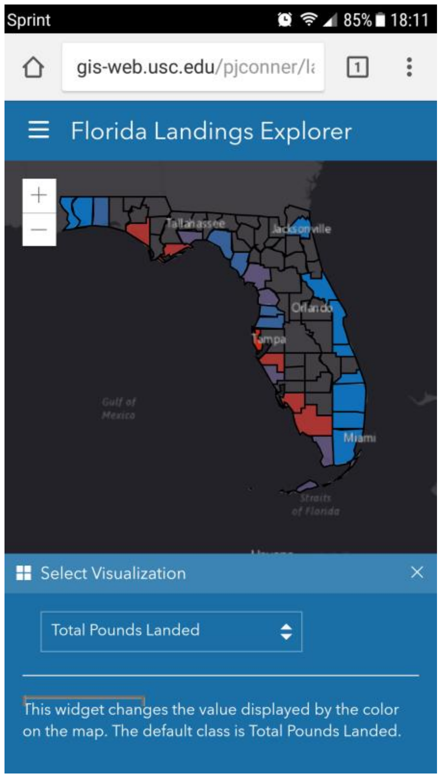
The Basemap widget allows the user to change the underlying map. Dont like the dark look and would prefer to use sattelite imagery? Not a problem, just use the dropdown in the widget to change the basemap to your liking.

Figure 13. Landings Explorer on a Google Nexus 10

Bootstrap provides base classes for styling buttons and glyphicons as seen in the menu. When creating a new panel, the developer does not need to worry about managing styling through manual CSS, the panel classes can be added to a new panel DIV, and the panel now inherits the panel styling. These classes control the background color, text color, font, heading styles, and more automatically. From a development standpoint, it reduces CSS mark up in each element as it exists in a separate stylesheet.

4.8.2. Calcite Maps

Calcite Maps is an Esri module that combines existing Dojo and Bootstrap functionality in a single package. Calcite Map pairs responsive design with pre-styled templates for an out-of-the-box web mapping solution (Figure 14). Classes are included for most currently supported Esri widgets. A menu bar with a click out menu panel is also available. A simple 2D template was chosen, as the design matched closely with the pre-development design. This also sped up development by prepackaging this functionality, styling, and responsive design. It is estimated that another six to eight months of development would have been required to incorporate the functionality of Calcite Maps. However, Calcite Maps is very young in development, and documentation can be difficult to find. Answers to questions can be found by contacting the developers on GitHub or through Esri's GeoNet.

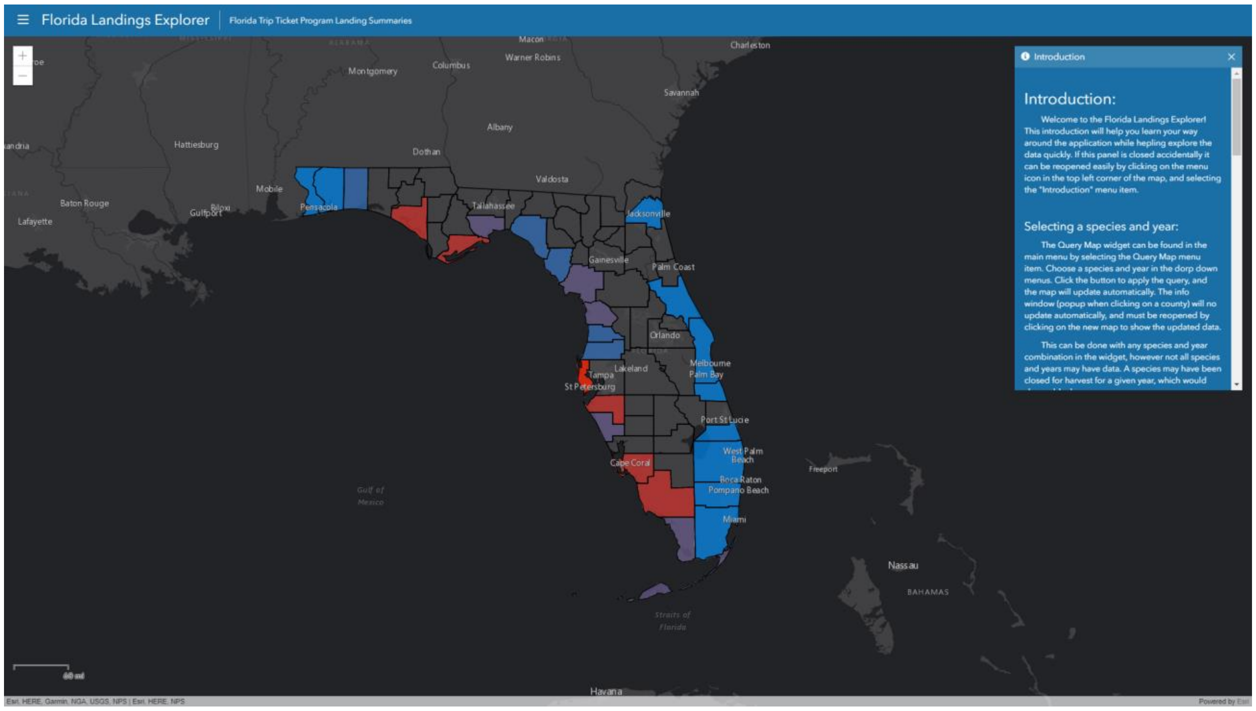


This widget changes the value displayed by the color on the map. The default class is Total Pounds Landed.

Figure 14. Calcite Maps on a Samsung Galaxy S7

Chapter 5 Results

The application (Figure 15) has surpassed performance expectations, despite some setbacks, which are outlined in Section 6.1. All of the requirements were met (Section 5.3), and at the time of publication, the application remains functional, and not in danger of losing stability with the next round of JS API updates provided by Esri. Multiple controls were developed to provide functionality to the user. Each user widget / user control is discussed in the following sections (Sections 5.2.1 through 5.2.4) touching functionality, code, and results of user interaction. Each section highlights a part of the application, with the final two sections showing the results of the use cases and user documentation, respectively.



Introduction:

Welcome to the Florida Landings Explorer! This introduction will help you learn your way around the application while hepling explore the data quickly. If this panel is closed accidentally it can be reopened easily by clicking on the menu icon in the top left corner of the map, and selecting the "Introduction" menu item.

Selecting a species and year:

The Query Map widget can be found in the main menu by selecting the Query Map menu item. Choose a species and year in the dorp down menus. Click the button to apply the query, and the map will update automatically. The info window (popup when clicking on a county) will no update automatically, and must be reopened by clicking on the new map to show the updated data.

This can be done with any species and year combination in the widget, however not all species and years may have data. A species may have been closed for harvest for a given year, which would

Figure 15. The Florida Landings Explorer

5.1. The Menu Bar and Menu Panel

At the top of the application is the menu bar (Figure 16). This stretches across the top of the application in both large screens and small screens. It houses the menu button, the application title, and subtitle. This menu bar can be hidden by selecting the Full-Screen menu item, but it is most often used to open the menu panel. The menu panel features a menu item for each widget including the print widget. If the user closes a widget completely, they can be reopened from this menu.

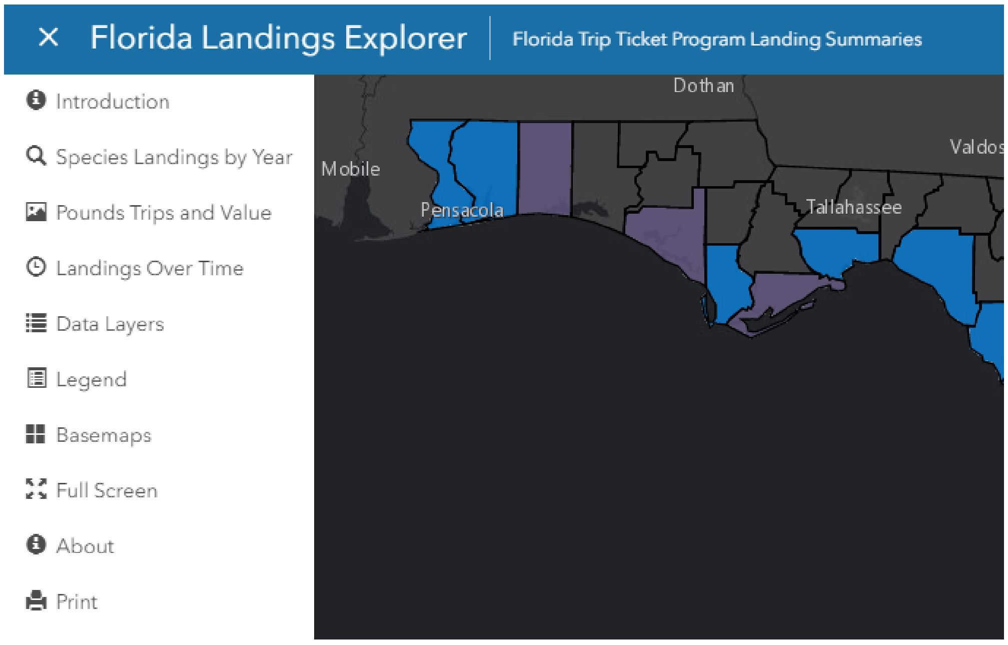


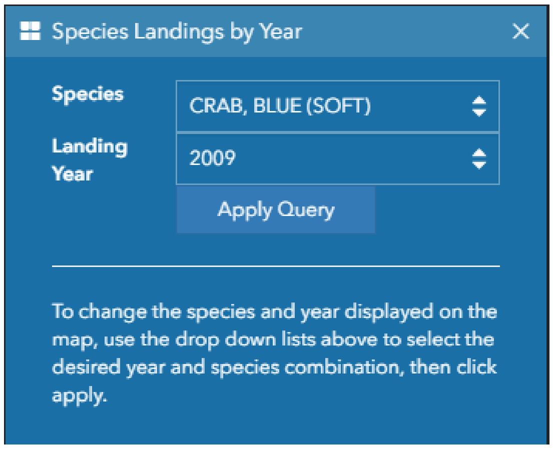
Figure 16. The main menu, featuring the menu bar and panel

5.2. Application Controls

The Landings Explorer consists of multiple widgets as stated previously, most with custom development. These widgets combine custom controls and built-in Esri functionality to create a web application that can display fisheries landings data in a meaningful way. Each section below discusses the individual user controls within the Landings Explorer. These can all be accessed through the Introduction widget or by the Main Menu.

5.2.1. Species Landings by Year

The Species Landings by Year widget allows the user to select a species and year from the data set. The species and year are then “set” in the definition expression. While this works well, there are cases where no data will display due to missing data for that species and year. Results of using this widget can be found in use cases one and two (Figure 21, Figure 22 respectively). Additional development could allow this widget to gracefully handle the missing data by displaying a popup modal or basic JS alert to the user, stating that there is no data for the selected time and species. The data could be missing due to closure or a change in protection status. However, data pertaining to management status is not included in the landings summaries.

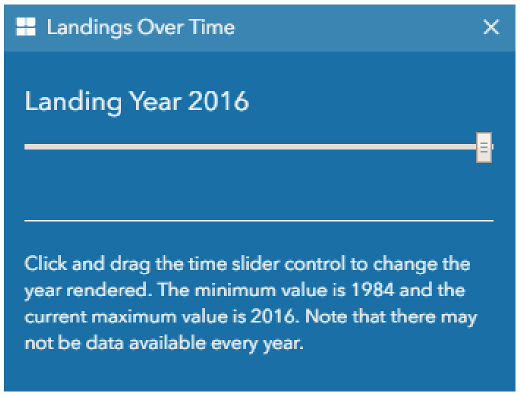


To change the species and year displayed on the map, use the drop down lists above to select the desired year and species combination, then click apply.

Figure 17. Species Landings by Year widget

5.2.2. Landings Over Time Widget

The Landings Over Time widget was also custom developed as no Esri time slider widget had been developed at the time (Figure 18). By clicking and dragging the slider control in the widget, the user and change the landings year in real time. This allows the use to quickly see how the catch of a specific species has changed over time. This widget can also be susceptible to the no data issue as mentioned in the previous section. The results of using this widget can be found in the discussion of use case three (Figure 23).

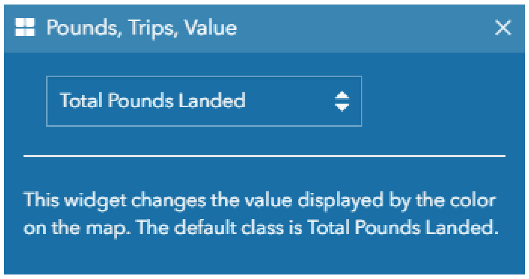


Click and drag the time slider control to change the year rendered. The minimum value is 1984 and the current maximum value is 2016. Note that there may not be data available every year.

Figure 18. Landings Over Time widget

5.2.3. Pounds, Trips, Value Widget

The Pounds, Trips, Value widget provides a control to manage the map renders within the application (Figure 19). By changing the drop down, the renderer will change in the map. Renders for pounds average price per pound, number of trips, and total estimated value can be selected. Each renderer features a unique color allowing the user to quickly identify the difference between the renderers. By default, the first attribute in the table, Total Pounds Landed, is selected. The application will show this upon initial load. The results of using this widget to change the renderer can be found in section 5.3, use case three (Figure 23).



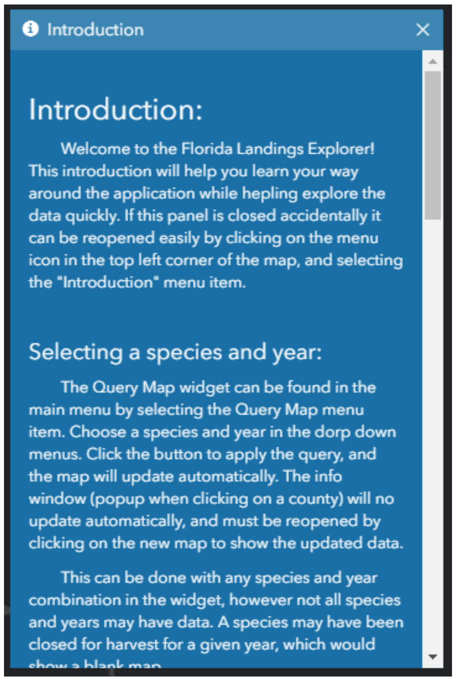
Total Pounds Landed

This widget changes the value displayed by the color on the map. The default class is Total Pounds Landed.

Figure 19. Pounds Trips Value widget

5.2.4. Introduction and About Widgets

The Introduction widget is a custom-made information panel designed to provide the user with usage information upon first load. This widget has multiple sections describing each map control in the application. A link is provided for each section that will bring the user right to the selected widget. The About widget was built from the same template as the Introduction widget however it was much simpler as it only contained text data and some headings. There are sections introducing the author, this thesis project, the Florida fishery, contact information, and data refences.



Introduction:

Welcome to the Florida Landings Explorer! This introduction will help you learn your way around the application while hepling explore the data quickly. If this panel is closed accidentally it can be reopened easily by clicking on the menu icon in the top left corner of the map, and selecting the "Introduction" menu item.

Selecting a species and year:

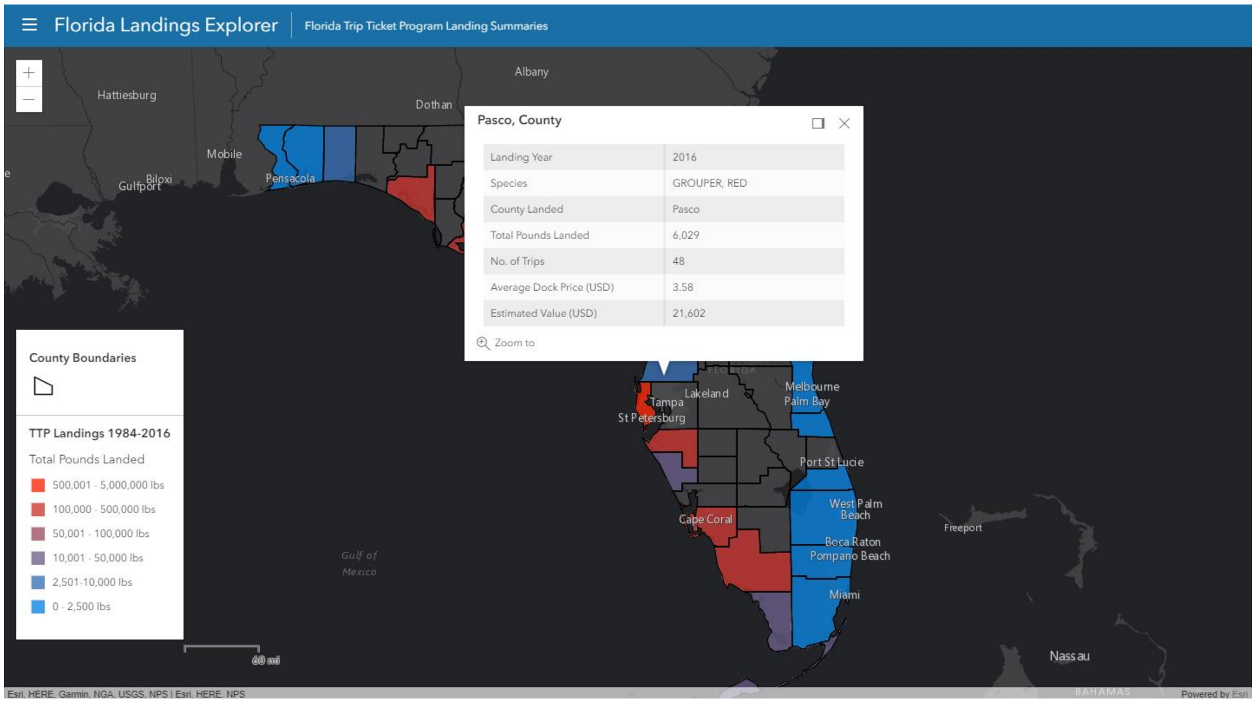
The Query Map widget can be found in the main menu by selecting the Query Map menu item. Choose a species and year in the dorp down menus. Click the button to apply the query, and the map will update automatically. The info window (popup when clicking on a county) will no update automatically, and must be reopened by clicking on the new map to show the updated data.

This can be done with any species and year combination in the widget, however not all species and years may have data. A species may have been closed for harvest for a given year, which would

Figure 20. The introduction widget

5.3. Use Case Results

In our first use case, a fisherman is looking for the best county to sell his Red Grouper, Gag Grouper, and Red Snapper in the year 2015. For this use case, it was assumed that he was leaving out of Tampa Bay. By setting the species to Red Grouper and the year to 2015 and clicking the search button in the Species Landings by Year widget, the map renders appropriately; the user finds that Pinellas County is notorious for landing Red Grouper. However, Pasco County, the next county North offered an average of $0.14 per pound more than Pinellas (Figure 21). For Black grouper, this difference was $0.27. Lastly, Red Snapper was $0.46 higher. Perhaps this trip the user will consider checking the prices in Pasco County.



Pasco, County

Landing Year	2016
Species	GROUPER, RED
County Landed	Pasco
Total Pounds Landed	6,029
No. of Trips	48
Average Dock Price (USD)	3.58
Estimated Value (USD)	21,602

County Boundaries

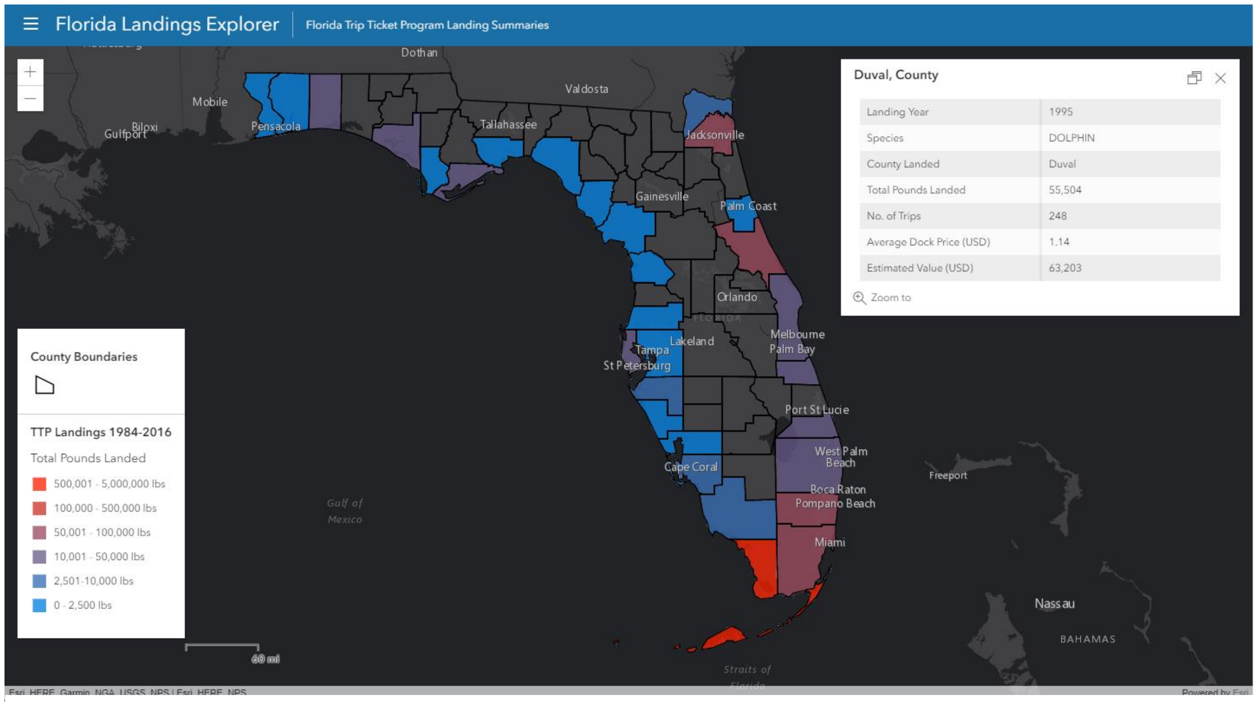
TTP Landings 1984-2016

Total Pounds Landed

🟥	500,001 - 5,000,000 lbs
🟥	100,000 - 500,000 lbs
🟪	50,001 - 100,000 lbs
🟪	10,001 - 50,000 lbs
🟦	2,501-10,000 lbs
🟦	0 - 2,500 lbs

Figure 22. Average Red Grouper price in Pasco County for 2016

In the next use case, a fish house in Panama City Beach was looking to source a few thousand pounds of Mahi Mahi to be freighted in by truck. Using the Species Landings by Year widget again, the fish house manager set the species to Dolphin (Mahi Mahi), and the year to 1995. The user was able to quickly identify that a lot of Mahi was being caught last year in Duval County. Not only did Duval have a much larger portion of the Mahi Mahi catch then his hometown in Bay County, it was on average more than forty cents cheaper.



Duval, County

Landing Year	1995
Species	DOLPHIN
County Landed	Duval
Total Pounds Landed	55,504
No. of Trips	248
Average Dock Price (USD)	1.14
Estimated Value (USD)	63,203

County Boundaries

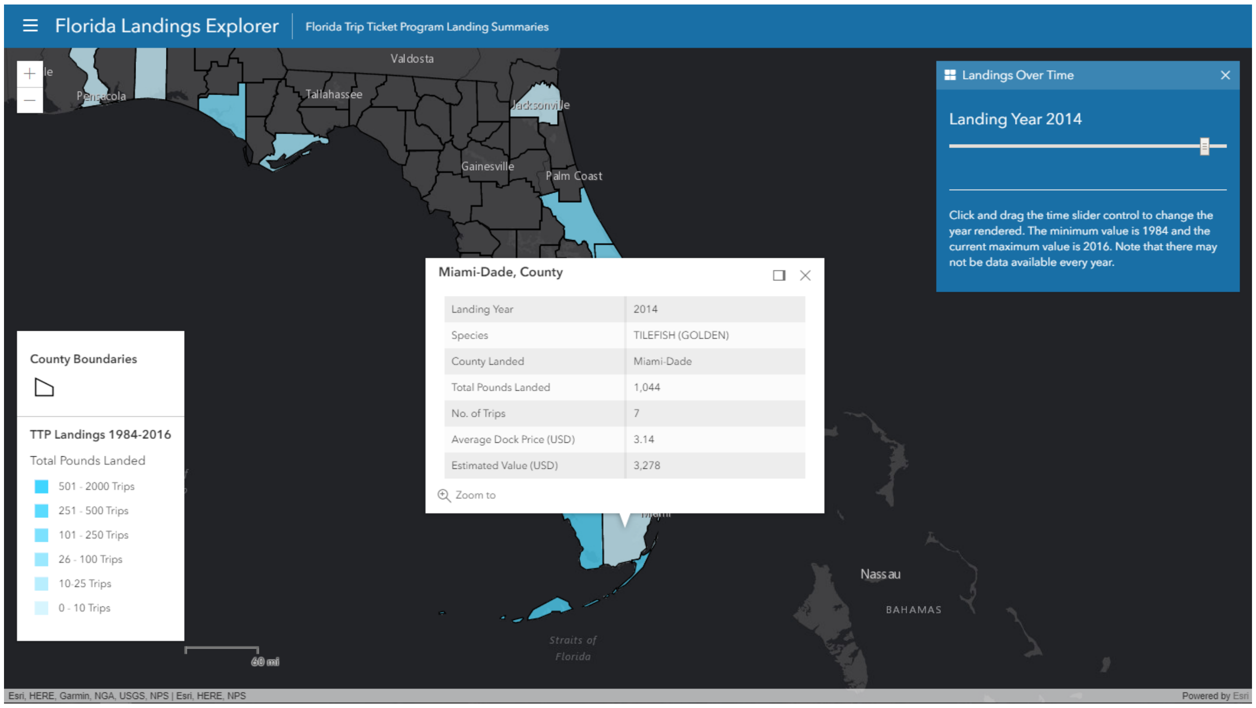
TTP Landings 1984-2016

Total Pounds Landed

🟥	500,001 - 5,000,000 lbs
🟥	100,000 - 500,000 lbs
🟪	50,001 - 100,000 lbs
🟪	10,001 - 50,000 lbs
🟦	2,501-10,000 lbs
🟦	0 - 2,500 lbs

Figure 23. Dolphin fish (Mahi Mahi) landings in Duval County for 1995

The last use case centered around a restaurant owner looking to see the availability and price of Swordfish, Golden Tile, and Amber Jacks. First looking at the Swordfish, they noticed that there was not much catch coming to shore in Miami Dade. However, Broward County to the North is consistently a high producer, dominating the landings almost every year. They also not that swordfish can be a bit expensive. Next, they looked at Tilefish by once again setting the year and species. They then opened the Landings over Time widget, and found that Miami Dade has very little catch, and by switching to the Trips render, they can see it is because there are few trips for Tilefish leaving the county (Figure 22). Lastly looking at the Amberjack, the user notes that while Miami-Dade sees few pounds crossing the docs, the poundage is greater than that of the Tilefish, with a much lower price. With this knowledge, the restauranteur now has the information they need to make the right decision for their financial and availability needs, though all three of these would make a fine addition to any menu.



Landings Over Time

Landing Year 2014

Click and drag the time slider control to change the year rendered. The minimum value is 1984 and the current maximum value is 2016. Note that there may not be data available every year.

Miami-Dade, County

Landing Year	2014
Species	TILEFISH (GOLDEN)
County Landed	Miami-Dade
Total Pounds Landed	1,044
No. of Trips	7
Average Dock Price (USD)	3.14
Estimated Value (USD)	3,278

County Boundaries

TTP Landings 1984-2016

Total Pounds Landed

- 🟦 501 - 2000 Trips
- 🟦 251 - 500 Trips
- 🟦 101 - 250 Trips
- 🟦 26 - 100 Trips
- 🟦 10-25 Trips
- 🟦 0 - 10 Trips

Figure 24. Golden Tilefish trips departing from Miami Dade County for 2014

5.4. User Documentation

The application handles the user documentation in the form of the Introduction widget. Since there is no part of the application that can be customized by the users, standalone documentation was deemed unnecessary. The Introduction widget handles the documentation of the Landings Explorer application. Each widget in the application receives a section and an a description of house to use it. Embedded within the documentation is a button titled “Take me there.” This will open each widget when the user is done reading the instructions. The instructions tell the user how to open the widget by means of the menu as well as the “Take me there” button.

Chapter 6 Conclusions and Discussion

The application is successful in meeting all of the requirements outlined in Chapter 3. However, this was not without significant challenges.

This chapter discusses successes, pitfalls and hurdles encountered during the development process, as well as how to use the Landings Explorer as a module, other uses, and continuing development. The first section discusses the substantial development problems encountered. This section is broken down into subsections to provide finer detail about specific development challenges. The second major section of this chapter discusses procedures for deploying the landings explorer within in a larger enterprise application and system. Section 6.3 discusses other potential uses for the application which are not related to the commercial fishery. Lastly, Section 6.4 outlines potential future development and a road map for achieving the next version of the Landings Explorer application.

6.1. Use Case Results

The first use case was successful in that the fisherman was able to locate a better price for his catch. The second and third use cases were also successful in that they were able to find an affordable source of seafood for whole sale and restaurant. The application was responsive as displayed in the web and mobile clients without a significant change in user experience. Multiple renderers were included in the application, allowing the user to change the colors and the attribute displayed on the map. While not currently deployed, the application is working in ASP.NET, meeting the architectural requirements.

6.2. Development Challenges

As with almost all development projects, many challenges were encountered. Some were caused by data limitations, some by the in-development technology used in the project. The following section is divided into four sub sections each relating to a specific challenge encountered while developing the Landings Explorer. The first section discusses the challenges of being an early adopter and working with software which is still currently in development. The second section discusses the challenges of working with a remote deployment situation. While more often than not, sites are deployed remotely, the developer has the option to go physically interact with the host machine if need be and had administrator privileges when working with the remote system. Section 6.1.3 discusses software module conflicts and problems arising when one module requires a specific version of a library while another requires a different version with varying levels of module support. The final development challenges section discusses working with broad data sets and programmatically representing those values.

6.2.1. Bleeding Edge Development

While staying at the fore of GIS technology can bring incredible functionality to an application, there are some caveats. Often times the code is released to the public with features incomplete or there is less support for specific functionality that was included in previous versions. This was the case when working with the new Esri JS API v4.x. Some modules that were released under the 3.x version of the API had not been developed for the 4.x version. One specific example of this is the time slider.

A time slider widget module can be easily imported for time-enabled data sets in the 3.x version. This has yet to see development in the most recent version of the API. It is likely that development will occur in the near future, but at the time of deployment of the Landings

Explorer application, it has not been released. This required that a custom JavaScript solution is used for time slider functionality. This came with some visual bugs in specific browsers like Internet Explorer. While it does work the solution is not ideal, and future development will include a redesign of the time slider widget.

6.2.2. Deploying Remotely in a High-Security System

Remote deployment of a project never seems to be without it’s downfalls. Licensing, administrator privileges, and not being on premises all add to the difficulty of these types of projects. The project was initially meant to utilize the now ArcGIS Enterprise SDE. This would require secure account creation, so that database connections from the GIS server to the SQL server would only have read permissions on the specific project database. This application was developed utilizing the authors own account, which had much greater read and write permissions within the project server. Utilizing the primary development account would have introduced potential security flaws within the USC GIST network. SQL server account creation was not the only roadblock in the Landings Explorer development.

Administrative access to the greater Windows environment through Active Directory permissions was problematic. On the author’s local development machine, direct administrative permissions are granted to run and configure IIS and the application. Lacking those permissions on the remote server, the configuring IIS is not possible. The deployment workflow for this project was: configure and test locally, deploy to IIS folder via FTP, load the site and debug. When ASP errors were encountered, it was found that an IIS misconfiguration was the cause. However, without administrative access IIS could not be properly configured. Other issues such as ensuring the proper version of .NET, SSL, and other application configurations could also not be remedied.

Local releases of the application work in ASP.NET without issue. However, due to the aforementioned security issues, an HTML version of the application was created for final deployment to the USC servers. Since there was no extra Esri SQL integration at the time of development, the site can stand alone in an HTML page without issue. If greater ASP SQL integration was required (for populating front end text, for example), SQL permissions and connections would need to be developed in ASP natively.

6.2.3. Module Conflicts and Versioning

Perhaps one of the largest setbacks in unifying front end design in a satisfying and pleasing way is version conflicts of required dependencies. Calcite maps require an in-house custom version of Bootstrap to work. It comes packaged with Calcite maps and works well out of the box with basic Bootstrap functionality. However, some functionality is missing from this deployment version. While developing the Introduction widget, using a Bootstrap popup modal was considered. However, the popup modal functionality does not exist within the Esri packaged Dojo-Bootstrap. When trying to import a full version of Bootstrap, the modules and functionality overwrite the custom Dojo version, breaking the built-in Calcite maps widgets, preventing the opening and closing of the widgets. It was for this reason that the intro widget was required. The information is not a nicely as displayed as possible, but it does work with the Dojo version of Bootstrap.

6.2.4. Rendering Values Across the Data Set

With such a large dataset, dynamic renderers become relevant. If a single or a few individual maps were published, then using the custom visualization and symbology are perfectly fine. Each individual map can easily be symbolized to fit the range of the dataset. If one was to inspect a Mullet data set, they would find the price range is very small, fifty cents to a dollar in times of scarcity. Red grouper on the other hand routinely fetches a dock price of 3 dollars. On the far end of the scale, extra-large stone crab claws can reach fifty dollars a pound. Since the entire data set ranges from cents to fifty dollars and up, we must stretch the class bins ($0 - $5.00, $5.01-$15.00, etcetera). This means subtle price variation within a single species becomes unreadable, as most counties will fall within the same bin, despite the renderer spanning the entire data set.

One way to handle these cases is to incorporate a third-party library to manage renderer ranges. The data set could be parsed in JSON, and the minimum and maximum values are found for the individual year and species. This could be taken a step further by adding classification schemes like Jenks Natural Breaks. The limitation with the 4.x API is that the MapImageLayer normalization does not exist. In the 3.x API, the user can use these normalization methods to show quartile, natural Jenks, and more. IT is for this reason that the average price per pound render was removed. The user can still click on the map and return the average price per pound in the popup. Enhancing application handling of visualization is a high priority for future application development.

6.3. Final Deployment Considerations

Final deployment into an enterprise solution would be simple. The web services could be copied directly to a local installation of ArcGIS server. The data would be transferred within a GeoDB via the service definition file. These self-contained definition files could then be published to the destination ArcGIS Enterprise environment. If direct SQL access is preferred, it would be at this time that the SDE connections would need to be created for the source databases, then published as a query service alongside the rest of the application services. Enterprise administrator permissions would be required, but tasks such as this should be easily accomplished by the GIS Administrator. Deployment to an already existing ASP.NET application would also relatively simple.

Incorporating the application in an already existing ASP.NET would be simple as the map DIV can be added as a stand-alone page, making the application “plug and play” from a front-end view. Embedding the DIV is also simple, through the additional HTML and includes would need to be migrated from the Landings Explorer page to the intended parent page. Paths for local files would need to be altered to point at the new location of the resources.

6.4. Alternate Usage

Other data could be used in the Landings Explorer with a time series. Ideally, this would be through an Enterprise SDE environment. Data such as distribution of sales of a specific item over time or almost any demographic data could easily fit into the application. Changes to service pointing, layer creation, and renderer class breaks configuration could be made to match the application to the data. Any flat, time series data source could be used. Demographic or Census information could be displayed, though high geographic data resolution, such as census tracts, might require a data architecture to properly implement with acceptable response times.

6.5. Future Development

The Landings Explorer is a living application, at least for the foreseeable future. Since there are no real production requirements at the time of publishing, development can continue without a deployment schedule or deadlines to meet. With most living applications, development will continue to improve on existing functionality, while incorporating new features as they become available (from the JS API 4.x for example). There are some notable future development considerations.

A redesign of the time slider widget is of high priority. The goal of this development would be to either incorporate a new widget or to add data result handling to alert the user in the case of no data. Unification of styling across all browsers would also be done. 3D development is also considered. The calcite maps theme already supports incorporating the menu into a 2D/3D switchable application. However, 3D maps do not currently work on the mobile browsers. The MapImageLayer is also not supported in 3D maps at the time of publication.

References

Anderson, Jeffery. L. 1995. “Incorporating Fisheries Databases into a GIS and Investigating Salmonid Biomass, Elevation, and Gradient Relations in Wyoming Streams.” Master’s thesis, University of Wyoming, Order No. EP17481

Berry, Robert., Gary Higgs, Richard Fry, and Mitch Langford. 2011. “Web-based GIS Approaches to Engage Public Participation in Wind Farm Planning.” *Transactions in GIS* 15, no. 2: 147-172.

Blee, Brendan. 2016. “Creating a Geodatabase and Web-GIS map to visualize drone legislation in the state of Maryland.” Master’s thesis, University of Southern California

Burrows, Jill, Alec Bodzin, David Anastasio, Dork Sahagian, Denise Bressler, Lori Cirucci, Scott Rutzmoser, and Allison Telezke. 2013. “Using Web GIS to enhance tectonics learning and geospatial thinking.” *Science Scope*, 37, no. 4 (December): 29-37.

Close C. H. and G. Brent Hall. 2006. “A GIS-based Protocol for the Collection and Use of Local Knowledge in Fisheries Management Planning.” *Journal of Environmental Management* 78 (August): 341-352

Djamaluddin, Ibrahim, Poppy Indrayani, Yasuhiro Mitani, Shuichiro Tagane, and Tetsukazu Yahara. 2015. “Geographic Information System (GIS) Web Server for Biodiversity Information System.” *Reinwardtia* 14, no. 2 (December): 249-256

Dragićević, Susana. 2004. “The potential of Web-based GIS.” *Journal of Geographic Systems* 6, no. 2 (June): 79-81

Dragićević, Susana and Shivanand Balram. 2004. “A Web GIS Collaborative Framework to Structure and Manage Distributed Planning Processes.” *Journal of Geographical Systems* 6, no. 2 (June): 133–153.

Dowling, Jennifer. 2014. “Finding You Best-Fit Neighborhood: A Web-GIS Application for Online Residential Property Searched of Anchorage, Alaska.” Master’s thesis, University of Southern California

Fu, Pinde. 2015. Getting to Know Web GIS. Redlands, CA US: Esri Press.

Fu, Pinde. and Jiulin Sun 2011. Web GIS: Principles and Applications. Redlands, CA US: Esri Press.

Huang, Dong Mei, Yan Ling Du, and Sheng Qi He. 2013. “Design and Implementation of the Marine Disaster Real-time Evacuation Path on Android Platform.” *Applied Mechanics and Materials* 336-338 (July):2091-2094

Kamel Boulos, Maged N. 2005. “Editorial: Web GIS in Practice III: Creating a Simple Interactive Map of England’s Strategic Health Authorities using Google Maps API, Google Earth KML, and MSN Virtual Earth Map Control.” *International Journal of Health Geographics* 4, no. 1 (September): 22

Kamel Boulos, Maged N, Bryan J Blanchard, Cory Walker, Julio Montero, Aalap Tripathy, and Ricardo Guiterrez-Osuna. 2011. “Editorial: Web GIS in practice X: A Microsoft Kinect natural user interface for Google Earth Navigation.” *International Journal of Health Geographics* 10, no.1 (July): 45

Kim, Minsung, Kamyoung Kim, and Sang-Il Lee. 2013. “Pedagogical Potential of a Web-Based GIS Application for Migration Data: A Preliminary Investigation in the Context of South Korea.” *Journal of Geography* 112, no. 3 (March): 97-107.

Kraak, Menno-Jan. 2004. “The Role of the Map in a Web-GIS Environment.” *Journal of Geographical Systems* 6, no. 2 (June): 83-93

Karnatak, Harish Chandra, Sameer Saran, Karamjit Bhatia, and P. S. Roy. 2007. “Multicriteria Spatial Decision Analysis in Web GIS Environment.” *Geoinformatica* 11, no. 4 (December): 407–429.

Kavadas, S., D. Damalas, S. Georgakarakos, C. Maravelias, G. Tserpes, C. Papaconstatntinou, and G. Bazigous. 2013. “IMAS-Fish: Integrated MAnagement System to Support the Sustainability of Greek Fisheries Resources. A Multidisciplinary Web-Based Database Management System: Implementation, Capabilities, Utilization and Future Prospects for Fisheries Stakeholders.” *Mediterranean Marine Science* 14, no. 1 (June): 109 -118.

Nourjou, Reza and Joel Thomas. 2016. “System Architecture of Cloud-based Web GIS for Real-Time Macroeconomic Loss Estimation.” *MobiGIS '16, Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*: 56-63.

Piarsa, I Nyoman, A.A. Kompyang Oka Sudana, and Gde Wahyu M. Gunadi. 2012. “Web-based GIS by using spatial decision support system (SDSS) concept for searching commercial marketplace - using google MAP API.” *International Journal of Computer Applications* 50, no. 7 (July): 1-5

Pimpler, Eric. 2014. eBook: Building Web and Mobile ArcGIS Server Applications with JavaScript. Olton, GB: Packt Publishing. Accessed June 9, 2017. ProQuest elibrary.

Zavala-Romero, Olmo, Arsalan Ahmed, Eric P. Chassignet, Jorge Zavala-Hidalgo, Agustin Fernández Eguiarte, and Anke Meyer-Baese. 2014. "An Open Source Java Web Application to Build Self-contained Web GIS Sites." *Environmental Modelling and Software* 62 (December): 210-220.

Bibliography

Florida Fish and Wildlife Conservation Commission (FWCC). *Commercial Fisheries Landings 1984 - 2015, 2016. https://public.myfwc.com/FWRI/PFDM/ReportCreator.aspx,* Last retrieved 06/02/2017.

94th U.S. Congress. 1976 "Magnuson-Stevens Fishery Conservation and Management Act (MSA).", Public Law 94-256, 109-479 (Reauthorized 2007)

U.S. Department of Commerce, Bureau of the Census (USCB). *Cartographic Boundary Shapefiles, 2015.* h*ttp://www2.census.gov/geo/tiger/GENZ2015/shp/cb_2015_us_county_500k.zip*, Last retrieved 11/28/2016.

Appendix A: JavaScript Code

```
//##########################
// File Name: landingsMap.js
// Purpose: This is the main javascript file for the Landings Explorer   //
application.
// Author: Philipp Conner
// Last Edited: 12/16/2017
// Contact Email: pjconner@usc.edu
// Based on original code
//##########################

require([
   "esri/Map",
   "esri/views/MapView",
   "esri/widgets/Search",
   "esri/widgets/Popup",
   "dojo/query",
   "esri/layers/MapImageLayer",
   "esri/layers/FeatureLayer",
   "esri/layers/TileLayer",
   "esri/renderers/SimpleRenderer",
   "esri/renderers/ClassBreaksRenderer",
   "esri/symbols/SimpleFillSymbol",
   "esri/widgets/Legend",
   "esri/widgets/LayerList",
   "esri/widgets/Print",
   "esri/widgets/ScaleBar",
   "dojo/dom",
   "dojo/on",
   "dojo/number",

   // Bootstrap
   "bootstrap/Collapse",
   "bootstrap/Dropdown",

   // Calcite Maps
   "calcite-maps/calcitemaps-v0.3",

   "dojo/domReady!"
], function (Map, MapView, Search, Popup, query, MapImageLayer, FeatureLayer,
TileLayer, SimpleRenderer, ClassBreaksRenderer, SimpleFillSymbol, Legend,
LayerList, Print, ScaleBar, dom, on, number) {



    // helper function to create a symbol
    function createSymbol(color) {
      return new SimpleFillSymbol({
        color: color,
        outline: {
          width: 0.5,
          color: [255, 255, 255, 0.4]
        },
        style: "solid"
```

```
  })
}

var defaultSym = new SimpleFillSymbol({
  outline: {
    color: "darkgray",
    width: "0.5"
  }
})


/*****************************************************************
 *
 * Create the map, view and widgets
 *
 *****************************************************************/
// Map
var map = new Map({
  basemap: "dark-gray",
});

// View
var mapView = new MapView({
  container: "mapViewDiv",

  map: map,
  center: [-82, 28],
  zoom: 7,
  padding: {
    top: 50,
    bottom: 0
  },
  breakpoints: {
    xsmall: 768,
    small: 769,
    medium: 992,
    large: 1200
  },
  constraints: {
    rotationEnabled: false
  }
});


// Search - add to navbar
var searchWidget = new Search({
  view: mapView
}, "searchWidgetDiv");


// Change basemaps with panel
query("#selectBasemapPanel").on("change", function(e) {
  mapView.map.basemap = e.target.value;
});

/*****************************************************************
 *
```

```
      * Create Layers
      *
      *****************************************************************/

    function createCountiesLayer() {

      var countiesfl = new FeatureLayer({
        url: "https://gis-server-
02.usc.edu:6443/arcgis/rest/services/pjconner/fisheriesLandings/MapServer/33"
,
        title: "County Boundaries"
      });
      return countiesfl;
    }

    function createTerritorialSeaLayer() {

      var territorialSeafl = new FeatureLayer({
        url:
"https://maritimeboundaries.noaa.gov/arcgis/rest/services/MaritimeBoundaries/
US_Maritime_Limits_Boundaries/MapServer/1",
        title: "12 NM Territorial Sea",
        visible: false
      });
      return territorialSeafl;
    }

    function createEEZLayer() {

      var eEZfl = new FeatureLayer({
        url:
"https://maritimeboundaries.noaa.gov/arcgis/rest/services/MaritimeBoundaries/
US_Maritime_Limits_Boundaries/MapServer/3",
        title: "200NM EEZ and Maritime Boundaries",
        visible: false
      });
      return eEZfl;
    }

    function createLandingsLayer() {
      var landingsLayer = new MapImageLayer({
        title: "TTP Landings 1984-2016",
        url: "https://gis-server-
02.usc.edu:6443/arcgis/rest/services/pjconner/fl_landings_annual/MapServer",
        sublayers: [{
          title: "Total Pounds Landed",
          visible: true,
          renderer: createPoundsRenderer(),
          outfields: ["*"],
          id: 0,
          opacity: 0.75,
          source: {
            type: "data-layer",
            dataSource: {
              type: "join-table",
              leftTableSource: {
                type: "map-layer",
          mapLayerId: 0
        },
        rightTableSource: {
          type: "data-layer",
          dataSource: {
            type: "table",
            workspaceId: "landingsGDB",
            dataSourceName: "annual_landings",
            oidFields: "OBJECTID"
          }
        },
        leftTableKey: "NAME",
        rightTableKey: "County",
        joinType: "left-outer-join"
      }
    },

    popupTemplate: {
      title: "{annual_landings.County}, County",
      content: [{
        type: "fields",
        fieldInfos: [
          {
            fieldName: "annual_landings.Year",
            label: "Landing Year",
            visible: true,
            format: {
              digitSeparator: false,
              places: 0
            }
          }, {
            fieldName: "annual_landings.Species",
            label: "Species",
            visible: true,
          }, {
            fieldName: "annual_landings.County",
            label: "County Landed",
            visible: true,
          }, {
            fieldName: "annual_landings.Pounds",
            label: "Total Pounds Landed",
            visible: true,
            format: {
              digitSeparator: true,
              places: 0
            }
          }, {
            fieldName: "annual_landings.Trips",
            label: "No. of Trips",
            visible: true,
            format: {
              digitSeparator: false,
              places: 0
            }
          }, {
            fieldName: "annual_landings.Avg_Price",
            label: "Average Dock Price (USD)",
                visible: true,
                format: {
                  digitSeparator: true,
                  places: 2
                }
              }, {
                fieldName: "annual_landings.Est_Value",
                label: "Estimated Value (USD)",
                visible: true,
                format: {
                  digitSeparator: true,
                  places: 0
                }
              }]

          }]
        }
      }]
    });
    return landingsLayer;
  }

    /*****************************************************************
    * Create renderers for each attribute: pounds, trips, avgprice,
totalval
    *****************************************************************/
  //Pounds Renderer
    function createPoundsRenderer() {
      var poundsRenderer = new ClassBreaksRenderer({
        title: "Percentage of total Annual Catch for
{annual_landings.Species}",
        field: "annual_landings.Pounds",
        legendOptions: {
          title: "Total Pounds Landed of {annual_landings.Species}"
        },
        classBreakInfos: [{
          minValue: 0,
          maxValue: 2500,
          symbol: createSymbol("#0080E1"),
          label: "0 - 2,500 lbs"
        }, {
          minValue: 2501,
          maxValue: 10000,
          symbol: createSymbol("#336CB4"),
          label: "2,501-10,000 lbs"
        }, {
          minValue: 10001,
          maxValue: 50000,
          symbol: createSymbol("#665987"),
          label: "10,001 - 50,000 lbs"
        }, {
          minValue: 50001,
          maxValue: 100000,
          symbol: createSymbol("#99465A"),
          label: "50,001 - 100,000 lbs"
        }, {
          minValue: 100001,
          maxValue: 500000,
          symbol: createSymbol("#CC332D"),
          label: "100,000 - 500,000 lbs"
        }, {
          minValue: 500001,
          maxValue: 5000000,
          symbol: createSymbol("#FF2000"),
          label: "500,001 - 5,000,000 lbs"
        }]
      });
      return poundsRenderer;
    }
  //Pounds per Trip Renderer
    function createPPTRenderer() {
      var pptRenderer = new ClassBreaksRenderer({
        field: "annual_landings.Pounds",
        normalizationField: "annual_landings.Trips",
        normalizationType: "field",
        legendOptions: {
          title: "Average Pounds Landed per Trip"
        },
        classBreakInfos: [{
          minValue: 0,
          maxValue: 250,
          symbol: createSymbol("#0080E1"),
          label: "0 - 250 lbs"
        }, {
          minValue: 251,
          maxValue: 500,
          symbol: createSymbol("#336CB4"),
          label: "251-500 lbs"
        }, {
          minValue: 501,
          maxValue: 1000,
          symbol: createSymbol("#665987"),
          label: "501 - 1000 lbs"
        }, {
          minValue: 1001,
          maxValue: 2500,
          symbol: createSymbol("#99465A"),
          label: "1001 - 2500 lbs"
        }, {
          minValue: 2501,
          maxValue: 5000,
          symbol: createSymbol("#CC332D"),
          label: "2501 - 5000 lbs"
        }, {
          minValue: 5000,
          maxValue: 20000,
          symbol: createSymbol("#FF2000"),
          label: "> 5000 lbs"
        }]
      });
      return pptRenderer;
    }

  //Number of Trips Renderer
  function createTripsRenderer() {
    var tripsRenderer = new ClassBreaksRenderer({
      field: "annual_landings.Trips",
      legendOptions: {
        title: "Number of Trips Annually"
      },
      classBreakInfos: [{
        minValue: 0,
        maxValue: 10,
        symbol: createSymbol("#CAF2FE"),
        label: "0 - 10 Trips"
      }, {
        minValue: 10,
        maxValue: 25,
        symbol: createSymbol("#A1E9FE"),
        label: "10-25 Trips"
      }, {
        minValue: 26,
        maxValue: 100,
        symbol: createSymbol("#79E0FE"),
        label: "26 - 100 Trips"
      }, {
        minValue: 101,
        maxValue: 250,
        symbol: createSymbol("#50D8FE"),
        label: "101 - 250 Trips"
      }, {
        minValue: 251,
        maxValue: 500,
        symbol: createSymbol("#28CFFE"),
        label: "251 - 500 Trips"
      }, {
        minValue: 501,
        maxValue: 2000,
        symbol: createSymbol("#00C7FF"),
        label: "501 - 2000 Trips"
      }]
    });
    return tripsRenderer;
  }

//Average Price Renderer
  function createPriceRenderer() {
    var priceRenderer = new ClassBreaksRenderer({
      field: "annual_landings.Avg_Price",
      legendOptions: {
        title: "Average Price Per Pound"
      },
      classBreakInfos: [{
        minValue: 0,
        maxValue: 1.00,
        symbol: createSymbol("#F4ED01"),
        label: "0 - $1.00"
      }, {
        minValue: 1.01,
        maxValue: 2.50,
        symbol: createSymbol("#F6F025"),
          label: "$1.01 - $2.50"
        }, {
          minValue: 2.51,
          maxValue: 5.00,
          symbol: createSymbol("#F8F349"),
          label: "$2.51 - $5.00"
        }, {
          minValue: 5.01,
          maxValue: 10.00,
          symbol: createSymbol("#FAF66D"),
          label: "$5.01 - $10.00"
        }, {
          minValue: 10.01,
          maxValue: 20.00,
          symbol: createSymbol("#FCF991"),
          label: "$10.01 - $20.00"
        }, {
          minValue: 20.01,
          maxValue: 50.00,
          symbol: createSymbol("#FFFCB5"),
          label: "$20.01 - $50.00"
        }]
      });
      return priceRenderer;
    }

  //Estimated Value Renderer
    function createValueRenderer() {
      var valueRenderer = new ClassBreaksRenderer({
        field: "annual_landings.Est_Value",
        legendOptions: {
          title: "Total Catch Est. Value (USD)"
        },

        classBreakInfos: [{
          minValue: 0,
          maxValue: 2500,
          symbol: createSymbol("#FFFCB5"),

          label: "$0 - $2,500"
        }, {
          minValue: 2501,
          maxValue: 10000,
          symbol: createSymbol("#FCF991"),

          label: "$2,501-$10,000"
        }, {
          minValue: 10001,
          maxValue: 50000,
          symbol: createSymbol("#FAF66D"),
          label: "$10,001 - $50,000"
        }, {
          minValue: 50001,
          maxValue: 100000,
          symbol: createSymbol("#F8F349"),
          label: "$50,001 - $100,000"
        }, {
          minValue: 100001,
          maxValue: 500000,
          symbol: createSymbol("#F6F025"),
          label: "$100,000 - $500,000"
        }, {
          minValue: 500001,
          maxValue: 10000000,
          symbol: createSymbol("#F4ED01"),
          label: "$500,001 - $5,000,000 "
        }]
      });
      return valueRenderer;
    }


//#################################################################
//
// Exec Layer Creates and add to map with initial definitions
//
//#################################################################
    var landingsLayer = createLandingsLayer()

    var landingSublayer = landingsLayer.sublayers.find(function (sublayer) {
      return sublayer.id === 0 ;
      });

    landingSublayer.definitionExpression = "Year = 2016 AND Species =
'GROUPER, RED'"

    landingSublayer.renderer = createPoundsRenderer();

    mapView.map.add(landingsLayer);
    mapView.map.add(createCountiesLayer());
    mapView.map.add(createTerritorialSeaLayer());
    mapView.map.add(createEEZLayer());


  //Timeslider Listener
    landingsLayer.then(function () {
      var slider = document.querySelector(".timeslider");
      var year = document.querySelector(".year");
      on(slider, "change", function () {
        var species = document.getElementById("selectQuerySpecies");
        landingSublayer.definitionExpression = "Year = " + slider.value + "
AND Species = '" +species.options[species.selectedIndex].value + "'";
      });
      on(slider, "input", function() {
        year.innerText = number.format(parseInt(slider.value), { pattern:
"0000" });
      });
    });

  //Renderer Listener
    landingsLayer.then(function () {
      var rendererSel = document.getElementById("selectRenderer");

      on(rendererSel, "change", function () {
        var rendererVal =
rendererSel.options[rendererSel.selectedIndex].value;
        landingSublayer.renderer = null;
        var newRenderer;
        switch (rendererVal) {
          case "poundsRenderer":
            newRenderer = createPoundsRenderer();
            break;
          case "pptRenderer":
            newRenderer = createPPTRenderer();
            break;
          case "tripsRenderer":
            newRenderer = createTripsRenderer();
            break;
          case "valueRenderer":
            newRenderer = createValueRenderer();
            break;
        }

        landingSublayer.renderer = newRenderer;
      });
    });

  //Query Handling
    document.getElementById("queryButton").addEventListener("click", function
() {
      updateQuery();
    });

    function updateQuery() {

      var species = document.getElementById("selectQuerySpecies");
      var speciesVal = species.options[species.selectedIndex].value;

      var year = document.getElementById("selectQueryYear");
      var yearVal = year.options[year.selectedIndex].value

      landingSublayer.definitionExpression = "Year = " + yearVal + " AND
Species = '" + speciesVal + "'";
    };

  //Introduction Help buttons
    document.getElementById("queryHelpButton").addEventListener("click",
function () {
      helpButton("#panelQuery", "#collapseQuery")
    });

    document.getElementById("renderHelpButton").addEventListener("click",
function () {
      helpButton("#panelRenderer", "#collapseRenderer")
    });

    document.getElementById("timeHelpButton").addEventListener("click",
function () {
      helpButton("#panelTimeSlider", "#collapseTimeSlider")
    });
    document.getElementById("layerHelpButton").addEventListener("click",
function () {
      helpButton("#panelLayers", "#collapseLayers")
    });

    document.getElementById("legendHelpButton").addEventListener("click",
function () {
      helpButton("#panelLegend", "#collapseLegend")
    });

    document.getElementById("basemapHelpButton").addEventListener("click",
function () {
      helpButton("#panelBasemaps", "#collapseBasemaps")
    });

    document.getElementById("aboutHelpButton").addEventListener("click",
function () {
      helpButton("#panelAbout", "#collapseAbout")
    });


    function helpButton(targetPanel, collapsePanel) {

      //Toggle the info panel as well as the collapbable div
      query("#panelInfo").collapse("toggle");
      query("#collapseInfo").collapse("toggle");

      // Toggle target panel to show and expand
      query(targetPanel).collapse("toggle");
      query(collapsePanel).collapse("toggle");

      };

  //################################################################
  // End Time Slider Playback
  //################################################################

  // Legend
    var legendWidget = new Legend({
      container: "legendWidgetDiv",
      view: mapView
    });

    var legend = new Legend({
      view: mapView
    });
    mapView.ui.add(legend, "bottom-left")

  // LayerList
    var layerWidget = new LayerList({
      container: "layersDiv",
      view: mapView
    });

  // Print
    var printWidget = new Print({
      container: "printDiv",
      view: mapView,
      printServiceUrl:
"https://utility.arcgisonline.com/arcgis/rest/services/Utilities/PrintingTool
s/GPServer/Export%20Web%20Map%20Task"
    });


  // Scalebar
    var scaleBar = new ScaleBar({
      view: mapView,

    });

    mapView.ui.add(scaleBar, {
      style: "ruler",
      position: "bottom-left"
    });

    /******************************************************************
     *
     * Synchronize popup and Bootstrap panels
     * Code modified from Calcite Maps - 2017
        *
     ******************************************************************/

    // Popup - dock top-right desktop, bottom for mobile
    mapView.watch("widthBreakpoint", function(breakPoint){
      if (breakPoint === "medium" || breakPoint === "large" || breakPoint ===
"xlarge") {
        mapView.popup.dockOptions.position = "top-right";
      } else {
        mapView.popup.dockOptions.position = "bottom-center";
      }
    });

    // Popup - show/hide panels when popup is docked
    mapView.popup.watch(["visible", "currentDockPosition"], function(){
      var docked = mapView.popup.visible &&
mapView.popup.currentDockPosition;
      if (docked) {
        query(".calcite-panels").addClass("invisible");
      } else {
        query(".calcite-panels").removeClass("invisible");
      }
    });

    // Panels - undock popup when panel shows
    query(".calcite-panels .panel").on("show.bs.collapse", function(e) {
      if (mapView.popup.currentDockPosition) {
        mapView.popup.dockEnabled = false;
      }
    });

});
```