In [1]:

```
import pandas as pd
import numpy as np
import os
import glob
%matplotlib inline
```

In [2]:

```
import matplotlib.pylab as plt
plt.rcParams['figure.figsize'] = 16, 12
```

Aggregovane informacie

Z roznych datasetov som zobral priebeh chyby a spocital nad nimi priemernu, maximalnu chybu a dalsie kvantily ako aj standardnu odchylku

In [3]:

```
def stats(data, prefix, result={}):
    fcts = ['mean', 'max', 'min', 'std']
    for fct in fcts:
        to_call = getattr(np, fct)
        result[prefix + '_' + fct] = to_call(data)
    pcts = [10, 25, 50, 75, 90]
    for pct in pcts:
        result["%s_%s_percentile" % (prefix, pct)] = np.percentile(data, pct)
    return(result)
```

In [4]:

```
rows = []
for filename in glob.glob('./MY/my_smape*.csv'):
    dataset_name = "_".join(filename.split("_")[-3:-1])
    my = pd.DataFrame.from_csv(filename, header=None)
    result = stats(my.reset_index()[1], 'my', {"dataset": dataset_name})
    fname = "./HW/smape_%s_suma.csv" % dataset_name
    if os.path.isfile(fname):
        hw = pd.DataFrame.from_csv(fname, header=None)
        result = stats(hw.reset_index()[1], 'hw', result)
    exp = pd.DataFrame.from_csv("./HWbetaFALSE/smape_%s_suma.csv" % dataset_name, hea
    result = stats(exp.reset_index()[1], 'exp', result)
    rows.append(result)
df = pd.DataFrame(rows)
```

In [106]:

```
df
```

Out[106]:

	dataset	exp_10_percentile	exp_25_percentile	exp_50_percentile	ex
0	01_zilina	3.930634	5.472191	9.230354	18
1	02_cadca	3.106817	4.367816	6.480772	9.7
2	03_martin	3.959719	6.993530	27.362989	44
3	04_kosice	3.417250	4.882142	8.206301	14
4	05_poprad	3.346723	5.112704	8.437261	15
5	06_humenne	2.758507	4.307479	9.911851	19
6	07_trebisov	2.293819	3.241012	5.122089	13
7	08_presov	4.605122	6.633235	11.849068	21
8	09_svidnik	3.956946	6.933342	11.729912	16
9	8_ba5psc	2.617806	3.951880	6.398209	14
10	8_ba	2.312259	3.596519	6.259489	13
11	90_zahorie	2.968220	4.783356	8.967085	14
12	91_trnava	2.881908	4.384094	7.292784	11
13	92_piestany	1.888619	2.891918	5.319072	8.7
14	93_dun-streda	2.527949	4.477762	7.484632	11
15	94_nitra	2.641418	4.109030	7.641776	15
16	95_partizanske	2.632609	4.359874	7.153344	14
17	96_zvolen	3.481050	4.942429	7.984412	15
18	97_bb	10.251914	12.698682	15.100307	18
19	98_rim-sobota	6.408773	9.263903	14.428853	22
20	99_velky-krtis	6.900238	9.250674	13.584035	20

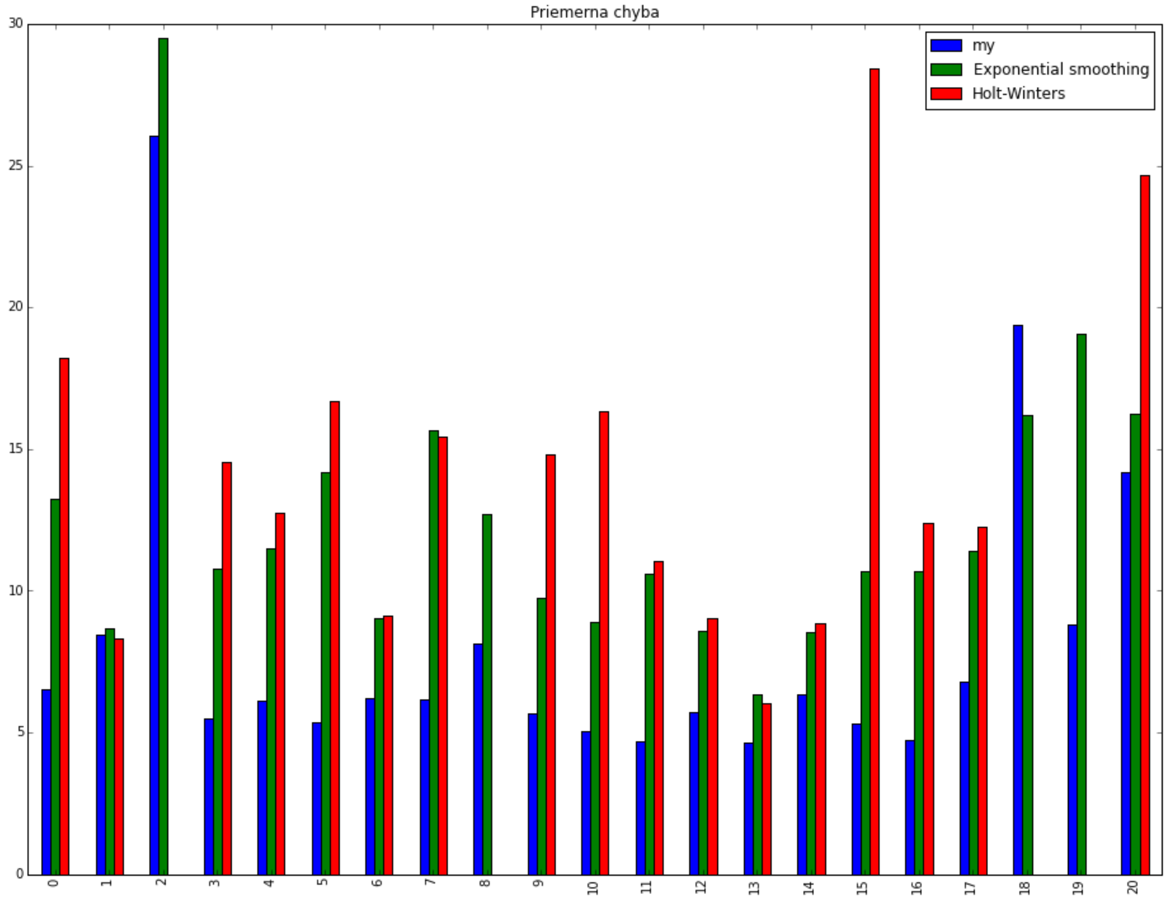
21 rows × 28 columns

Priemerna chyba porovnavanych metod pre rozne datasety

takmer pri vsetkych datasetoch bola nasa metoda ovela lepsia ako porovnavane metody. Pri niektorych datasetoch chyba metoda Holt-Winters, kedze ta nebola schopna pre ne predikovat.

In [107]:

```
df[['my_mean', 'exp_mean', 'hw_mean']].plot(kind='bar', title="Priemerna chyba")
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('Exponential smoothing')
L.get_texts()[2].set_text('Holt-Winters')
```

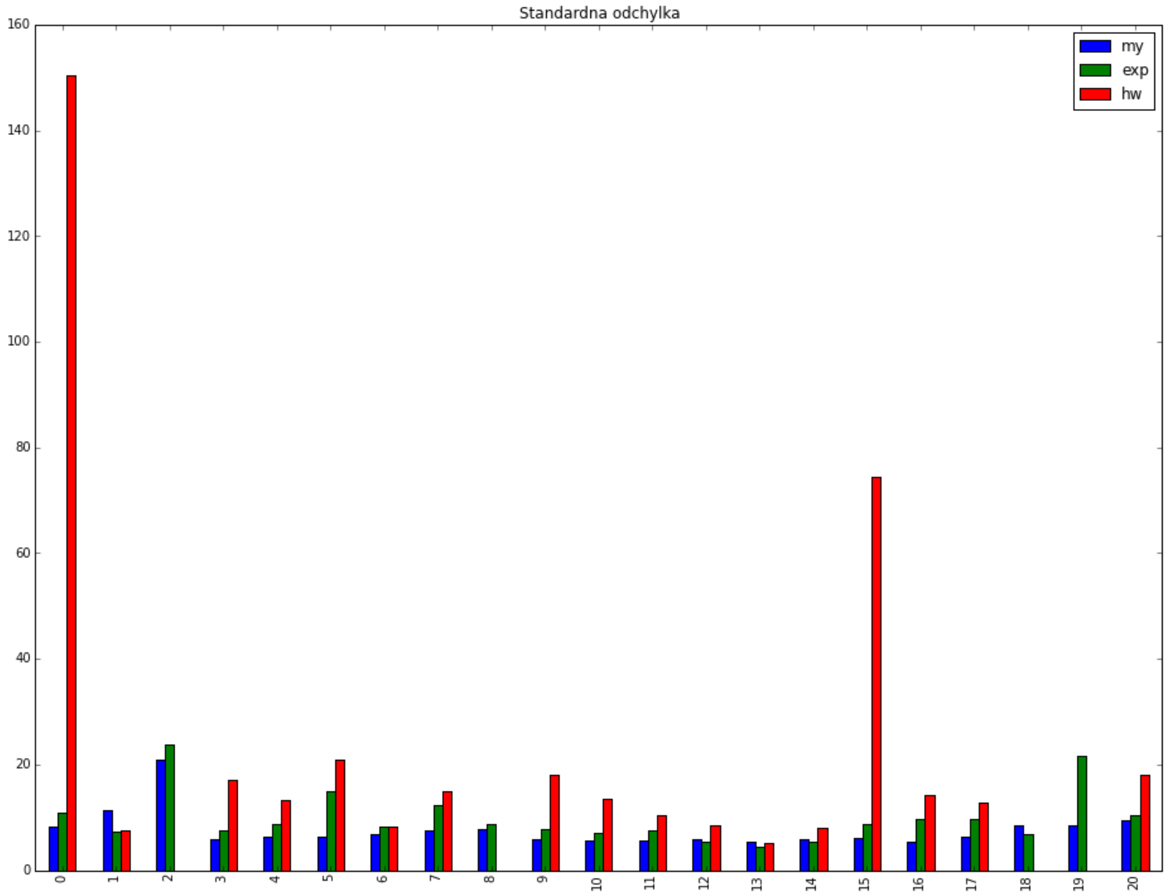


Standardna odchylka chyby

Podla tychto dat vidiet, ze aj priebeh chyby bol pri nasej metode podstatne stabilnejsi.

In [32]:

```
df[['my_std', 'exp_std', 'hw_std']].plot(kind='bar', title="Standardna odchylka")
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('exp')
L.get_texts()[2].set_text('hw')
```

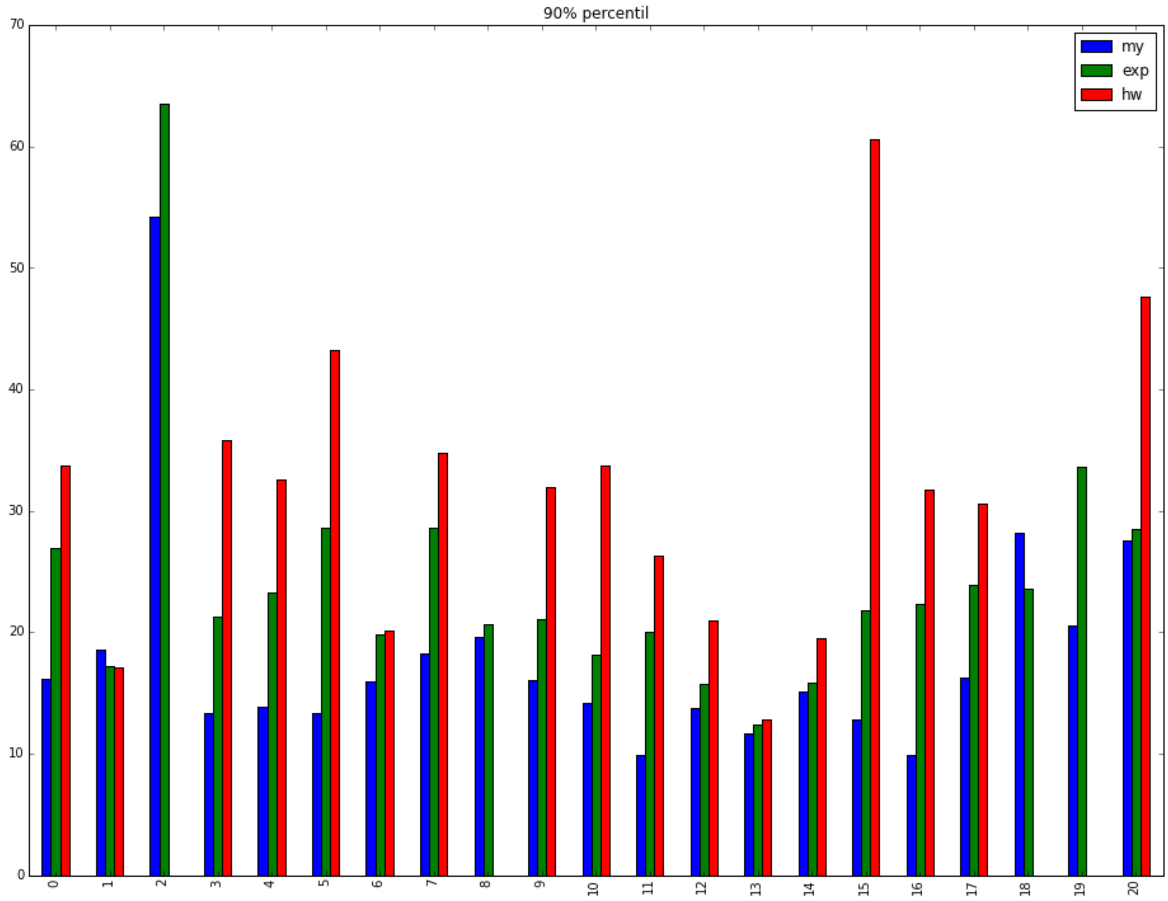


90% percentil chyby

percentily sa daju pouzit na to, aby sme urcili, ci nahodou neboli chyby sustredene v niektorej casti. Teda ci nahodou nebolo skoro rovnake a zopar vynimiek to kazi.

In [33]:

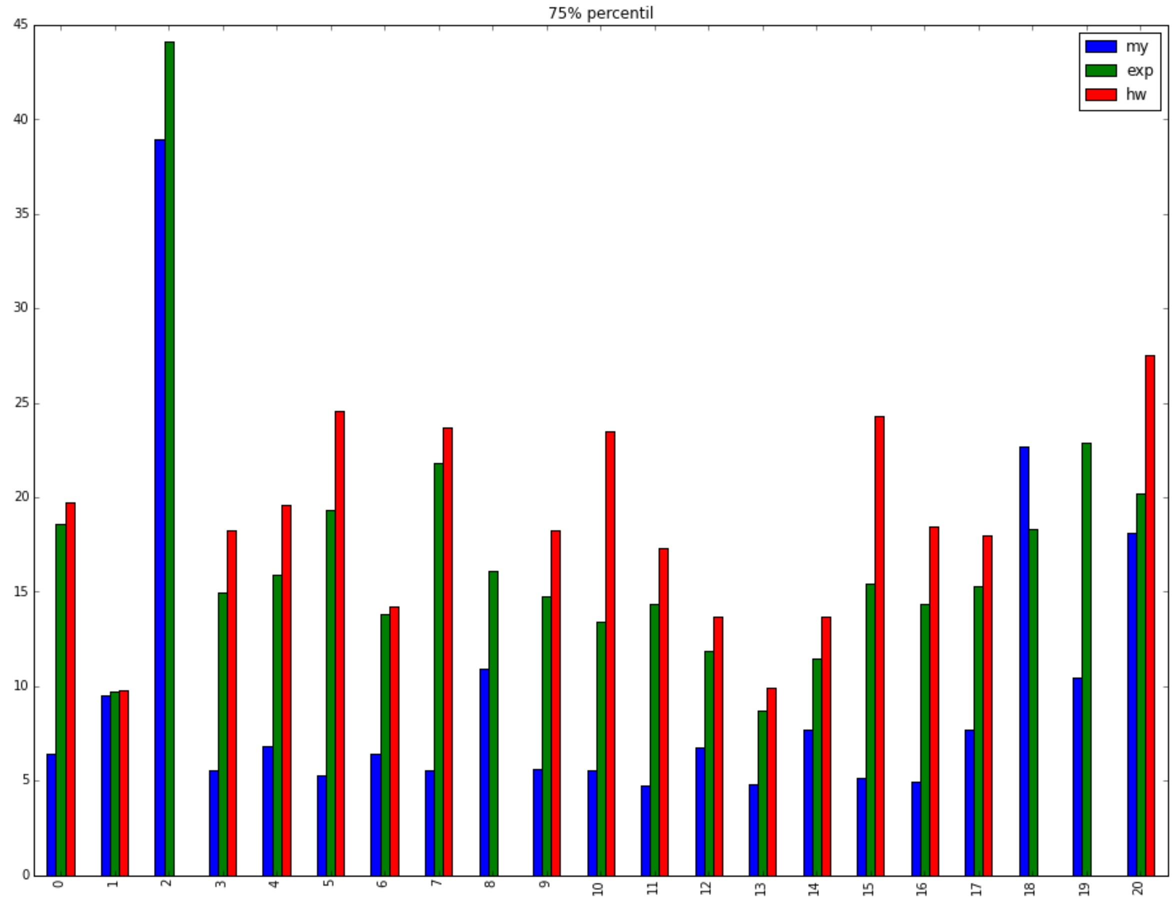
```
df[['my_90_percentile', 'exp_90_percentile', 'hw_90_percentile']].plot(kind='bar', ti
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('exp')
L.get_texts()[2].set_text('hw')
```



70% percentil

In [34]:

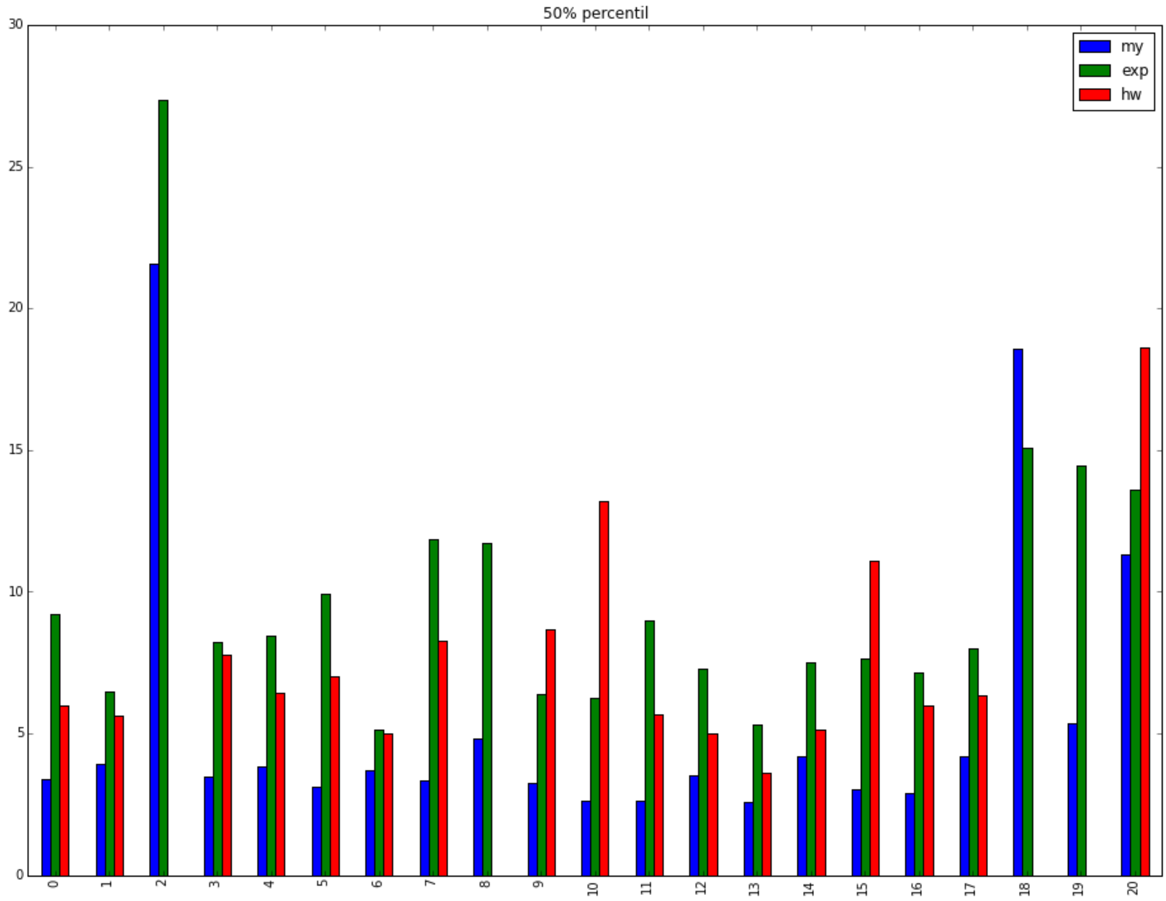
```
df[['my_75_percentile', 'exp_75_percentile', 'hw_75_percentile']].plot(kind='bar', ti
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('exp')
L.get_texts()[2].set_text('hw')
```



50% percentil

In [110]:

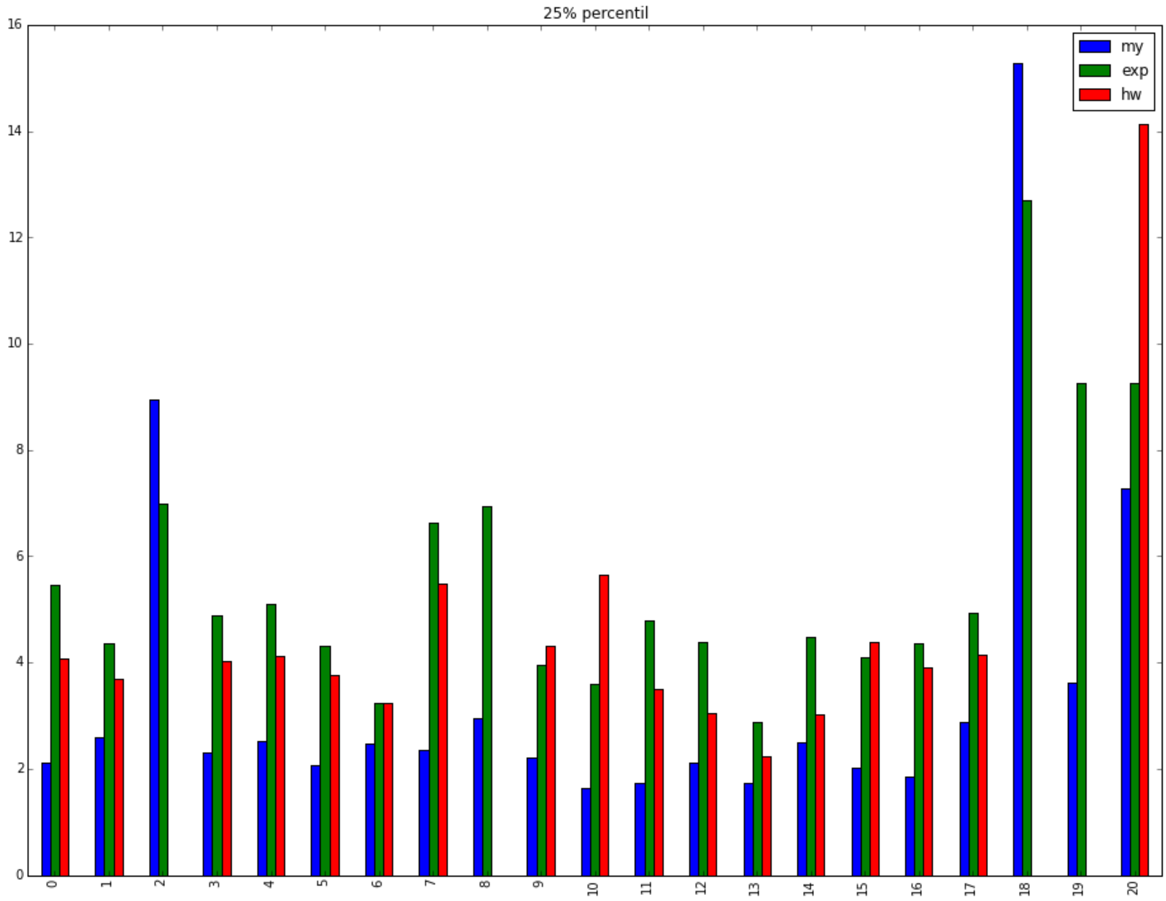
```
df[['my_50_percentile', 'exp_50_percentile', 'hw_50_percentile']].plot(kind='bar', ti
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('exp')
L.get_texts()[2].set_text('hw')
```



25% percentil

In [108]:

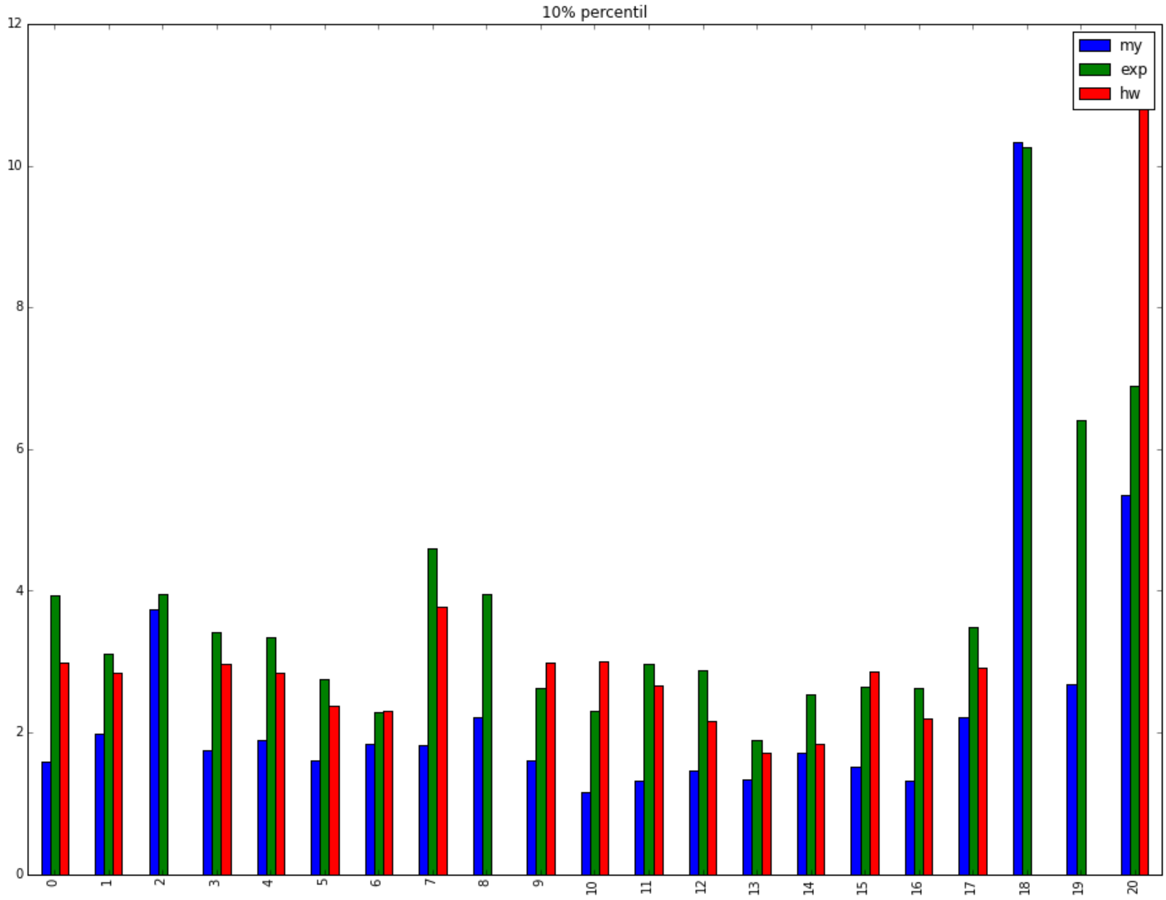
```
df[['my_25_percentile', 'exp_25_percentile', 'hw_25_percentile']].plot(kind='bar', ti
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('exp')
L.get_texts()[2].set_text('hw')
```



10% percentil

In [109]:

```
df[['my_10_percentile', 'exp_10_percentile', 'hw_10_percentile']].plot(kind='bar', ti
L=plt.legend()
L.get_texts()[0].set_text('my')
L.get_texts()[1].set_text('exp')
L.get_texts()[2].set_text('hw')
```



V Piestanoch to fungovalo velmi dobre

Podla priemernych chyb sa zda, ze pre Piestany funguje predikcia velmi dobre. Pozriem sa na to teda trochu podrobnejsie.

In [98]:

```
pn_my_smape= pd.DataFrame.from_csv('./MY/my_smape_92_piestany_suma.csv', header=None)
pn_hw_smape= pd.DataFrame.from_csv('./HW/smape_92_piestany_suma.csv', header=None)
pn_exp_smape= pd.DataFrame.from_csv('./HWbetaFALSE/smape_92_piestany_suma.csv', heade

pn_target = pd.DataFrame.from_csv('./MY/my_target_92_piestany_suma.csv', header=None)
pn_my_pred = pd.DataFrame.from_csv('./MY/my_prediction_92_piestany_suma.csv', header=
pn_hw_pred = pd.DataFrame.from_csv('./HW/prediction_92_piestany_suma.csv', header=Non
pn_exp_pred = pd.DataFrame.from_csv('./HWbetaFALSE/prediction_92_piestany_suma.csv',
```

Predikovanie bezneho tyzna - Piestany

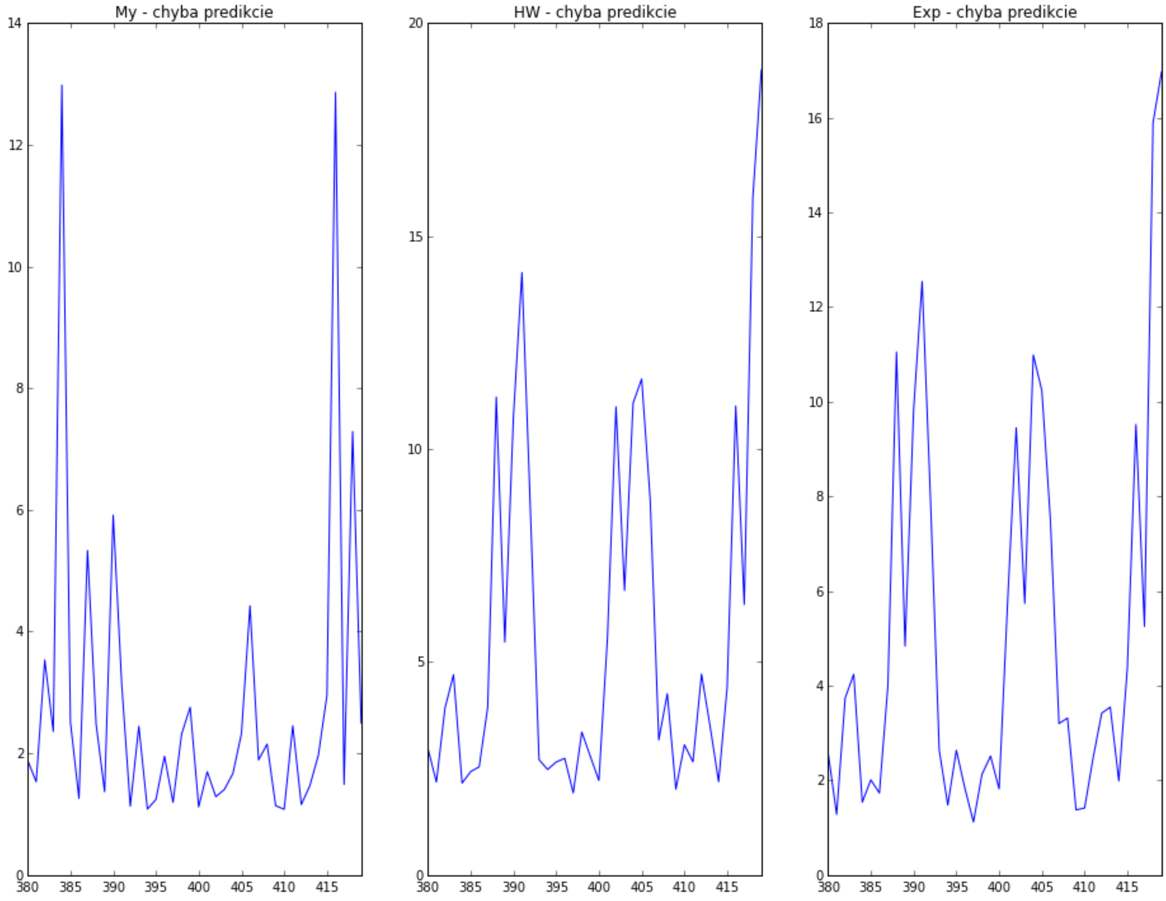
Tu je zobrazena predikcia spotreby pre zopar za sebou nasledujucich beznych tyzdnov. Pocas tyzdna niesu velke vykyvy v chybe predikcie, ale cez vikend sa to meni. Navrhovana metoda sa pomerne rychlo prisposoby zmene absolutnej hodnoty ako aj vzoru. Pre porovnanie ostatne dve metody s tymto maju problem. Strednu hodnotu sice prisposobia, ale vzor sa nemeni. Su schopne naucit sa len jediny vzor.

In [85]:

```
fig, axs = plt.subplots(1,3)
start = 380 # minimalne 0
end = 420 #maximalne 1177
pn_my_smape.reset_index()[1][start:end].plot(ax=axs[0], title='My - chyba predikcie')
pn_hw_smape.reset_index()[1][start:end].plot(ax=axs[1], title='HW - chyba predikcie')
pn_exp_smape.reset_index()[1][start:end].plot(ax=axs[2], title='Exp - chyba predikcie
```

Out[85]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d6f330d0>
```



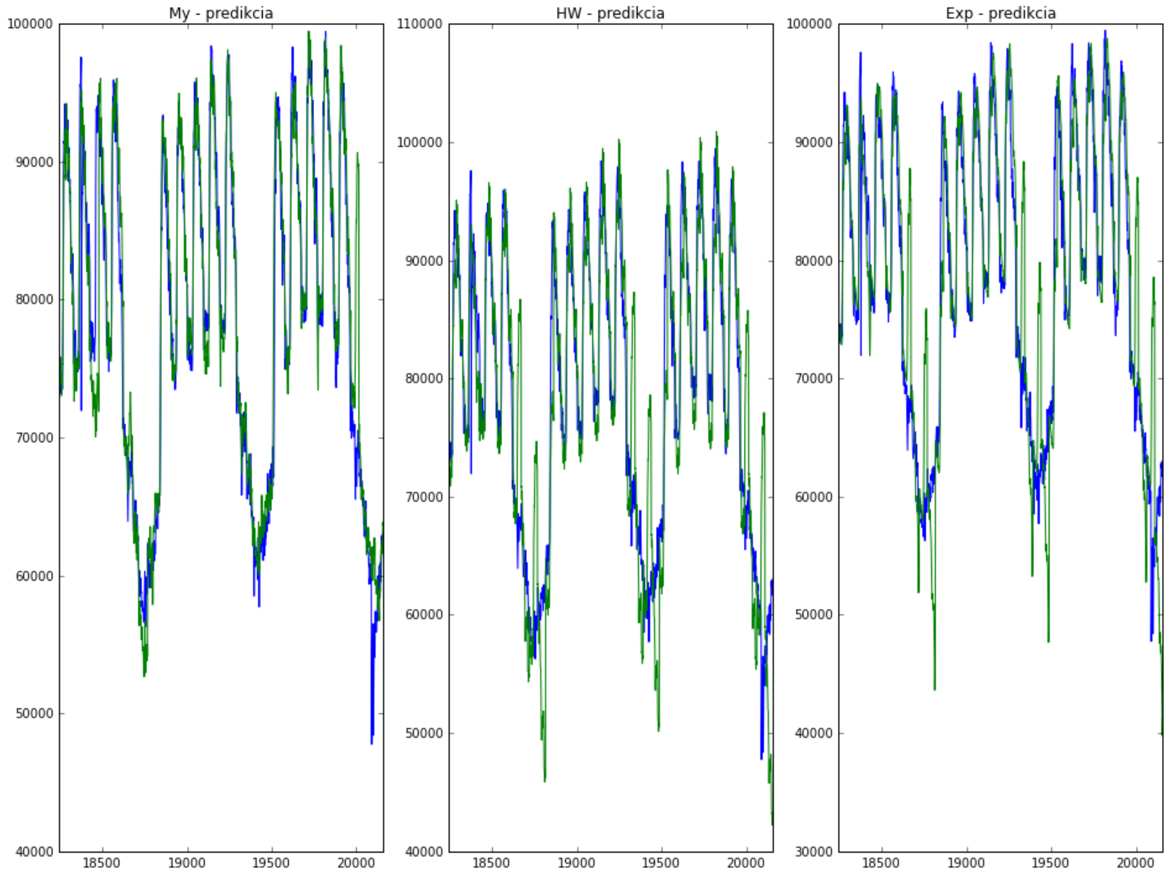
zelena je predikcia a modra je skutocna hodnota

In [86]:

```
fig, axs = plt.subplots(1,3)
p_start = start*48
p_end = end*48
pn_target.reset_index()[2][p_start:p_end].plot(ax=axs[0])
pn_my_pred.reset_index()[1][p_start:p_end].plot(ax=axs[0], title='My - predikcia')
pn_target.reset_index()[2][p_start:p_end].plot(ax=axs[1])
pn_hw_pred.reset_index()[1][p_start:p_end].plot(ax=axs[1], title='HW - predikcia')
pn_target.reset_index()[2][p_start:p_end].plot(ax=axs[2])
pn_exp_pred.reset_index()[1][p_start:p_end].plot(ax=axs[2], title='Exp - predikcia')
```

Out[86]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d6c97f90>
```



Prisposobenie sa nahlej zmene vo vzore - Piestany

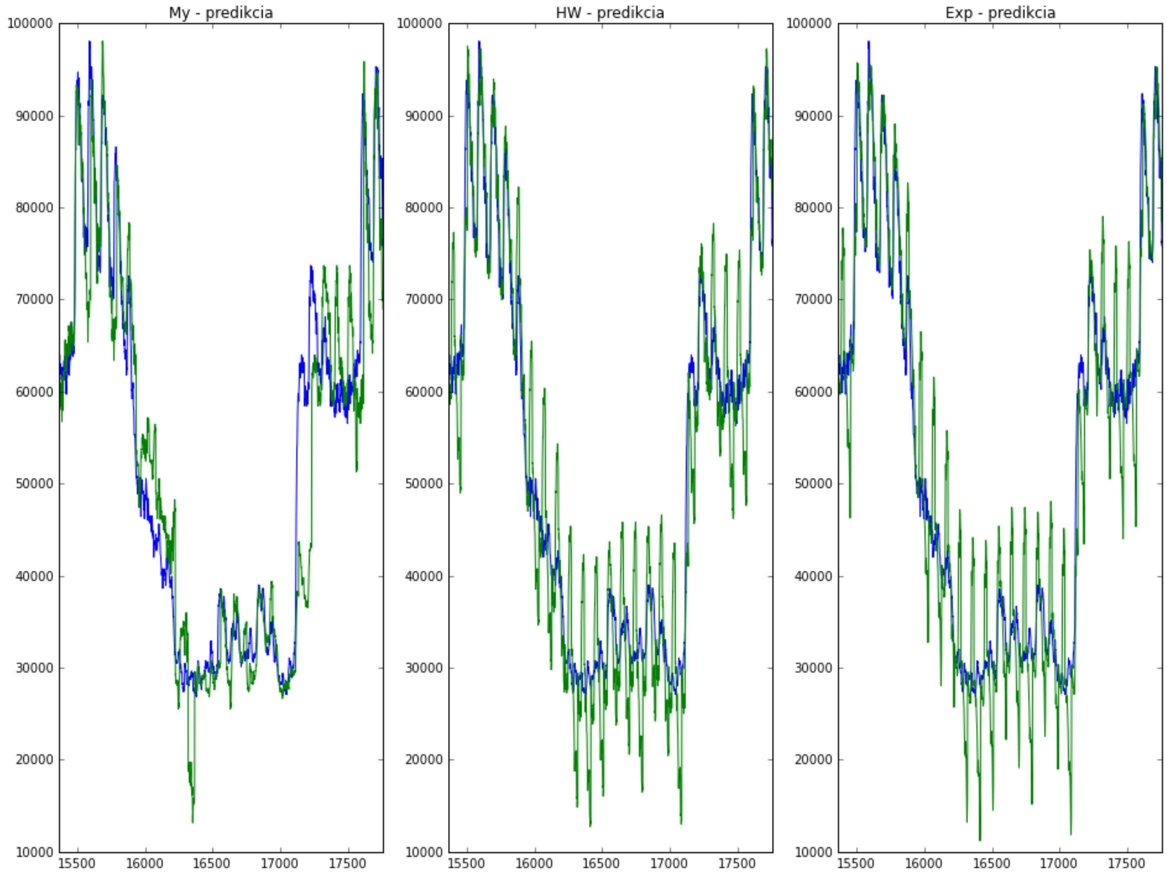
tu je v datach vidiet nahlu zmenu vo vzore, ktora trva minilamne tyzden. zmena nieje len v strednej hodnote, ale aj vo vzore. Navrhnuta metoda sa velmi rychlo prisposobi strednej hodnote a pomerne rychlo aj zmene vzoru. Vo vyslednej chybe su teda len dva skoky a inak je tam stale pomerne mala chyba. Ostatne metody sa tiez ryclo prisposobili zmene strednej hodnoty, ale vzor neupravovali

In [87]:

```
start = 320 # minimalne 0
end = 370 #maximalne 1177
p_start = start*48
p_end = end*48
fig, axs = plt.subplots(1,3)
pn_target.reset_index()[2][p_start:p_end].plot(ax=axs[0])
pn_my_pred.reset_index()[1][p_start:p_end].plot(ax=axs[0], title='My - predikcia')
pn_target.reset_index()[2][p_start:p_end].plot(ax=axs[1])
pn_hw_pred.reset_index()[1][p_start:p_end].plot(ax=axs[1], title='HW - predikcia')
pn_target.reset_index()[2][p_start:p_end].plot(ax=axs[2])
pn_exp_pred.reset_index()[1][p_start:p_end].plot(ax=axs[2], title='Exp - predikcia')
```

Out[87]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d6af4990>
```

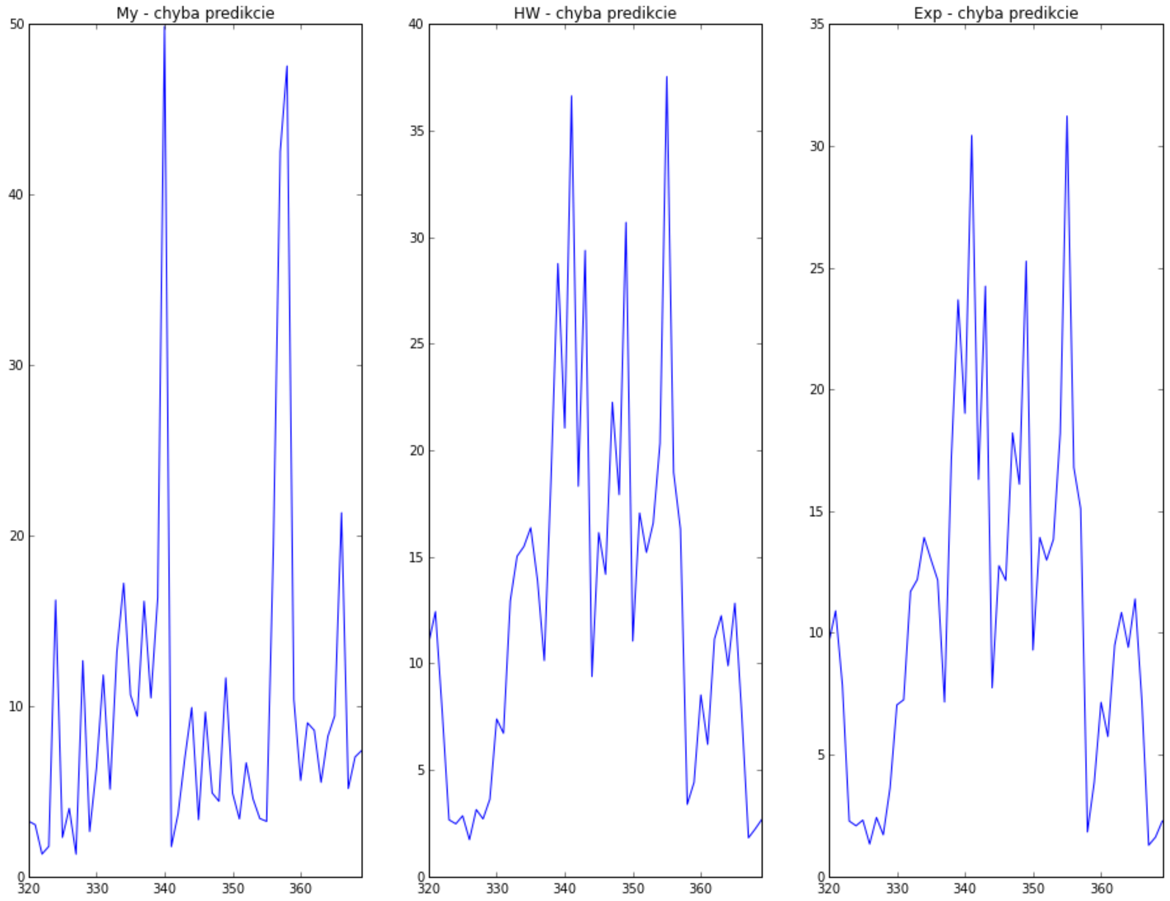


In [88]:

```
fig, axs = plt.subplots(1,3)
pn_my_smape.reset_index()[1][start:end].plot(ax=axs[0], title='My - chyba predikcie')
pn_hw_smape.reset_index()[1][start:end].plot(ax=axs[1], title='HW - chyba predikcie')
pn_exp_smape.reset_index()[1][start:end].plot(ax=axs[2], title='Exp - chyba predikcie
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d6895190>
```



Naopak v Martine to celkovo fungovalo zle

dokonca samotna metoda Holt-Winters tam nebola schopna predikovat

In [93]:

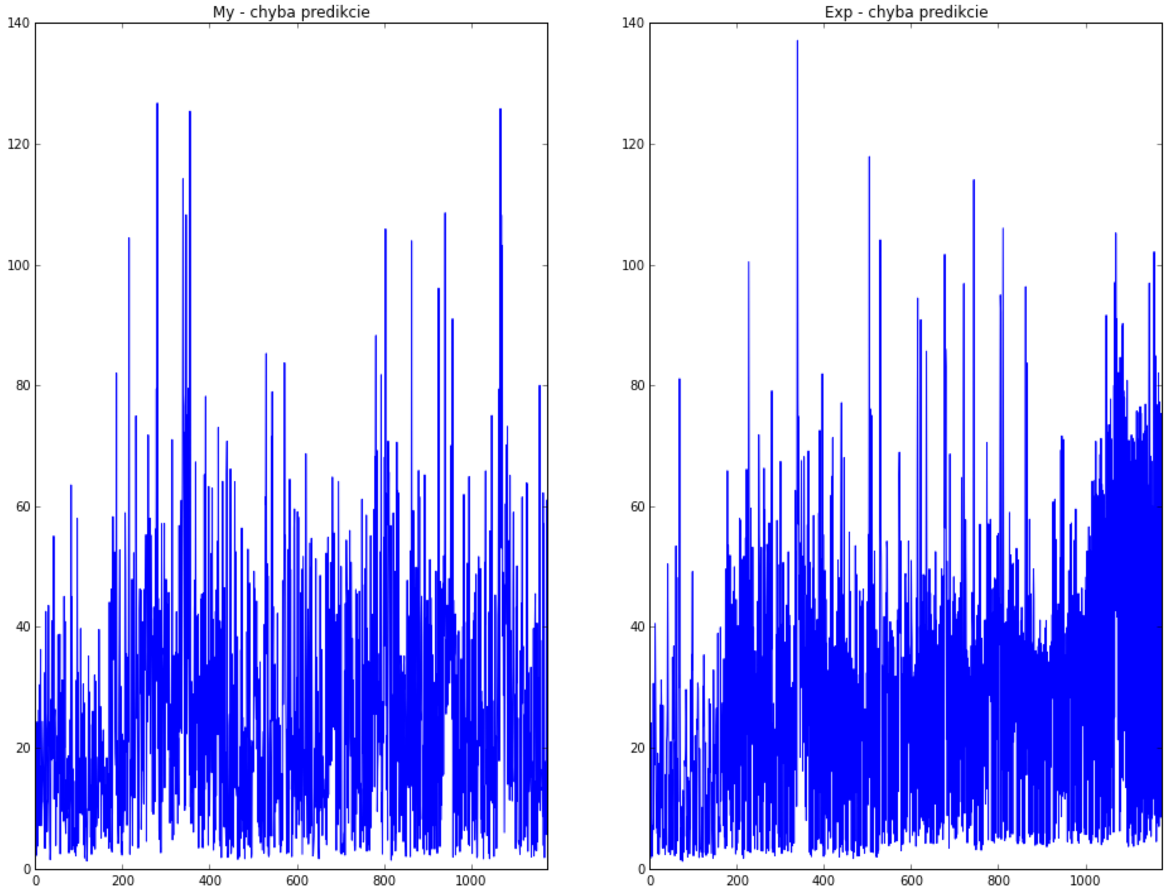
```
mt_my_smape= pd.DataFrame.from_csv('./MY/my_smape_03_martin_suma.csv', header=None)
mt_exp_smape= pd.DataFrame.from_csv('./HWbetaFALSE/smape_03_martin_suma.csv', header=

mt_target = pd.DataFrame.from_csv('./MY/my_target_03_martin_suma.csv', header=None)
mt_my_pred = pd.DataFrame.from_csv('./MY/my_prediction_03_martin_suma.csv', header=No
mt_exp_pred = pd.DataFrame.from_csv('./HWbetaFALSE/prediction_03_martin_suma.csv', he
```

In [97]:

```
fig, axs = plt.subplots(1,2)
start = 0 # minimalne 0
end = 1177 #maximalne 1177
mt_my_smape.reset_index()[1][start:end].plot(ax=axs[0], title='My - chyba predikcie')
mt_exp_smape.reset_index()[1][start:end].plot(ax=axs[1], title='Exp - chyba predikcie
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d62154d0>
```

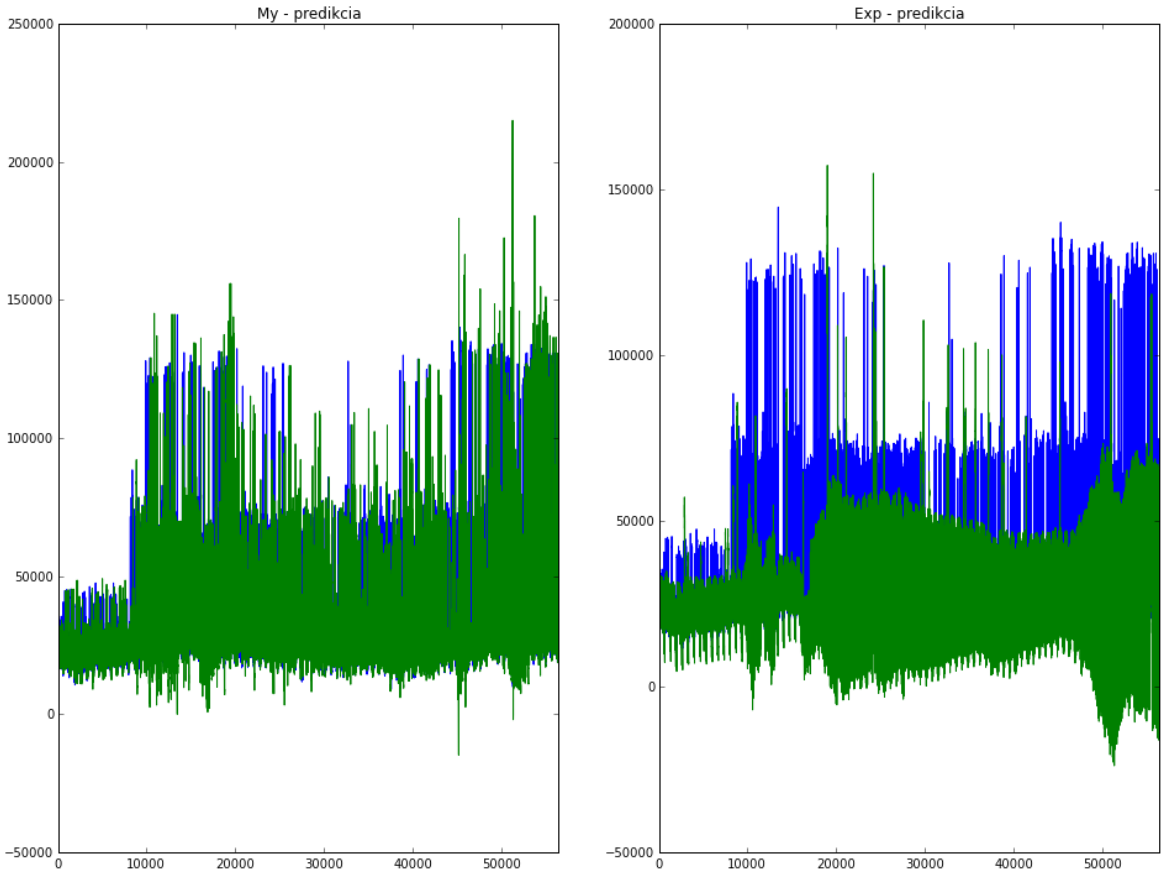


In [101]:

```
# zelena je predikcia a modra su skutocne data
fig, axs = plt.subplots(1,2)
p_start = start*48
p_end = end*48
mt_target.reset_index()[2][p_start:p_end].plot(ax=axs[0])
mt_my_pred.reset_index()[1][p_start:p_end].plot(ax=axs[0], title='My - predikcia')
mt_target.reset_index()[2][p_start:p_end].plot(ax=axs[1])
mt_exp_pred.reset_index()[1][p_start:p_end].plot(ax=axs[1], title='Exp - predikcia')
```

Out[101]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d5a94810>
```



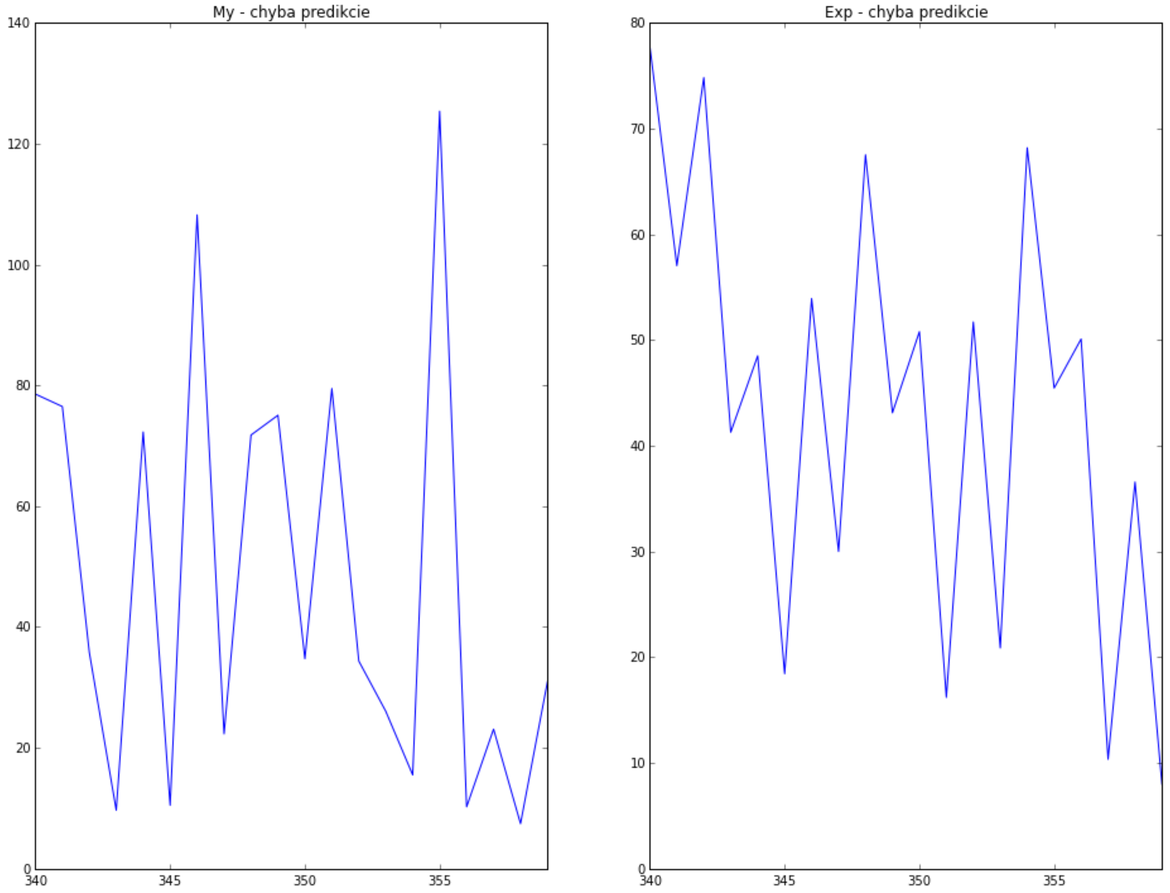
Na prvy pohlad sa zda, ze sa exponencialne vyrovnavanie nevie vysporiadat s velkymi zmenami v amplitude

In [104]:

```
fig, axs = plt.subplots(1,2)
start = 340 # minimalne 0
end = 360 #maximalne 1177
mt_my_smape.reset_index()[1][start:end].plot(ax=axs[0], title='My - chyba predikcie')
mt_exp_smape.reset_index()[1][start:end].plot(ax=axs[1], title='Exp - chyba predikcie
```

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d59c73d0>
```

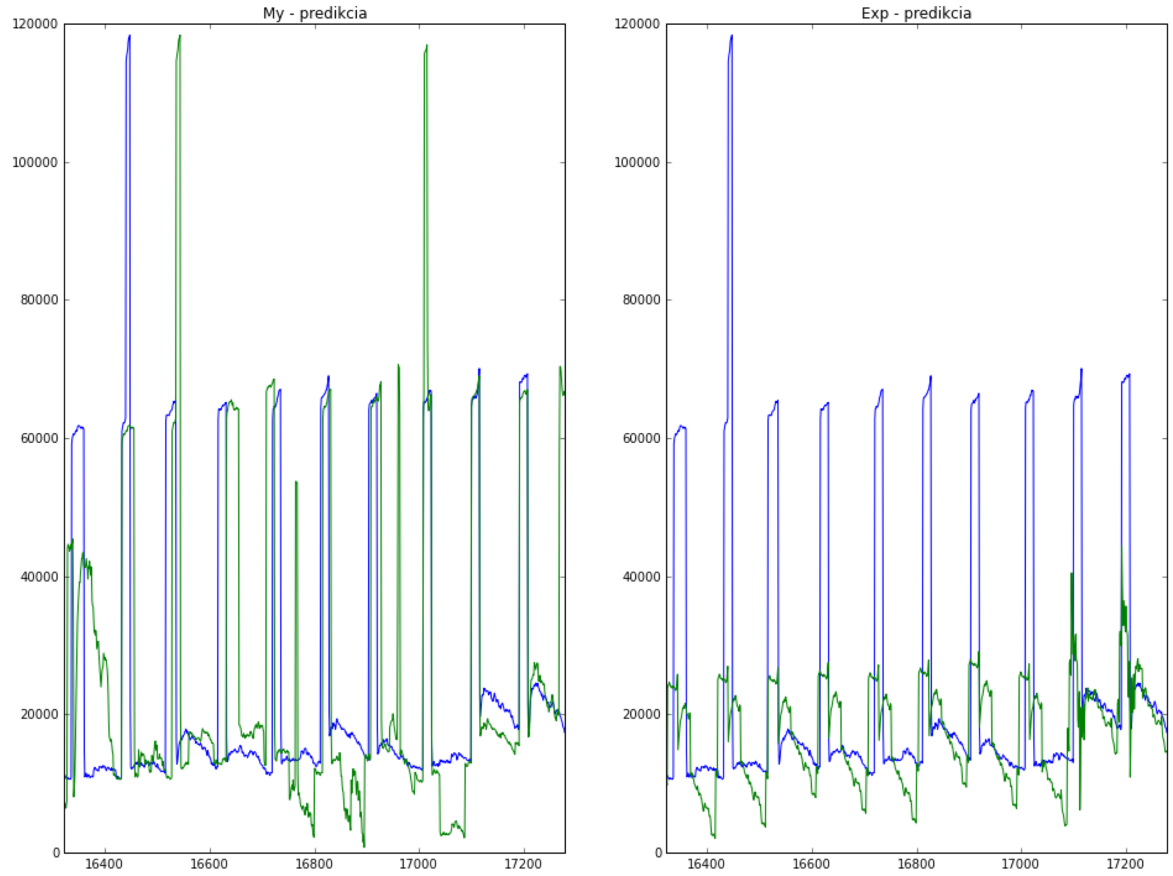


In [105]:

```
# zelena je predikcia a modra su skutocne data
fig, axs = plt.subplots(1,2)
p_start = start*48
p_end = end*48
mt_target.reset_index()[2][p_start:p_end].plot(ax=axs[0])
mt_my_pred.reset_index()[1][p_start:p_end].plot(ax=axs[0], title='My - predikcia')
mt_target.reset_index()[2][p_start:p_end].plot(ax=axs[1])
mt_exp_pred.reset_index()[1][p_start:p_end].plot(ax=axs[1], title='Exp - predikcia')
```

Out[105]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4d58350d0>
```



V datach je velmi vela vzorov a hlavne su velmi velke vykivy medzi jednotlivymi dnami a v priebehu dna. Obe metody tu funguju dost zle. Exponencialne vyrovnavanie sa nedokaze naucit nove vzory a stale opakuje jeden, ktory pomaly deformuje. Navrhovana metoda sa zas snazi najst najlepsi vzor v historii, ale castokrat sa stane, ze sa vytiahne vzor, ktory sa vyskytol raz a uz nikdy viac. To sposobi, ze napriek tomu, ze sa vyberie najpodobnejsi vzor na zaklade jeho zaciatku, je velka sanca, ze to pokracovanie uz nebude sediet. Chcelo by to teda zolzitejsiu metodu, pouzivajucu symboly ktora by ratala s frekvenciou vyskytov symbolov a/alebo by sa pozerala aj dalej do minulosti.

Hlavnou vyhodou navrhovanej metody je teda to, ze sa dokaze velmi rychlo prisposobit novym datam

Boxplot chyb pre rozne datasety a rozne metody

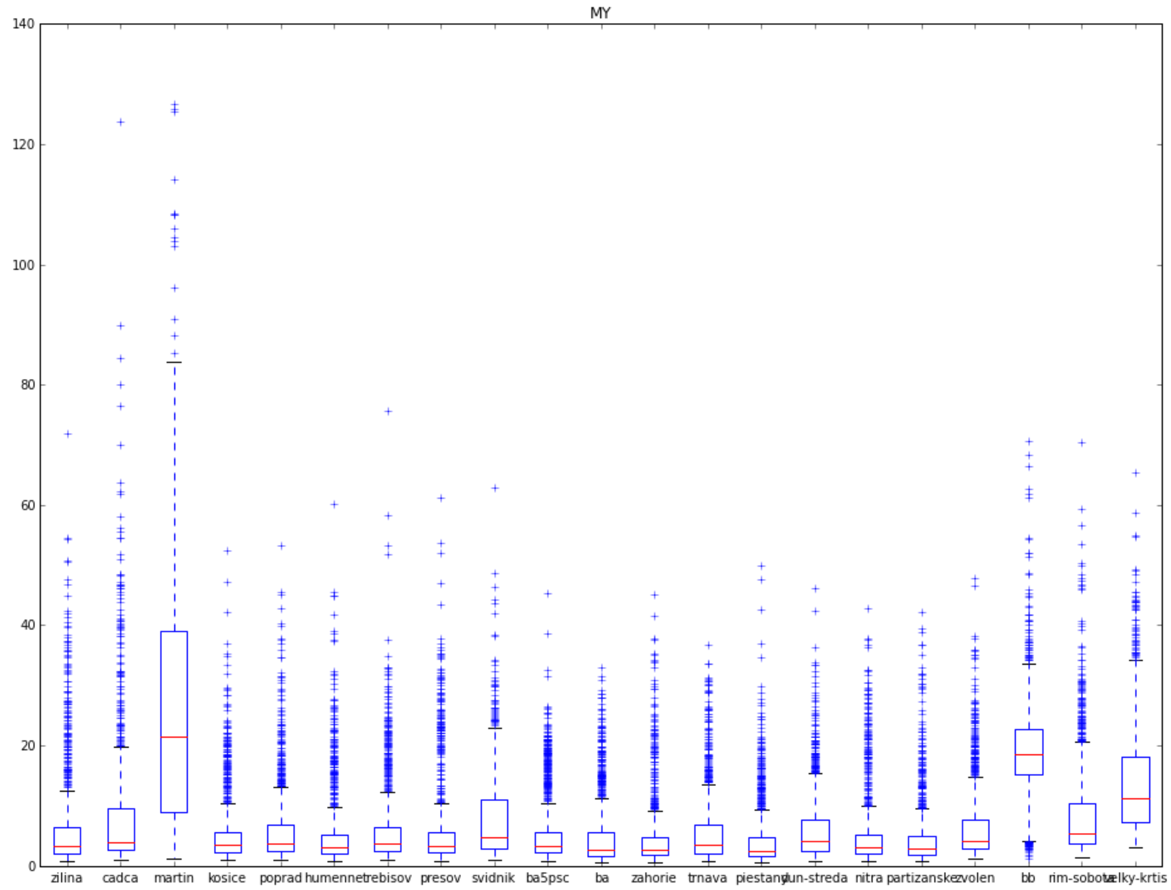
Este zopar boxplotov chyb na roznych metodach na to, aby sa dali porovnat aj kvanitly, maximalne hodnoty a vynimky.

In [19]:

```
my_data = pd.DataFrame()
for filename in glob.glob('./MY/my_smape*.csv'):
    dataset_name = filename.split("_")[-2]
    tmp_df = pd.DataFrame.from_csv(filename, header=None)
    my_data[dataset_name] = tmp_df.reset_index()[1]
my_data.plot(kind="box", title="MY")
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4e17fc490>
```



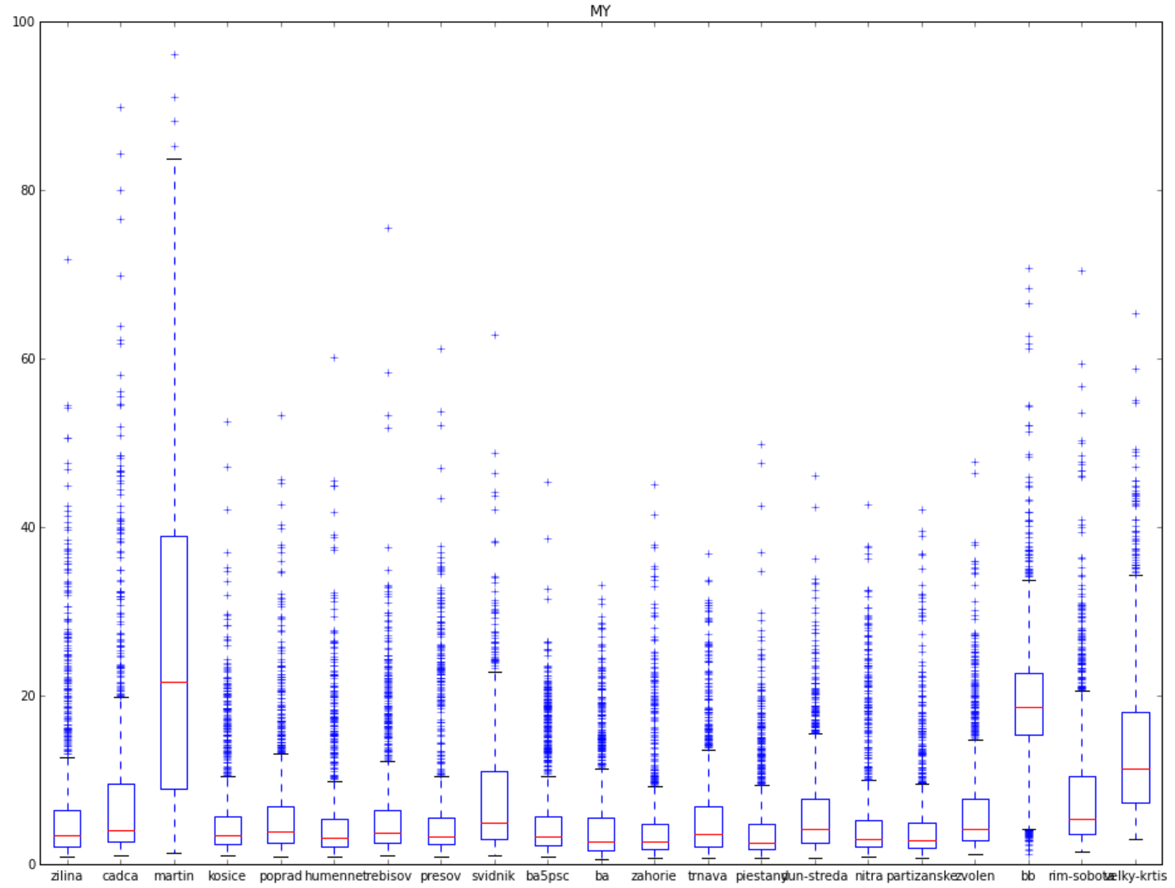
To iste, len maximum y-ovej osi je nastavene na 100

In [24]:

```
my_data.plot(kind="box", title="MY")
plt.ylim((0,100))
```

Out[24]:

```
(0, 100)
```

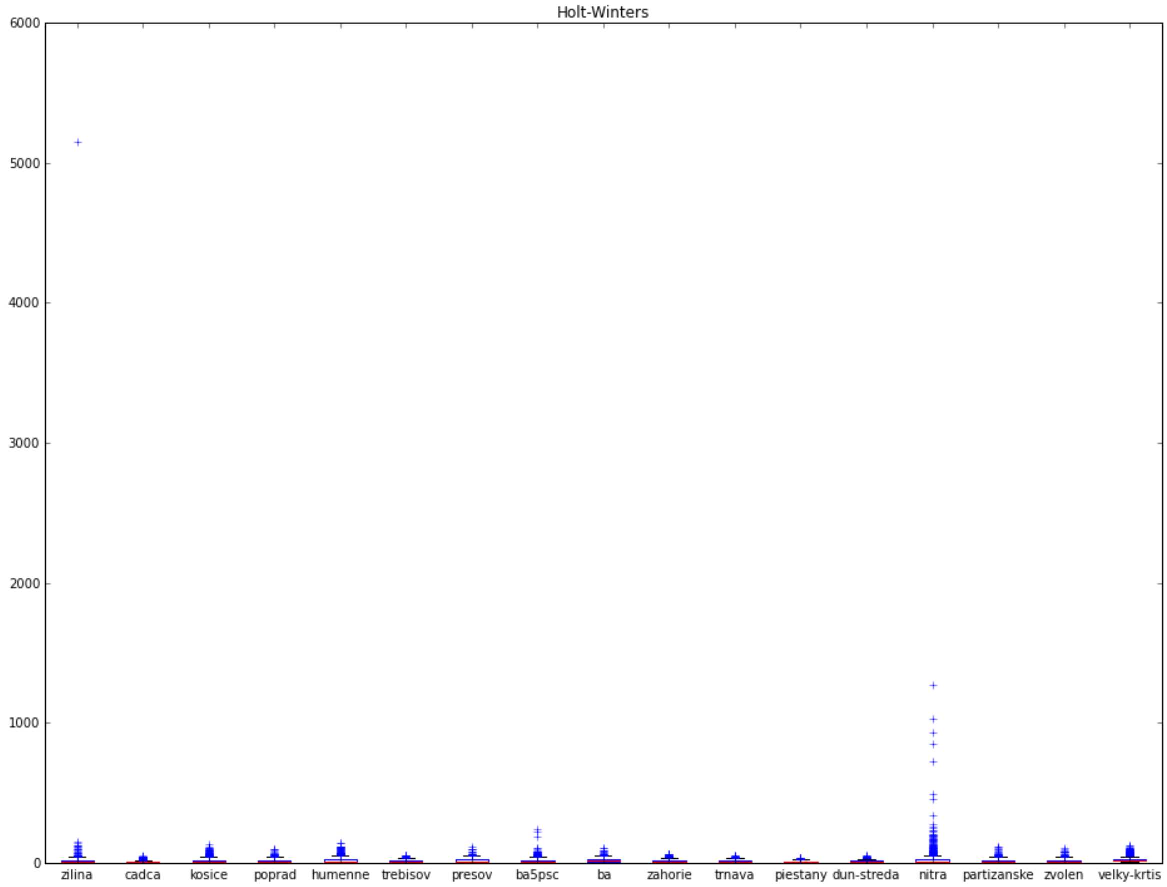


In [20]:

```
hw_data = pd.DataFrame()
for filename in glob.glob('./hw/smape*.csv'):
    dataset_name = filename.split("_")[-2]
    tmp_df = pd.DataFrame.from_csv(filename, header=None)
    hw_data[dataset_name] = tmp_df.reset_index()[1]
hw_data.plot(kind="box", title="Holt-Winters")
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4e13dc250>
```



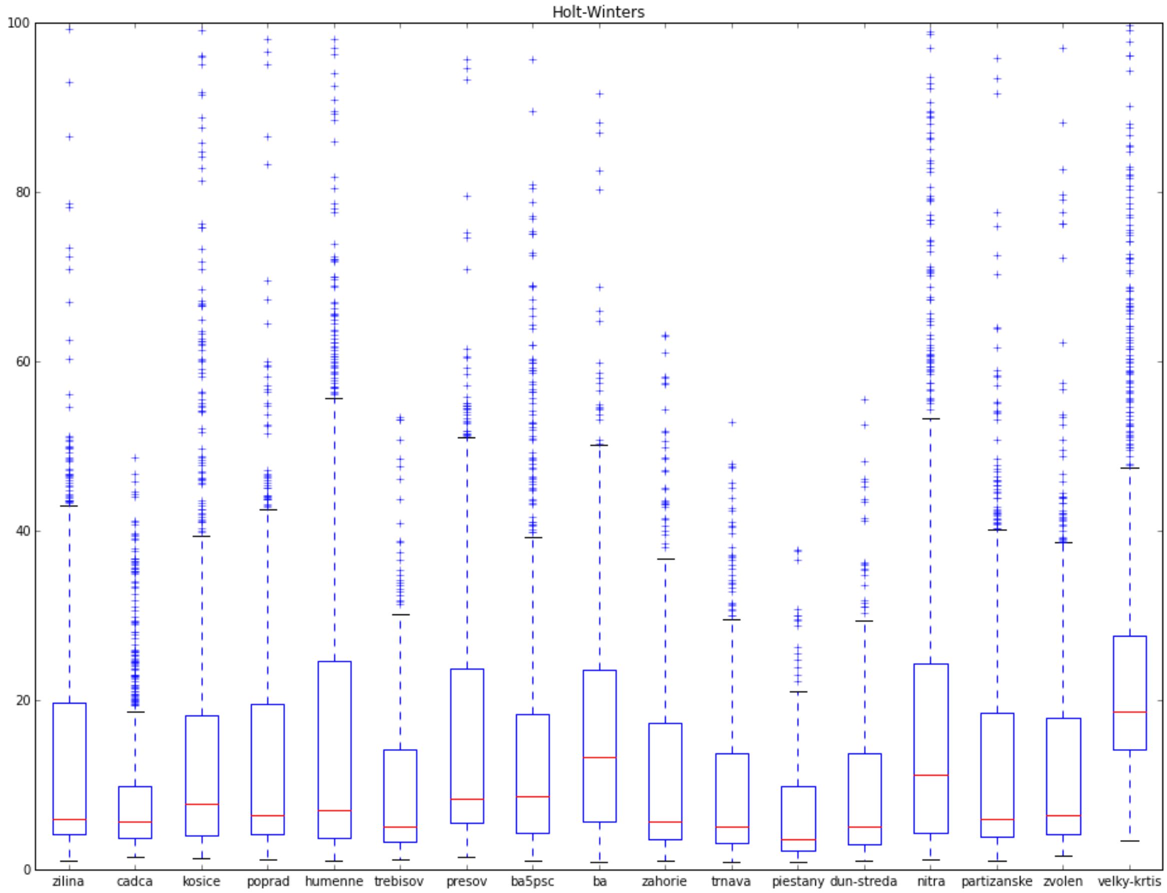
To iste, len maximum y-ovej osi je nastavene na 100

In [23]:

```
hw_data.plot(kind="box", title="Holt-Winters")
plt.ylim((0,100))
```

Out[23]:

```
(0, 100)
```

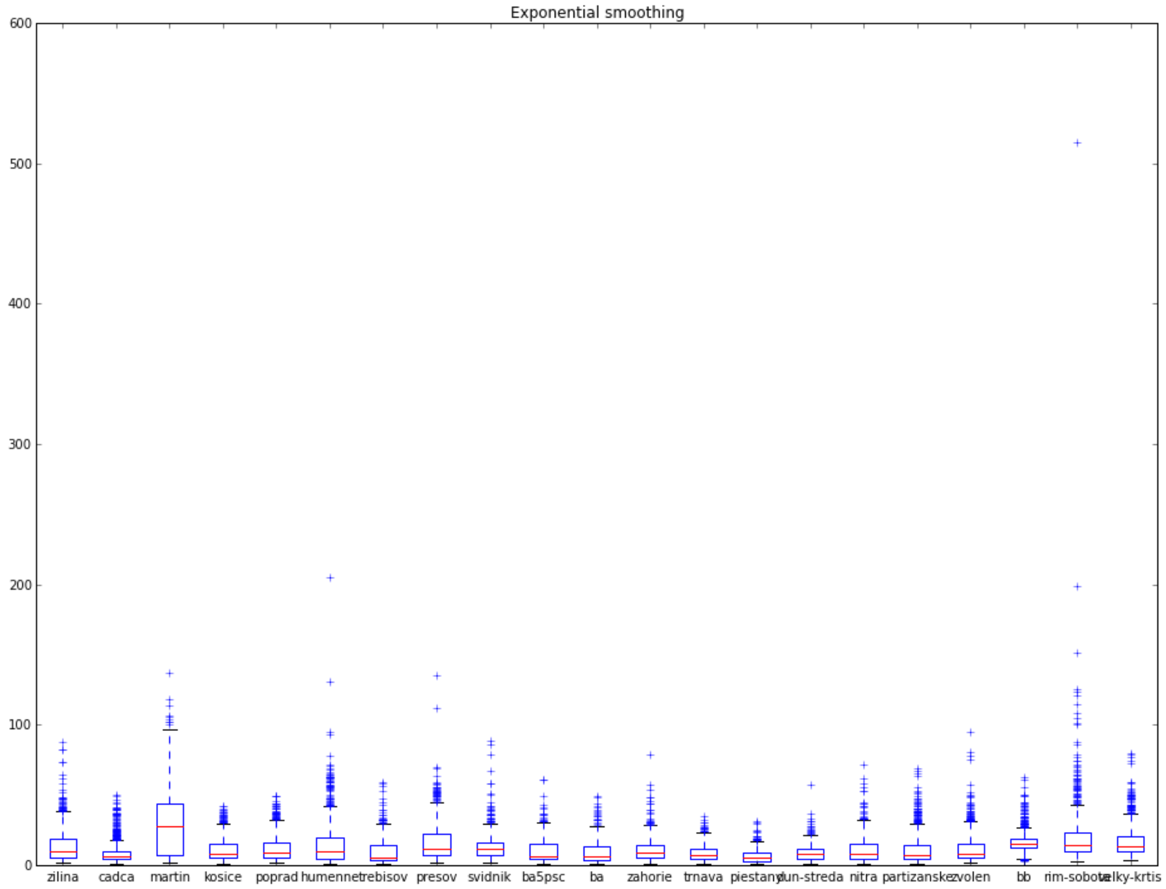


In [25]:

```
hw_false_data = pd.DataFrame()
for filename in glob.glob('./HWbetaFALSE/smape*.csv'):
    dataset_name = filename.split("_")[-2]
    tmp_df = pd.DataFrame.from_csv(filename, header=None)
    hw_false_data[dataset_name] = tmp_df.reset_index()[1]
hw_false_data.plot(kind="box", title="Exponential smoothing")
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4e073e490>
```



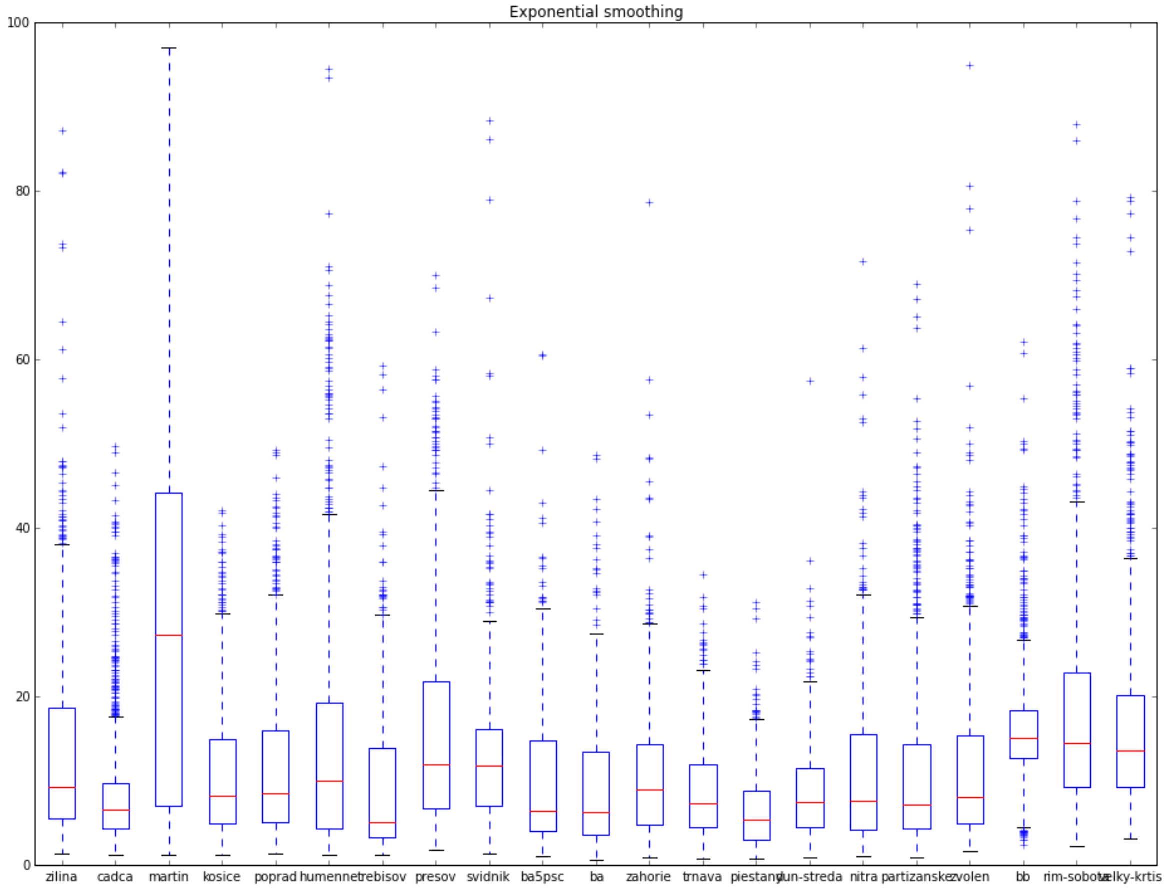
To iste, len maximum y-ovej osi je nastavene na 100

In [27]:

```
hw_false_data.plot(kind="box", title="Exponential smoothing")
plt.ylim((0,100))
```

Out[27]:

```
(0, 100)
```


Exponential smoothing
zilina
cadca
martin
kosice
poprad
humenne
trebisov
presov
svidnik
ba5psc
ba
zahorie
trnava
piestany
dun-streda
nitra
partizanske
zvolen
bb
rim-sobota
velky-krtis